Package ‘VGAM’

May 11, 2015

Version 0.9-8
Date 2015-05-11
Title Vector Generalized Linear and Additive Models
Author Thomas W. Yee <t.yee@auckland.ac.nz>
Maintainer Thomas Yee <t.yee@auckland.ac.nz>
Depends R (>= 3.0.0), methods, stats, stats4, splines
Suggests VGAMdata, MASS
Description An implementation of about 6 major classes of statistical regression models. At the heart of it are the vector generalized linear and additive model (VGLM/VGAM) classes. Currently only fixed-effects models are implemented, i.e., no random-effects models. Many (150+) models and distributions are estimated by maximum likelihood estimation (MLE) or penalized MLE, using Fisher scoring. VGLMs can be loosely thought of as multivariate GLMs. VGAMs are data-driven VGLMs (i.e., with smoothing). The other classes are RR-VGLMs (reduced-rank VGLMs), quadratic RR-VGLMs, reduced-rank VGAMs, RCIMs (row-column interaction models)---these classes perform constrained and unconstrained quadratic ordination (CQO/UQO) models in ecology, as well as constrained additive ordination (CAO). Note that these functions are subject to change, especially before version 1.0.0 is released; see the NEWS file for latest changes.
License GPL-2
URL https://www.stat.auckland.ac.nz/~yee/VGAM
NeedsCompilation yes
BuildVignettes yes
LazyLoad yes
LazyData yes
Repository CRAN
Date/Publication 2015-05-11 11:14:54
R topics documented:

- VGAM-package
- A1A2A3
- AA.Aa.aa
- AB.Ab.aB.ab
- ABO
- aca
- AICvlm
- alaplace
- alaplaceUC
- ambinomial
- amexp
- amnorm
- ampoisson
- AR1
- auuc
- aux.posbernoulli.t
- backPain
- beggs
- Benford
- Benini
- Benini1
- Betabinom
- betabinomial
- betabinomialff
- betaf
- Betageom
- betageometric
- betaii
- Betanorm
- betaprime
- betaR
- Biamhcop
- biamhcop
- Biclaytoncop
- biclaytoncop
- BICvlm
- Bifgmcop
- bifgmcop
- bifgme
- bifrankcop
- bigammain
- bigumbell
- bilogis
- bilogistic
- Binom2.or
- binom2.or
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Binom2.rho</td>
<td>93</td>
</tr>
<tr>
<td>binom2.rho</td>
<td>94</td>
</tr>
<tr>
<td>binomialf</td>
<td>97</td>
</tr>
<tr>
<td>Binorm</td>
<td>100</td>
</tr>
<tr>
<td>binormal</td>
<td>102</td>
</tr>
<tr>
<td>binormalcop</td>
<td>103</td>
</tr>
<tr>
<td>Binormcop</td>
<td>105</td>
</tr>
<tr>
<td>Biplackett</td>
<td>106</td>
</tr>
<tr>
<td>biplackettcp</td>
<td>108</td>
</tr>
<tr>
<td>biplot-methods</td>
<td>109</td>
</tr>
<tr>
<td>Bisa</td>
<td>110</td>
</tr>
<tr>
<td>bisa</td>
<td>111</td>
</tr>
<tr>
<td>Bistudentt</td>
<td>113</td>
</tr>
<tr>
<td>bistudentt</td>
<td>114</td>
</tr>
<tr>
<td>bmi.nz</td>
<td>116</td>
</tr>
<tr>
<td>borel.tanner</td>
<td>117</td>
</tr>
<tr>
<td>Bort</td>
<td>118</td>
</tr>
<tr>
<td>Brat</td>
<td>119</td>
</tr>
<tr>
<td>brat</td>
<td>121</td>
</tr>
<tr>
<td>bratt</td>
<td>123</td>
</tr>
<tr>
<td>calibrate</td>
<td>125</td>
</tr>
<tr>
<td>calibrate-methods</td>
<td>126</td>
</tr>
<tr>
<td>calibrate.qrrvglm</td>
<td>126</td>
</tr>
<tr>
<td>calibrate.qrrvglm.control</td>
<td>129</td>
</tr>
<tr>
<td>cao</td>
<td>130</td>
</tr>
<tr>
<td>cao.control</td>
<td>134</td>
</tr>
<tr>
<td>Card</td>
<td>138</td>
</tr>
<tr>
<td>cardioid</td>
<td>139</td>
</tr>
<tr>
<td>cauchit</td>
<td>141</td>
</tr>
<tr>
<td>cauchy</td>
<td>143</td>
</tr>
<tr>
<td>cdf.lmscreg</td>
<td>145</td>
</tr>
<tr>
<td>cens.gumbel</td>
<td>146</td>
</tr>
<tr>
<td>cens.normal</td>
<td>148</td>
</tr>
<tr>
<td>cens.poisson</td>
<td>150</td>
</tr>
<tr>
<td>cfibrosis</td>
<td>152</td>
</tr>
<tr>
<td>ego</td>
<td>153</td>
</tr>
<tr>
<td>chest.nz</td>
<td>154</td>
</tr>
<tr>
<td>chinese.nz</td>
<td>155</td>
</tr>
<tr>
<td>chisq</td>
<td>156</td>
</tr>
<tr>
<td>clo</td>
<td>157</td>
</tr>
<tr>
<td>cloglog</td>
<td>158</td>
</tr>
<tr>
<td>coalminers</td>
<td>160</td>
</tr>
<tr>
<td>Coef</td>
<td>161</td>
</tr>
<tr>
<td>Coef.qrrvglm</td>
<td>162</td>
</tr>
<tr>
<td>Coef.qrrvglm-class</td>
<td>164</td>
</tr>
<tr>
<td>Coef.rvglm</td>
<td>166</td>
</tr>
<tr>
<td>Coef.rvglm-class</td>
<td>167</td>
</tr>
<tr>
<td>Coef.vlm</td>
<td>168</td>
</tr>
</tbody>
</table>
R topics documented:

- coefvlm .............................................................. 169
- CommonVGAMffArguments ........................................ 170
- concoef ............................................................... 176
- concoef-methods .................................................... 178
- constraints .......................................................... 178
- corbet ................................................................. 180
- cqo ...................................................................... 181
- crashes ............................................................... 188
- cratio ................................................................. 190
- cumulative ............................................................ 191
- Dagum ................................................................. 195
- dagum ................................................................. 196
- dAR1 ................................................................. 198
- deermice ............................................................. 200
- depplot.lmscreg ...................................................... 201
- depvar ............................................................... 202
- df.residual .......................................................... 203
- dgenpois ............................................................. 204
- dhuber ............................................................... 206
- dirichlet ............................................................. 207
- dirmul.old .......................................................... 209
- dirmultinomial ....................................................... 211
- dlogF ................................................................. 213
- double.cens.normal ................................................ 214
- double.expbinomial ............................................... 216
- ducklings ............................................................ 218
- enzyme ............................................................... 219
- erf ................................................................. 220
- erlang .............................................................. 222
- Expectiles-Exponential .......................................... 223
- Expectiles-Normal ................................................ 224
- Expectiles-sc.t2 .................................................... 226
- Expectiles-Uniform ................................................ 228
- expexpff ............................................................ 230
- expexpff1 ........................................................... 232
- expgeom ............................................................ 234
- expgeometric ....................................................... 235
- expint .............................................................. 237
- explink ............................................................... 238
- explog .............................................................. 239
- explogff ............................................................ 241
- exponential ........................................................ 242
- exppois ............................................................. 244
- exppoisson .......................................................... 245
- Felix ................................................................. 246
- felix ................................................................. 247
- fff ................................................................. 249
- fill ................................................................. 250
R topics documented:

finney44 ........................................ 254
fisherz ........................................ 255
Fisk ........................................ 256
fisk ........................................ 257
fittedvlm ........................................ 259
flourbeetle ..................................... 261
Foldnorm ......................................... 262
foldnormal ....................................... 263
foldsqrt ......................................... 265
Frank ........................................ 267
Frechet ......................................... 268
frechet ......................................... 270
freund61 ......................................... 271
gamma1 ........................................ 274
gamma2 ........................................ 275
gammahyperbola .................................. 277
gammaR ......................................... 278
garma ........................................ 280
gaussianff ....................................... 282
GenbetaII ........................................ 284
genbetaII ......................................... 285
gengamma.stacy .................................. 288
gengammaUC ...................................... 290
genpoisson ....................................... 291
genray ........................................ 293
genrayleigh ...................................... 294
geometric ........................................ 296
get.smart ........................................ 297
get.smart.prediction ............................. 298
gev ........................................ 299
gevUC ........................................ 302
gew ........................................ 304
golf ........................................ 305
Gompertz ........................................ 307
gompertz ........................................ 308
gpd ........................................ 310
gpdUC ........................................ 313
grain.us ......................................... 315
grc ........................................ 316
gumbel .......................................... 321
Gumbel-II ........................................ 324
gumbelII .......................................... 326
gumbelUC .......................................... 327
guplot ........................................ 329
hatvalues ....................................... 331
hormone ......................................... 333
hspider ......................................... 335
huber2 .......................................... 336
R topics documented:

Huggins89.t1 ......................................................... 338
hunua ................................................................. 340
hyperg ............................................................... 342
hypersecant ......................................................... 343
Hzeta ................................................................. 345
hzeta ................................................................. 346
iam ................................................................. 348
identitylink ......................................................... 349
inv.binomial ....................................................... 350
Inv.gaussian ....................................................... 352
inv.gaussianff ..................................................... 353
Inv.lomax .......................................................... 355
inv.lomax .......................................................... 356
Inv.paralogistic ................................................... 358
inv.paralogistic ................................................... 359
is.buggy ........................................................... 361
is.parallel ........................................................ 362
is.smart ........................................................... 363
is.zero .............................................................. 364
kendall.tau ........................................................ 365
Kumar ............................................................... 366
kumar ............................................................... 368
lakeO ................................................................. 369
lambertW ............................................................ 370
laplace .............................................................. 371
laplaceUC .......................................................... 373
latvar ............................................................... 375
leipnik ............................................................. 376
lerch ................................................................. 378
leukemia ............................................................ 379
levy ................................................................. 380
lgamma1 ............................................................. 381
lgammaUC .......................................................... 383
Lindley .............................................................. 385
lindley .............................................................. 386
linkfun ............................................................. 387
linkfun.vglm ....................................................... 388
Links ............................................................... 389
Lino ................................................................. 392
lino ................................................................. 393
lirat ................................................................. 395
lms.bcg ............................................................. 396
lms.bcn ............................................................. 398
lms.yjn ............................................................. 401
Log ................................................................. 404
log1pexp ........................................................... 405
loge ................................................................. 406
loge ................................................................. 407
R topics documented:

logF ......................................................... 409
logff ......................................................... 410
logistic ..................................................... 412
logit ......................................................... 413
loglaplace .................................................. 416
loglapUC .................................................... 419
logLik.vlm .................................................. 421
loglinb2 ..................................................... 422
loglinb3 ..................................................... 424
loglog ......................................................... 426
lognormal .................................................... 427
logoff ......................................................... 428
Lomax ......................................................... 429
lomax ......................................................... 431
lqnorm ......................................................... 433
lrtest ......................................................... 435
lvplot ......................................................... 436
lvplot.qrrvglm .............................................. 438
lvplot.rrvglm ............................................... 442
machinists .................................................. 445
Makeham ..................................................... 446
makeham ..................................................... 447
margeff ....................................................... 449
marital.nz .................................................... 451
Max ......................................................... 452
Maxwell ....................................................... 453
maxwell ....................................................... 454
mccullagh89 ............................................... 456
melbmaxtemp ............................................... 457
meplot ......................................................... 458
micmen ....................................................... 460
mix2exp ....................................................... 462
mix2normal .................................................. 464
mix2poisson .................................................. 466
MNSs ......................................................... 468
model.framevlm .............................................. 470
model.matrixvlm ............................................ 471
moffset ...................................................... 473
multilogit .................................................... 474
multinomial .................................................. 476
Nakagami .................................................... 480
nakagami ..................................................... 481
nbcanlink .................................................... 483
nbolf ......................................................... 485
negbinomial .................................................. 487
negbinomial.size .......................................... 493
normal.vcm .................................................. 494
nparam.vlm .................................................. 498
R topics documented:
olympics ................................................................. 499
Opt ................................................................. 500
ordpoisson ........................................................... 502
oxtemp ............................................................... 504
Paralogistic .......................................................... 505
paralogistic .......................................................... 506
Pareto ............................................................... 508
paretoff ............................................................ 509
ParetoIV ............................................................. 511
paretoIV ............................................................ 513
Perks ................................................................. 516
perks ................................................................. 517
perspqrrvglm .......................................................... 519
pgamma.deriv .......................................................... 522
pgamma.deriv.unscaled ............................................... 523
plotdeplot.lmscreg ......................................................... 524
plotqrrvglm ............................................................ 526
plotqtlplot.lmscreg ....................................................... 527
plotrcim0 ............................................................. 529
plotvgam .............................................................. 531
plotvgam.control ......................................................... 534
plotvglm .............................................................. 535
pneumo ............................................................... 537
poisson.points .......................................................... 538
poissonff .............................................................. 539
PoissonPoints .......................................................... 542
polf ................................................................. 543
Polono ............................................................... 545
posbernoulli.b .......................................................... 547
posbernoulli.t .......................................................... 549
posbernoulli.tb .......................................................... 552
posbernUC ............................................................ 555
Posbinom ............................................................. 557
posbinomial .......................................................... 559
Posgeom ............................................................. 561
Posnegbin ............................................................ 562
posnegbinomial .......................................................... 564
Posnorm ............................................................. 566
posnormal ............................................................ 568
Pospois .............................................................. 569
pospoisson ........................................................... 571
powerlink ............................................................ 572
prats ................................................................. 573
predictqrrvglm .......................................................... 575
predictvglm ............................................................ 576
prentice74 ............................................................ 578
prinia ................................................................. 580
probit ................................................................. 581
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>propodds</td>
<td>583</td>
</tr>
<tr>
<td>prplot</td>
<td>584</td>
</tr>
<tr>
<td>put.smart</td>
<td>585</td>
</tr>
<tr>
<td>qrrvglm.control</td>
<td>586</td>
</tr>
<tr>
<td>qtplot.gumbel</td>
<td>592</td>
</tr>
<tr>
<td>qtplot.lmscreg</td>
<td>594</td>
</tr>
<tr>
<td>quasibinomialff</td>
<td>595</td>
</tr>
<tr>
<td>quasipoissonff</td>
<td>597</td>
</tr>
<tr>
<td>Qvar</td>
<td>599</td>
</tr>
<tr>
<td>qvar</td>
<td>602</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>603</td>
</tr>
<tr>
<td>rayleigh</td>
<td>605</td>
</tr>
<tr>
<td>Reim</td>
<td>607</td>
</tr>
<tr>
<td>reqo</td>
<td>608</td>
</tr>
<tr>
<td>rdirc</td>
<td>612</td>
</tr>
<tr>
<td>rec.exp1</td>
<td>614</td>
</tr>
<tr>
<td>rec.normal</td>
<td>615</td>
</tr>
<tr>
<td>reciprocal</td>
<td>616</td>
</tr>
<tr>
<td>rhobit</td>
<td>618</td>
</tr>
<tr>
<td>Rice</td>
<td>619</td>
</tr>
<tr>
<td>riceff</td>
<td>620</td>
</tr>
<tr>
<td>rigff</td>
<td>622</td>
</tr>
<tr>
<td>rlplot.egev</td>
<td>623</td>
</tr>
<tr>
<td>rrar</td>
<td>625</td>
</tr>
<tr>
<td>rrvglm</td>
<td>627</td>
</tr>
<tr>
<td>rrvglm-class</td>
<td>630</td>
</tr>
<tr>
<td>rrvglm.control</td>
<td>633</td>
</tr>
<tr>
<td>rrvglm.optim.control</td>
<td>636</td>
</tr>
<tr>
<td>ruge</td>
<td>637</td>
</tr>
<tr>
<td>s</td>
<td>638</td>
</tr>
<tr>
<td>sc.studentt2</td>
<td>639</td>
</tr>
<tr>
<td>Select</td>
<td>641</td>
</tr>
<tr>
<td>seq2binomial</td>
<td>643</td>
</tr>
<tr>
<td>setup.smart</td>
<td>645</td>
</tr>
<tr>
<td>Simplex</td>
<td>646</td>
</tr>
<tr>
<td>simplex</td>
<td>647</td>
</tr>
<tr>
<td>simulate.vlm</td>
<td>649</td>
</tr>
<tr>
<td>Sinmad</td>
<td>651</td>
</tr>
<tr>
<td>sinnad</td>
<td>652</td>
</tr>
<tr>
<td>Skellam</td>
<td>654</td>
</tr>
<tr>
<td>skellam</td>
<td>655</td>
</tr>
<tr>
<td>skewnorm</td>
<td>657</td>
</tr>
<tr>
<td>skewnormal</td>
<td>658</td>
</tr>
<tr>
<td>Slash</td>
<td>660</td>
</tr>
<tr>
<td>slash</td>
<td>661</td>
</tr>
<tr>
<td>smart.expression</td>
<td>663</td>
</tr>
<tr>
<td>smart.mode.is</td>
<td>664</td>
</tr>
<tr>
<td>smartpred</td>
<td>665</td>
</tr>
</tbody>
</table>
### R topics documented:

- `sratio` .................................................. 667
- `studentt` ............................................. 669
- `SURff` ................................................ 671
- `SurvS4` ............................................... 673
- `SurvS4-class` ....................................... 675
- `Tikuv` ................................................ 676
- `tikuv` ................................................ 678
- `Tobit` ................................................ 679
- `tobit` ................................................ 681
- `Tol` .................................................. 685
- `toxop` ............................................... 687
- `Triangle` ............................................. 688
- `triangle` .......................................... 689
- `trplot` ............................................... 691
- `trplot.qrrvglm` .................................... 692
- `Truncpareto` ....................................... 694
- `truncweibull` ....................................... 696
- `ucberk` ............................................. 698
- `uninormal` .......................................... 699
- `V1` .................................................. 700
- `venice` ............................................. 702
- `vgam` ............................................... 703
- `vgam-class` ......................................... 707
- `vgam.control` ....................................... 710
- `vglm` ............................................... 712
- `vglm-class` ......................................... 717
- `vglm.control` ....................................... 720
- `vglmff-class` ....................................... 723
- `vonmises` ........................................... 726
- `vsmooth.spline` .................................... 728
- `waitakere` .......................................... 730
- `waldff` ............................................. 732
- `weibullR` ........................................... 733
- `weightsvglm` ........................................ 736
- `wine` ................................................ 737
- `wrapup.smart` ....................................... 738
- `yeo.johnson` ........................................ 739
- `yip88` ............................................... 740
- `Yules` ............................................... 743
- `yulesimon` ......................................... 744
- `Zabinom` ............................................ 745
- `zabinomial` ......................................... 746
- `Zageom` ............................................. 748
- `zageometric` ........................................ 750
- `Zanegbin` ........................................... 752
- `zanegbinomial` ...................................... 753
- `Zapois` ............................................. 756
- `zapoison` .......................................... 757
VGAM-package

Description

VGAM provides functions for fitting vector generalized linear and additive models (VGLMs and VGAMs), and associated models (Reduced-rank VGLMs, Quadratic RR-VGLMs, Reduced-rank VGAMs). This package fits many models and distributions by maximum likelihood estimation (MLE) or penalized MLE. Also fits constrained ordination models in ecology such as constrained quadratic ordination (CQO).

Details

This package centers on the iteratively reweighted least squares (IRLS) algorithm. Other key words include Fisher scoring, additive models, penalized likelihood, reduced-rank regression and constrained ordination. The central modelling functions are `vglm`, `vgam`, `rrvglm`, `rcim`, `cqo`, `cao`. For detailed control of fitting, each of these has its own control function, e.g., `vglm.control`. The package uses S4 (see `methods-package`). A companion package called `VGAMdata` contains some larger data sets which were shifted from VGAM.

The classes of GLMs and GAMs are special cases of VGLMs and VGAMs. The VGLM/VGAM framework is intended to be very general so that it encompasses as many distributions and models as possible. VGLMs are limited only by the assumption that the regression coefficients enter through a set of linear predictors. The VGLM class is very large and encompasses a wide range of multivariate response types and models, e.g., it includes univariate and multivariate distributions, categorical data analysis, time series, survival analysis, generalized estimating equations, extreme values, correlated binary data, quantile and expectile regression, bioassay data and nonlinear least-squares problems.

VGAMs are to VGLMs what GAMs are to GLMs. Vector smoothing (see `vsmooth.spline`) allows several additive predictors to be estimated as a sum of smooth functions of the covariates.
For a complete list of this package, use `library(help = "VGAM")`. New VGAM family functions are continually being written and added to the package. A monograph about VGLM and VGAMs etc. is currently in the making.

Warning

This package is undergoing continual development and improvement. Until my monograph comes out and this package is released as version 1.0-0 the user should treat everything subject to change. This includes the family function names, argument names, many of the internals, the use of link functions, and slot names. Some future pain can be minimized by using good programming techniques, e.g., using extractor/accessor functions such as `coef()`, `weights()`, `vcov()`, `predict()`. Nevertheless, please expect changes in all aspects of the package. See the NEWS file for a list of changes from version to version.

Author(s)

Thomas W. Yee, `<t.yee@auckland.ac.nz>`.
Maintainer: Thomas Yee `<t.yee@auckland.ac.nz>`.

References


My website for the VGAM package is at [https://www.stat.auckland.ac.nz/~yee/VGAM](https://www.stat.auckland.ac.nz/~yee/VGAM) and I hope to put more resources there in the future, especially as relating to my book.

See Also

`vglm, vgam, rrvglm, rcim, cqq, TypicalVGAMfamilyFunction, CommonVGAMffArguments, Links.`
Examples

# Example 1; proportional odds model
pneumo <- transform(pneumo, let = log(exposure.time))
(fit1 <- vglm(cbind(normal, mild, severe) ~ let, propodds, data = pneumo))
depvar(fit1) # Better than using fit0y; dependent variable (response)
weights(fit1, matrix = TRUE) # Number of observations
coef(fit1, matrix = TRUE) # Constraint matrices
summary(fit1)

# Example 2; zero-inflated Poisson model
zdata <- data.frame(x2 = runif(nn <- 2000))
zdata <- transform(zdata, pstr0 = logit(-0.5 + 1*x2, inverse = TRUE),
                   lambda = loge (0.5 + 2*x2, inverse = TRUE))
zdata <- transform(zdata, y = rzipois(nn, lambda, pstr0 = pstr0))
with(zdata, table(y))
fit2 <- vglm(y ~ x2, zipoisson, data = zdata, trace = TRUE)
coef(fit2, matrix = TRUE) # These should agree with the above values

# Example 3; fit a two species GAM simultaneously
fit3 <- vgam(cbind(agaaus, kniexc) ~ s(altitude, df = c(2, 3)),
             binomialff(multiple.responses = TRUE), data = hunua)
coef(fit3, matrix = TRUE) # Not really interpretable
## Not run: plot(fit3, se = TRUE, overlay = TRUE, lcol = 3:4, scol = 3:4)

ooo <- with(hunua, order(altitude))
with(hunua, matplot(altitude[ooo], fitted(fit3)[ooo, ], type = "l",
                lwd = 2, col = 3:4,
                xlab = "Altitude (m)", ylab = "Probability of presence", las = 1,
                main = "Two plant species' response curves", ylim = c(0, 0.8)))
with(hunua, rug(altitude))
## End(Not run)

# Example 4; LMS quantile regression
fit4 <- vgam(BMI ~ s(age, df = c(4, 2)), lms.bcn(zero = 1),
             data = bmi.nz, trace = TRUE)
head(predict(fit4))
head(fitted(fit4))
head(bmi.nz) # Person 1 is near the lower quartile among people his age
head(cdf(fit4))

## Not run: par(mfrow = c(1, 1), bty = "l", mar = c(5,4,4,3)+0.1, xpd = TRUE)
qtplot(fit4, percentiles = c(5,50,90,99), main = "Quantiles", las = 1,
       xlim = c(15, 90), ylab = "BMI", lwd = 2, lcol = 4) # Quantile plot

tygrid <- seq(15, 43, len = 100) # BMI ranges
par(mfrow = c(1,1), lwd = 2) # Density plot
aa <- deplot(fit4, x0 = 20, y = ygrid, xlab = "BMI", col = "black",
              main = "Density functions at Age = 20 (black), 42 (red) and 55 (blue)"
The A1A2A3 Blood Group System

Description

Estimates the three independent parameters of the the A1A2A3 blood group system.

Usage

A1A2A3(link = "logit", inbreeding = FALSE, ip1 = NULL, ip2 = NULL, if = NULL)

Arguments

link Link function applied to p1, p2 and f. See Links for more choices.
inbreeding Logical. Is there inbreeding?
ip1, ip2, if Optional initial value for p1, p2 and f.

Details

The parameters p1 and p2 are probabilities, so that p3=1-p1-p2 is the third probability. The parameter f is the third independent parameter if inbreeding = TRUE. If inbreeding = FALSE then f = 0 and Hardy-Weinberg Equilibrium (HWE) is assumed.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.
Note

The input can be a 6-column matrix of counts, with columns corresponding to A1A1, A1A2, A1A3, A2A2, A2A3, A3A3 (in order). Alternatively, the input can be a 6-column matrix of proportions (so each row adds to 1) and the weights argument is used to specify the total number of counts for each row.

Author(s)

T. W. Yee

References


See Also

AA.Aa.aa, AB.Ab.ab, AB0, MNS.

Examples

```r
ymat <- cbind(108, 196, 429, 143, 513, 559)
fit <- vglm(ymat ~ 1, AA2A2(link = probit), trace = TRUE, crit = "coef")
fit <- vglm(ymat ~ 1, AA2A2(link = logit, ip = 0.3, ip2 = 0.3, if = 0.02),
trace = TRUE, crit = "coef")
Coef(fit) # Estimated p1 and p2
rbind(ymat, sum(ymat) * fitted(fit))
sqrt(diag(vcov(fit)))
```

---

**AA.Aa.aa**

*The AA-Aa-aa Blood Group System*

Description

Estimates the parameter of the AA-Aa-aa blood group system, with or without Hardy Weinberg equilibrium.

Usage

```r
AA.Aa.aa(linkp = "logit", linkf = "logit", inbreeding = FALSE, ipA = NULL, ifp = NULL, zero = NULL)
```

Arguments

- `linkp, linkf`: Link functions applied to pA and f. See Links for more choices.
- `ipA, ifp`: Optional initial values for pA and f.
- `inbreeding`: Logical. Is there inbreeding?
- `zero`: See CommonVGAMffArguments for information.
Details

This one or two parameter model involves a probability called \( p_A \). The probability of getting a count in the first column of the input (an AA) is \( p_A^2 p_A \). When \texttt{inbreeding} = \texttt{TRUE}, an additional parameter \( f \) is used. If \texttt{inbreeding} = \texttt{FALSE} then \( f = 0 \) and Hardy-Weinberg Equilibrium (HWE) is assumed.

Value

An object of class "vglmff" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm} and \texttt{vgam}.

Warning

Setting \texttt{inbreeding} = \texttt{FALSE} makes estimation difficult with non-intercept-only models. Currently, this code seems to work with intercept-only models.

Note

The input can be a 3-column matrix of counts, where the columns are AA, Ab and aa (in order). Alternatively, the input can be a 3-column matrix of proportions (so each row adds to 1) and the \texttt{weights} argument is used to specify the total number of counts for each row.

Author(s)

T. W. Yee

References


See Also

\texttt{AB, Ab, ab, ABO, A1A2A3, MNSs}.

Examples

```r
y <- cbind(53, 95, 38)
fit1 <- vglm(y ~ 1, AA.Aa.aa, trace = TRUE)
fit2 <- vglm(y ~ 1, AA.Aa.aa(inbreeding = TRUE), trace = TRUE)
rbind(y, sum(y) * fitted(fit1))
Coeff(fit1) # Estimated pA
Coeff(fit2) # Estimated pA and f
summary(fit1)
```
The AB-Ab-aB-ab Blood Group System

Description

Estimates the parameter of the AB-Ab-aB-ab blood group system.

Usage

AB.Ab.aB.ab(link = "logit", init.p = NULL)

Arguments

link 
Link function applied to p. See Links for more choices.

init.p 
Optional initial value for p.

Details

This one parameter model involves a probability called p.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

The input can be a 4-column matrix of counts, where the columns are AB, Ab, aB and ab (in order). Alternatively, the input can be a 4-column matrix of proportions (so each row adds to 1) and the weights argument is used to specify the total number of counts for each row.

Author(s)

T. W. Yee

References


See Also

AA.Aa.aa, ABO, A1A2A3, MNSs.
Examples

ymat <- cbind(AB=1997, Ab=906, aB=904, ab=32) # Data from Fisher (1925)
fit <- vglm(ymat ~ 1, AB.Ab.aB.ab(link = "identitylink"), trace = TRUE)
fit <- vglm(ymat ~ 1, AB.Ab.aB.ab, trace = TRUE)
rbind(ymat, sum(ymat)*fitted(fit))
coef(fit) # Estimated p
p <- sqrt(4*(fitted(fit)[, 4]))
p*p
summary(fit)

ABO  The ABO Blood Group System

Description

Estimates the two independent parameters of the ABO blood group system.

Usage

ABO(link.pa = "logit", link.pB = "logit", ipA = NULL, ipB = NULL,
ipo = NULL, zero = NULL)

Arguments

link.pa, link.pB  Link functions applied to pa and pB. See Links for more choices.
ipA, ipB, ipO  Optional initial value for pa and pB and pO. A NULL value means values are computed internally.
zero  Details at CommonVGAMArguments.

Details

The parameters pa and pB are probabilities, so that pO=1−pa−pB is the third probability. The probabilities pa and pB correspond to A and B respectively, so that pO is the probability for O. It is easier to make use of initial values for pO than for pB. In documentation elsewhere I sometimes use pa=p, pB=q, pO=r.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

The input can be a 4-column matrix of counts, where the columns are A, B, AB, O (in order). Alternatively, the input can be a 4-column matrix of proportions (so each row adds to 1) and the weights argument is used to specify the total number of counts for each row.
Author(s)
T. W. Yee

References

See Also
AA.Aa.aa, AB.Ab.aB.ab, A1A2A3, MNSs.

Examples
```r
ymat <- cbind(A = 725, B = 258, AB = 72, O = 1073)  # Order matters, not the name
fit <- vglm(ymat - 1, ABO(link.pA = identitylink,
              link.pB = identitylink), trace = TRUE, cri = "coef")
coef(fit, matrix = TRUE)  
Coef(fit)  # Estimated pA and pB
rbind(ymat, sum(ymat) * fitted(fit))
sqrt(diag(vcov(fit)))
```

acat  
*Ordinal Regression with Adjacent Categories Probabilities*

Description
Fits an adjacent categories regression model to an ordered (preferably) factor response.

Usage
```r
acat(link = "loge", parallel = FALSE, reverse = FALSE,
     zero = NULL, whitespace = FALSE)
```

Arguments
- **link**: Link function applied to the ratios of the adjacent categories probabilities. See [Links](Link) for more choices.
- **parallel**: A logical, or formula specifying which terms have equal/unequal coefficients.
- **reverse**: Logical. By default, the linear/additive predictors used are \( \eta_j = \log(P[Y = j + 1]/P[Y = j]) \) for \( j = 1, \ldots, M \). If `reverse` is TRUE then \( \eta_j = \log(P[Y = j]/P[Y = j + 1]) \) will be used.
- **zero**: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \{1,2,\ldots,M\}.
- **whitespace**: See [CommonVGAMffArguments](CommonVGAMffArguments) for information.
Details
In this help file the response \( Y \) is assumed to be a factor with ordered values 1, 2, \ldots, \( M + 1 \), so that \( M \) is the number of linear/additive predictors \( \eta_j \).

By default, the log link is used because the ratio of two probabilities is positive.

Value
An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning
No check is made to verify that the response is ordinal if the response is a matrix; see ordered.

Note
The response should be either a matrix of counts (with row sums that are all positive), or an ordered factor. In both cases, the y slot returned by vglm/vgam/rrvglm is the matrix of counts.

For a nominal (unordered) factor response, the multinomial logit model (multinomial) is more appropriate.

Here is an example of the usage of the parallel argument. If there are covariates \( x_1, x_2 \) and \( x_3 \), then \( \text{parallel} = \text{TRUE} \sim x_1 + x_2 \sim 1 \) and \( \text{parallel} = \text{FALSE} \sim x_3 \) are equivalent. This would constrain the regression coefficients for \( x_1 \) and \( x_2 \) to be equal; those of the intercepts and \( x_3 \) would be different.

Author(s)
Thomas W. Yee

References

See Also
cumulative, cratio, sratio, multinomial, pneumo.

Examples
pneumo <- transform(pneumo, let = log(exposure.time))
(fit <- vglm(cbind(normal, mild, severe) ~ let, acat, data = pneumo))
coef(fit, matrix = TRUE)
constraints(fit)
model.matrix(fit)
**AICvlm**

*Akaike's Information Criterion*

---

**Description**

Calculates the Akaike information criterion for a fitted model object for which a log-likelihood value has been obtained.

**Usage**

```r
AICvlm(object, ..., corrected = FALSE, k = 2)
AICvgam(object, ..., k = 2)
AICrrvglm(object, ..., k = 2)
AICqrrvglm(object, ..., k = 2)
AICrrvgam(object, ..., k = 2)
```

**Arguments**

- `object`: Some **VGAM** object, for example, having class `vglmff-class`.
- `...`: Other possible arguments fed into `logLik` in order to compute the log-likelihood.
- `corrected`: Logical, perform the finite sample correction?
- `k`: Numeric, the penalty per parameter to be used; the default is the classical AIC.

**Details**

The following formula is used for VGLMs: $-2\log\text{likelihood} + kn_{par}$, where $n_{par}$ represents the number of parameters in the fitted model, and $k = 2$ for the usual AIC. One could assign $k = \log(n)$ ($n$ the number of observations) for the so-called BIC or SBC (Schwarz's Bayesian criterion). This is the function `AICvlm()`.

This code relies on the log-likelihood being defined, and computed, for the object. When comparing fitted objects, the smaller the AIC, the better the fit. The log-likelihood and hence the AIC is only defined up to an additive constant.

Any estimated scale parameter (in GLM parlance) is used as one parameter.

For VGAMs and CAO the nonlinear effective degrees of freedom for each smoothed component is used. This formula is heuristic. These are the functions `AICvgam()` and `AICcao()`.

The finite sample correction is usually recommended when the sample size is small or when the number of parameters is large. When the sample size is large their difference tends to be negligible. The correction is described in Hurvich and Tsai (1989), and is based on a (univariate) linear model with normally distributed errors.

**Value**

Returns a numeric value with the corresponding AIC (or BIC, or ..., depending on $k$).
Warning

This code has not been double-checked. The general applicability of AIC for the VGLM/VGAM classes has not been developed fully. In particular, AIC should not be run on some VGAM family functions because of violation of certain regularity conditions, etc.

Note

AIC has not been defined for QRR-VGLMs yet.

Using AIC to compare posbinomial models with, e.g., posbernoulli.tb models, requires posbinomial(omit.constant = TRUE) = TRUE. See posbinomial for an example. A warning is given if it suspects a wrong omit.constant value was used.

Where defined, AICC(...) is the same as AIC(..., corrected = TRUE).

Author(s)

T. W. Yee.

References


See Also

VGLMs are described in vglm-class; VGAMs are described in vgam-class; RR-VGLMs are described in rrvglm-class; AIC, BICvlm.

Examples

pneumo <- transform(pneumo, let = log(exposure.time))
(fit1 <- vglm(cbind(normal, mild, severe) ~ let,
            cumulative(parallel = TRUE, reverse = TRUE), data = pneumo))
coef(fit1, matrix = TRUE)
AIC(fit1)
AICC(fit1)  # Quick way
AIC(fit1, corrected = TRUE)  # Slow way
(fit2 <- vglm(cbind(normal, mild, severe) ~ let,
            cumulative(parallel = FALSE, reverse = TRUE), data = pneumo))
coef(fit2, matrix = TRUE)
AIC(fit2)
AICC(fit2)
AIC(fit2, corrected = TRUE)
alaplace

Description

Maximum likelihood estimation of the 1, 2 and 3-parameter asymmetric Laplace distributions (ALDs). The 2-parameter ALD may, with trepidation and lots of skill, sometimes be used as an approximation of quantile regression.

Usage

alaplace1(tau = NULL, llocation = "identitylink",
           location = NULL, kappa = sqrt(tau/(1 - tau)), Scale.arg = 1,
           ishrinkage = 0.95, parallel.locat = TRUE ~ 0, digt = 4,
           idf.mu = 3, zero = NULL, imethod = 1)

alaplace2(tau = NULL, llocation = "identitylink", lscale = "loge",
           location = NULL, iscale = NULL, kappa = sqrt(tau/(1 - tau)),
           ishrinkage = 0.95,
           parallel.locat = TRUE ~ 0,
           parallel.scale = FALSE ~ 0,
           digt = 4, idf.mu = 3, imethod = 1, zero = -2)

alaplace3(llocation = "identitylink", lscale = "loge", lkappa = "loge",
           location = NULL, iscale = NULL, ikappa = 1,
           imethod = 1, zero = 2:3)

Arguments

tau, kappa Numeric vectors with 0 < τ < 1 and κ > 0. Most users will only specify tau since the estimated location parameter corresponds to the τth regression quantile, which is easier to understand. See below for details.

llocation, lscale, lkappa Character. Parameter link functions for location parameter ξ, scale parameter σ, asymmetry parameter κ. See Links for more choices. For example, the argument llocation can help handle count data by restricting the quantiles to be positive (use llocation = "loge"). However, llocation is best left alone since the theory only works properly with the identity link.

ilocation, iscale, ikappa Optional initial values. If given, it must be numeric and values are recycled to the appropriate length. The default is to choose the value internally.

parallel.locat, parallel.scale See the parallel argument of CommonVGAMffArguments. These arguments apply to the location and scale parameters. It generally only makes sense for the scale parameters to be equal, hence set parallel.scale = TRUE. Note that
assigning `parallel.locat` the value `TRUE` circumvents the seriously embarrassing quantile crossing problem because all constraint matrices except for the intercept correspond to a parallelism assumption.

**imethod**
Initialization method. Either the value 1, 2, 3 or 4.

**idf.mu**
Degrees of freedom for the cubic smoothing spline fit applied to get an initial estimate of the location parameter. See `vsmooth.spline`. Used only when `imethod = 3`.

**ishrinkage**
How much shrinkage is used when initializing $\xi$. The value must be between 0 and 1 inclusive, and a value of 0 means the individual response values are used, and a value of 1 means the median or mean is used. This argument is used only when `imethod = 4`. See `CommonVGAMffArguments` for more information.

**Scale.arg**
The value of the scale parameter $\sigma$. This argument may be used to compute quantiles at different $\tau$ values from an existing fitted `alaplace()` model (practical only if it has a single value). If the model has `parallel.locat = TRUE` then only the intercept need be estimated; use an offset. See below for an example.

**digt**
Passed into `Round` as the digits argument for the `tau` values; used cosmetically for labelling.

**zero**
See `CommonVGAMffArguments` for more information. Where possible, the default is to model all the $\sigma$ and $\kappa$ as an intercept-only term.

**Details**
These **VGAM** family functions implement one variant of asymmetric Laplace distributions (ALDs) suitable for quantile regression. Kotz et al. (2001) call it the ALD. Its density function is

$$f(y; \xi, \sigma, \kappa) = \frac{\sqrt{2}}{\sigma} \frac{\kappa}{1 + \kappa^2} \exp\left(-\frac{\sqrt{2}}{\sigma \kappa} |y - \xi|\right)$$

for $y \leq \xi$, and

$$f(y; \xi, \sigma, \kappa) = \frac{\sqrt{2}}{\sigma} \frac{\kappa}{1 + \kappa^2} \exp\left(-\frac{\sqrt{2} \kappa}{\sigma} |y - \xi|\right)$$

for $y > \xi$. Here, the ranges are for all real $y$ and $\xi$, positive $\sigma$ and positive $\kappa$. The special case $\kappa = 1$ corresponds to the (symmetric) Laplace distribution of Kotz et al. (2001). The mean is $\xi + \sigma(1/\kappa - \kappa)/\sqrt{2}$ and the variance is $\sigma^2(1 + \kappa^4)/(2\kappa^2)$. The enumeration of the linear/additive predictors used for `alaplace2()` is the first location parameter followed by the first scale parameter, then the second location parameter followed by the second scale parameter, etc. For `alaplace3()`, only a vector response is handled and the last (third) linear/additive predictor is for the asymmetry parameter.

It is known that the maximum likelihood estimate of the location parameter $\xi$ corresponds to the regression quantile estimate of the classical quantile regression approach of Koenker and Bassett (1978). An important property of the ALD is that $P(Y \leq \xi) = \tau$ where $\tau = \kappa^2/(1 + \kappa^2)$ so that $\kappa = \sqrt{\tau/(1 - \tau)}$. Thus `alaplace2()` might be used as an alternative to `rq` in the **quantreg** package.

Both `alaplace1()` and `alaplace2()` can handle multiple responses, and the number of linear/additive predictors is dictated by the length of `tau` or `kappa`. The functions `alaplace1()` and `alaplace2()`
can also handle multiple responses (i.e., a matrix response) but only with a single-valued \( \tau \) or kappa.

**Value**

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`. In the extra slot of the fitted object are some list components which are useful, e.g., the sample proportion of values which are less than the fitted quantile curves.

**Warning**

These functions are experimental and especially subject to change or withdrawal. The MLE regularity conditions do not hold for this distribution so that misleading inferences may result, e.g., in the `summary` and `vcov` of the object.

Care is needed with \( \tau \) values which are too small, e.g., for count data with `location = "loge"` and if the sample proportion of zeros is greater than \( \tau \).

**Note**

These VGAM family functions use Fisher scoring. Convergence may be slow and half-stepping is usual (although one can use `trace = TRUE` to see which is the best model and then use `maxit` to choose that model) due to the regularity conditions not holding. Often the iterations slowly crawl towards the solution so monitoring the convergence (set `trace = TRUE`) is highly recommended.

For large data sets it is a very good idea to keep the length of \( \tau/\kappa \) low to avoid large memory requirements. Then for `parallel.locat = FALSE` one can repeatedly fit a model with `alaplace1()` with one \( \tau \) at a time; and for `parallel.locat = TRUE` one can refit a model with `alaplace1()` with one \( \tau \) at a time but using offsets and an intercept-only model.

A second method for solving the noncrossing quantile problem is illustrated below in Example 3. This is called the *accumulative quantile method* (AQM) and details are in Yee (2014). It does not make the strong parallelism assumption.

The functions `alaplace2()` and `laplace` differ slightly in terms of the parameterizations.

**Author(s)**

Thomas W. Yee

**References**


**See Also**

`ralap`, `laplace`, `CommonVGAMffArguments`, `lms.bcn`, `amlnormal`, `sc.studentt2`, `simulate.vlm`. 
Examples

```r
## Not run:
# Example 1: quantile regression with smoothing splines
set.seed(123); adata <- data.frame(x2 = sort(runif(n <- 500)))
mymu <- function(x) exp(-2 + 6*sin(2*x-.02) / (x+.05)^2)
adata <- transform(adata, y = rpois(n, lambda = mymu(x2)))
mytau <- c(0.25, 0.75); mydof <- 4
fit <- vgam(y ~ s(x2, df = mydof), data = adata, trace = TRUE, maxit = 900,
            alaplace2(tau = mytau, llocat = "loge",
                       parallel.locat = FALSE))
fitp <- vgam(y ~ s(x2, df = mydof), data = adata, trace = TRUE, maxit = 900,
            alaplace2(tau = mytau, llocat = "loge", parallel.locat = TRUE))

par(las = 1); mylwd <- 1.5
with(adata, plot(x2, jitter(y, factor = 0.5), col = "orange",
                 main = "Example 1; green: parallel.locat = TRUE",
                 ylab = "y", pch = "o", cex = 0.75))
with(adata, matlines(x2, fitted(fit ), col = "blue",
                     lty = "solid", lwd = mylwd))
with(adata, matlines(x2, fitted(fitp), col = "green",
                     lty = "solid", lwd = mylwd))

finexgrid <- seq(0, 1, len = 100)
for (ii in 1:length(mytau))
  lines(finexgrid, qpois(p = mytau[ii], lambda = mymu(finexgrid)),
        col = "blue", lwd = mylwd)
fit@extra # Contains useful information

## Example 2: regression quantile at a new tau value from an existing fit
# Nb. regression splines are used here since it is easier.
fitp2 <- vglm(y ~ sm.bs(x2, df = mydof), data = adata, trace = TRUE,
             alaplace1(tau = mytau, llocation = "loge",
                       parallel.locat = TRUE))

newtau <- 0.5 # Want to refit the model with this tau value
fitp3 <- vglm(y ~ 1 + offset(predict(fitp2)[, 1]),
              alaplace1(tau = newtau, llocation = "loge"), data = adata)
with(adata, plot(x2, jitter(y, factor = 0.5), col = "orange",
                 pch = "o", cex = 0.75, ylab = "y",
                 main = "Example 2; parallel.locat = TRUE"))
with(adata, matlines(x2, fitted(fitp2), col = "blue",
                     lty = 1, lwd = mylwd))
with(adata, matlines(x2, fitted(fitp3), col = "black",
                     lty = 1, lwd = mylwd))

## Example 3: noncrossing regression quantiles using a trick: obtain
# successive solutions which are added to previous solutions; use a log
# link to ensure an increasing quantiles at any value of x.

mytau <- seq(0.2, 0.9, by = 0.1)
```
answer <- matrix(0, nrow(adata), length(mytau))  # Stores the quantiles
adata <- transform(adata, offsety = y*0)
usetau <- mytau
for (ii in 1:length(mytau)) {
  # cat("\n\ni = ", ii, "\n")
  adloc <- transform(adata, usey = y-offsety)
  iloc <- ifelse(ii == 1, with(adata, median(y)), 1.0)  # Well-chosen!
  mydf <- ifelse(ii == 1, 5, 3)  # Maybe less smoothing will help
  # lloc <- ifelse(ii == 1, "loge", "loge")  # 2nd value must be "loge"
  fit3 <- vglm(usey ~ sm.ns(x2, df = mydf), data = adata, trace = TRUE,
               laplace2(tau = usetau[ii], lloc = "loge", iloc = iloc))
  answer[, ii] <- (if(ii == 1) 0 else answer[, ii-1]) + fitted(fit3)
  adata <- transform(adata, offsety = answer[, ii])
}

# Plot the results.
with(adata, plot(x2, y, col = "blue",
               main = paste("Noncrossing and nonparallel; tau = ",
                            paste(mytau, collapse = ", ", "\n"))))
with(adata, matlines(x2, answer, col = "orange", lty = 1))

# Zoom in near the origin.
with(adata, plot(x2, y, col = "blue", xlim = c(0, 0.2), ylim = 0:1,
               main = paste("Noncrossing and nonparallel; tau = ",
                            paste(mytau, collapse = ", ", "\n"))))
with(adata, matlines(x2, answer, col = "orange", lty = 1))

## End(Not run)

---

## The Laplace Distribution

### Description

Density, distribution function, quantile function and random generation for the 3-parameter asymmetric Laplace distribution with location parameter location, scale parameter scale, and asymmetry parameter kappa.

### Usage

```r
dalap(x, location = 0, scale = 1, tau = 0.5, kappa = sqrt(tau/(1-tau)), log = FALSE)
palap(q, location = 0, scale = 1, tau = 0.5, kappa = sqrt(tau/(1-tau)),
      lower.tail = TRUE, log.p = FALSE)
qalap(p, location = 0, scale = 1, tau = 0.5, kappa = sqrt(tau/(1-tau)),
      lower.tail = TRUE, log.p = FALSE)
ralap(n, location = 0, scale = 1, tau = 0.5, kappa = sqrt(tau/(1-tau)))
```
Arguments

- **x**, **q**: vector of quantiles.
- **p**: vector of probabilities.
- **n**: number of observations. If `length(n) > 1` then the length is taken to be the number required.
- **location**: the location parameter $\xi$.
- **scale**: the scale parameter $\sigma$. Must consist of positive values.
- **tau**: the quantile parameter $\tau$. Must consist of values in $(0, 1)$. This argument is used to specify kappa and is ignored if kappa is assigned.
- **kappa**: the asymmetry parameter $\kappa$. Must consist of positive values.
- **log**: if TRUE, probabilities $p$ are given as $\log(p)$.
- **lower.tail, log.p**: Same meaning as in `pnorm` or `qnorm`.

Details

There are many variants of asymmetric Laplace distributions (ALDs) and this one is known as the ALD by Kotz et al. (2001). See `alaplace3`, the VGAM family function for estimating the three parameters by maximum likelihood estimation, for formulae and details.

Value

dalap gives the density, palap gives the distribution function, qalap gives the quantile function, and ralap generates random deviates.

Author(s)

T. W. Yee and Kai Huang

References


See Also

`alaplace3`.

Examples

```r
x <- seq(-5, 5, by = 0.01)
loc <- 0; sigma <- 1.5; kappa <- 2
## Not run: plot(x, dalap(x, loc, sigma, kappa = kappa), type = "l", col = "blue",
       main = "Blue is density, orange is cumulative distribution function",
       ylim = c(0, 1), sub = "Purple are 5, 10, ..., 95 percentiles",
       las = 1, ylab = "", cex.main = 0.5)
```
amlbinomial

```
abline(h = 0, col = "blue", lty = 2)
lines(qalap(seq(0.05, 0.95, by = 0.05), loc, sigma, kappa = kappa),
      dalap(qalap(seq(0.05, 0.95, by = 0.05), loc, sigma, kappa = kappa),
           loc, sigma, kappa = kappa), col = "purple", lty = 3, type = "h")
lines(x, palap(x, loc, sigma, kappa = kappa), type = "l", col = "orange")
abline(h = 0, lty = 2)
## End(Not run)

pp <- seq(0.05, 0.95, by = 0.05) # Test two functions
max(abs(palap(qalap(pp, loc, sigma, kappa = kappa),
              loc, sigma, kappa = kappa) - pp)) # Should be 0
```

---

**amlbinomial**

*Binomial Logistic Regression by Asymmetric Maximum Likelihood Estimation*

### Description

Binomial quantile regression estimated by maximizing an asymmetric likelihood function.

### Usage

```r
amlbinomial(w.aml = 1, parallel = FALSE, digw = 4, link = "logit")
```

### Arguments

- **w.aml**: Numeric, a vector of positive constants controlling the percentiles. The larger the value the larger the fitted percentile value (the proportion of points below the “w-regression plane”). The default value of unity results in the ordinary maximum likelihood (MLE) solution.

- **parallel**: If `w.aml` has more than one value then this argument allows the quantile curves to differ by the same amount as a function of the covariates. Setting this to be TRUE should force the quantile curves to not cross (although they may not cross anyway). See `CommonVGAMffArguments` for more information.

- **digw**: Passed into `Round` as the digits argument for the `w.aml` values; used cosmetically for labelling.

- **link**: See `binomialff`.

### Details

The general methodology behind this **VGAM** family function is given in Efron (1992) and full details can be obtained there. This model is essentially a logistic regression model (see `binomialff`) but the usual deviance is replaced by an asymmetric squared error loss function; it is multiplied by `w.aml` for positive residuals. The solution is the set of regression coefficients that minimize the sum of these deviance-type values over the data set, weighted by the `weights` argument (so that it can contain frequencies). Newton-Raphson estimation is used here.
Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

If w.aml has more than one value then the value returned by deviance is the sum of all the (weighted) deviances taken over all the w.aml values. See Equation (1.6) of Efron (1992).

Note

On fitting, the extra slot has list components "w.aml" and "percentile". The latter is the percent of observations below the “w-regression plane”, which is the fitted values. Also, the individual deviance values corresponding to each element of the argument w.aml is stored in the extra slot.

For amlbinomial objects, methods functions for the generic functions qtplot and cdf have not been written yet.

See amlpoisson about comments on the jargon, e.g., expectiles etc.

In this documentation the word quantile can often be interchangeably replaced by expectile (things are informal here).

Author(s)

Thomas W. Yee

References


See Also

amlpoisson, amlexponential, amlnormal, alaplaceQ, denorm.

Examples

```r
# Example: binomial data with lots of trials per observation
set.seed(1234)
sizevec <- rep(100, length = (nn <- 200))
mydat <- data.frame(x = sort(runif(nn)))
mydat <- transform(mydat, prob = logit(-0 + 2.5*x + x^2, inverse = TRUE))
mydat <- transform(mydat, y = rbinom(nn, size = sizevec, prob = prob))
(fit <- vgam(cbind(y, sizevec - y) ~ s(x, df = 3),
            amlbinomial(w = c(0.01, 0.2, 1, 5, 60)),
            mydat, trace = TRUE))
fit@extra

## Not run:
par(mfrow = c(1,2))
# Quantile plot
with(mydat, plot(x, jitter(y), col = "blue", las = 1, main =
```
amlexponential

Exponential Regression by Asymmetric Maximum Likelihood Estimation

Description

Exponential expectile regression estimated by maximizing an asymmetric likelihood function.

Usage

amlexponential(w.aml = 1, parallel = FALSE, imethod = 1, digw = 4,
link = "loge")

Arguments

w.aml
   Numeric, a vector of positive constants controlling the expectiles. The larger the
   value the larger the fitted expectile value (the proportion of points below the "w-
   regression plane"). The default value of unity results in the ordinary maximum
   likelihood (MLE) solution.

parallel
   If w.aml has more than one value then this argument allows the quantile curves
differ to be a different function of the covariates. Setting this to be
TRUE should force the quantile curves to not cross (although they may not cross
anyway). See CommonVGAMffArguments for more information.

imethod
   Integer, either 1 or 2 or 3. Initialization method. Choose another value if con-
   vergence fails.

digw
   Passed into Round as the digits argument for the w.aml values; used cosmeti-
   cally for labelling.

link
   See exponential and the warning below.
Details

The general methodology behind this VGAM family function is given in Efron (1992) and full details can be obtained there.

This model is essentially an exponential regression model (see exponential) but the usual deviance is replaced by an asymmetric squared error loss function; it is multiplied by $w.aml$ for positive residuals. The solution is the set of regression coefficients that minimize the sum of these deviance-type values over the data set, weighted by the weights argument (so that it can contain frequencies). Newton-Raphson estimation is used here.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

Note that the link argument of exponential and amlexponential are currently different: one is the rate parameter and the other is the mean (expectile) parameter.

If $w.aml$ has more than one value then the value returned by deviance is the sum of all the (weighted) deviances taken over all the $w.aml$ values. See Equation (1.6) of Efron (1992).

Note

On fitting, the extra slot has list components "w.aml" and "percentile". The latter is the percent of observations below the "w-regression plane", which is the fitted values. Also, the individual deviance values corresponding to each element of the argument $w.aml$ is stored in the extra slot.

For amlexponential objects, methods functions for the generic functions qtplot and cdf have not been written yet.

See amlpoisson about comments on the jargon, e.g., expectiles etc.

In this documentation the word quantile can often be interchangeably replaced by expectile (things are informal here).

Author(s)

Thomas W. Yee

References


See Also

exponential, amlbinomial, amlpoisson, amlnormal, alaplace1, lms.bc, deexp.
Examples

```r
nn <- 2000
mydat <- data.frame(x = seq(0, 1, length = nn))
mydat <- transform(mydat, mu = loge(-0 + 1.5*x + 0.2*x^2, inverse = TRUE))
mydat <- transform(mydat, mu = loge(0 - sin(8*x), inverse = TRUE))
mydat <- transform(mydat, y = rexp(nn, rate = 1/mu))
(fit <- vgam(y ~ s(x,df = 5), amlexponential(w = c(0.001, 0.1, 0.5, 5, 60)),
              mydat, trace = TRUE))
fit@extra

## Not run: # These plots are against the sqrt scale (to increase clarity)
par(mfrow = c(1,2))
# Quantile plot
with(mydat, plot(x, sqrt(y), col = "blue", las = 1, main =
                 paste(paste(round(fit@extra$percentile, digits = 1), collapse = ", "),
                 "percentile-expectile curves")))
with(mydat, matlines(x, sqrt(fitted(fit)), lwd = 2, col = "blue", lty = 1))

# Compare the fitted expectiles with the quantiles
with(mydat, plot(x, sqrt(y), col = "blue", las = 1, main =
                 paste(paste(round(fit@extra$percentile, digits = 1), collapse = ", "),
                 "percentile curves are orange")))
with(mydat, matlines(x, sqrt(fitted(fit)), lwd = 2, col = "blue", lty = 1))

for (ii in fit@extra$percentile)
  with(mydat, matlines(x, sqrt(qexp(p = ii/100, rate = 1/mu)), col = "orange"))
## End(Not run)
```

Description

Asymmetric least squares, a special case of maximizing an asymmetric likelihood function of a normal distribution. This allows for expectile/quantile regression using asymmetric least squares error loss.

Usage

```r
amlnormal(w.aml = 1, parallel = FALSE, lexpectile = "identitylink",
          iexpectile = NULL, imethod = 1, digw = 4)
```

Arguments

- **w.aml**: Numeric, a vector of positive constants controlling the percentiles. The larger the value the larger the fitted percentile value (the proportion of points below the “w-regression plane”). The default value of unity results in the ordinary least squares (OLS) solution.
parallel

If \( w.\text{aml} \) has more than one value then this argument allows the quantile curves to differ by the same amount as a function of the covariates. Setting this to be \( \text{TRUE} \) should force the quantile curves to not cross (although they may not cross anyway). See \texttt{CommonVGAMffArguments} for more information.

\texttt{ilexpectile, iexpectile}

See \texttt{CommonVGAMffArguments} for more information.

\texttt{imethod}

Integer, either 1 or 2 or 3. Initialization method. Choose another value if convergence fails.

\texttt{digw}

Passed into \texttt{Round} as the \texttt{digits} argument for the \( w.\text{aml} \) values; used cosmetically for labelling.

Details

This is an implementation of Efron (1991) and full details can be obtained there. Equation numbers below refer to that article. The model is essentially a linear model (see \texttt{lm}), however, the asymmetric squared error loss function for a residual \( r \) is \( r^2 \) if \( r \leq 0 \) and \( wr^2 \) if \( r > 0 \). The solution is the set of regression coefficients that minimize the sum of these over the data set, weighted by the \texttt{weights} argument (so that it can contain frequencies). Newton-Raphson estimation is used here.

Value

An object of class "\texttt{vglmff}" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm} and \texttt{vgam}.

Note

On fitting, the extra slot has list components "\texttt{w.aml}" and "\texttt{percentile}". The latter is the percent of observations below the “\texttt{w-regression plane}”, which is the fitted values.

One difficulty is finding the \( w.\text{aml} \) value giving a specified percentile. One solution is to fit the model within a root finding function such as \texttt{uniroot}; see the example below.

For \texttt{amlnormal} objects, methods functions for the generic functions \texttt{qtplot} and \texttt{cdf} have not been written yet.

See the note in \texttt{amlpoisson} on the jargon, including \textit{expectiles} and \textit{regression quantiles}.

The deviance slot computes the total asymmetric squared error loss (2.5). If \( w.\text{aml} \) has more than one value then the value returned by the slot is the sum taken over all the \( w.\text{aml} \) values.

This \texttt{VGAM} family function could well be renamed \texttt{amlnormal()} instead, given the other function names \texttt{amlpoisson}, \texttt{amlbinomial}, etc.

In this documentation the word \textit{quantile} can often be interchangeably replaced by \textit{expectile} (things are informal here).

Author(s)

Thomas W. Yee

References

See Also

amlpoisson, amlbimomial, amlexponential, bmi.nz, alaplace1, denorm, lms.bcn and similar variants are alternative methods for quantile regression.

Examples

```r
## Not run
# Example 1
ooo <- with(bmi.nz, order(age))
bmi.nz <- bmi.nz[ooo, ] # Sort by age
(fit <- vglm(BMI ~ sm.bs(age), amlnormal(w.aml = 0.1), data = bmi.nz))
fit@extra # Gives the w value and the percentile
coef(fit, matrix = TRUE)

# Quantile plot
with(bmi.nz, plot(age, BMI, col = "blue", main = 
  paste(round(fit@extra$percentile, digits = 1), 
  "expectile-percentile curve")))
with(bmi.nz, lines(age, c(fitted(fit)), col = "black"))

# Example 2
# Find the w values that give the 25, 50 and 75 percentiles
find.w <- function(w, percentile = 50) {
  fit2 <- vglm(BMI ~ sm.bs(age), amlnormal(w = w), data = bmi.nz)
  fit2@extra$percentile - percentile
}

# Quantile plot
with(bmi.nz, plot(age, BMI, col = "blue", las = 1, main = 
  "25, 50 and 75 expectile-percentile curves"))
for (myp in c(25, 50, 75)) {
  # Note: uniroot() can only find one root at a time
  bestw <- uniroot(f = find.w, interval = c(1/10^4, 10^4), percentile = myp)
  fit2 <- vglm(BMI ~ sm.bs(age), amlnormal(w = bestw$root), data = bmi.nz)
  with(bmi.nz, lines(age, c(fitted(fit2)), col = "orange"))
}

# Example 3; this is Example 1 but with smoothing splines and
# a vector w and a parallelism assumption.
ooo <- with(bmi.nz, order(age))
bmi.nz <- bmi.nz[ooo, ] # Sort by age
fit3 <- vgam(BMI ~ s(age, df = 4), data = bmi.nz, trace = TRUE,
  amlnormal(w = c(0.1, 1, 10), parallel = TRUE))
fit3@extra # The w values, percentiles and weighted deviances

# The linear components of the fit; not for human consumption:
coef(fit3, matrix = TRUE)

# Quantile plot
with(bmi.nz, plot(age, BMI, col="blue", main = 
  paste(paste(round(fit3@extra$percentile, digits = 1), collapse = " ", 
  "expectile-percentile curves")))
with(bmi.nz, matlines(age, fitted(fit3), col = 1:fit3@extra$M, lwd = 2))
```
Poisson Regression by Asymmetric Maximum Likelihood Estimation

Description
Poisson quantile regression estimated by maximizing an asymmetric likelihood function.

Usage
amlpoisson(w.aml = 1, parallel = FALSE, imethod = 1, digw = 4, link = "log")

Arguments

  w.aml  Numeric, a vector of positive constants controlling the percentiles. The larger the value the larger the fitted percentile value (the proportion of points below the “w-regression plane”). The default value of unity results in the ordinary maximum likelihood (MLE) solution.

  parallel  If w.aml has more than one value then this argument allows the quantile curves to differ by the same amount as a function of the covariates. Setting this to be TRUE should force the quantile curves to not cross (although they may not cross anyway). See CommonVGAMffArguments for more information.

  imethod  Integer, either 1 or 2 or 3. Initialization method. Choose another value if convergence fails.

  digw  Passed into Round as the digits argument for the w.aml values; used cosmetically for labelling.

  link  See poissonff.

Details
This method was proposed by Efron (1992) and full details can be obtained there.

The model is essentially a Poisson regression model (see poissonff) but the usual deviance is replaced by an asymmetric squared error loss function; it is multiplied by w.aml for positive residuals. The solution is the set of regression coefficients that minimize the sum of these deviance-type values over the data set, weighted by the weights argument (so that it can contain frequencies). Newton-Raphson estimation is used here.

Value
An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.
**Warning**

If \( \text{w.a.ml} \) has more than one value then the value returned by deviance is the sum of all the (weighted) deviances taken over all the \( \text{w.a.ml} \) values. See Equation (1.6) of Efron (1992).

**Note**

On fitting, the extra slot has list components "\( \text{w.a.ml} \)" and "percentile". The latter is the percent of observations below the "\( \text{w} \)-regression plane", which is the fitted values. Also, the individual deviance values corresponding to each element of the argument \( \text{w.a.ml} \) is stored in the extra slot.

For \( \text{amlpoisson} \) objects, methods functions for the generic functions \( \text{qtplot} \) and \( \text{cdf} \) have not been written yet.

About the jargon, Newey and Powell (1987) used the name *expectiles* for regression surfaces obtained by asymmetric least squares. This was deliberate so as to distinguish them from the original *regression quantiles* of Koenker and Bassett (1978). Efron (1991) and Efron (1992) use the general name *regression percentile* to apply to all forms of asymmetric fitting. Although the asymmetric maximum likelihood method very nearly gives regression percentiles in the strictest sense for the normal and Poisson cases, the phrase *quantile regression* is used loosely in this \text{VGAM} documentation.

In this documentation the word *quantile* can often be interchangeably replaced by *expectile* (things are informal here).

**Author(s)**

Thomas W. Yee

**References**


**See Also**

\( \text{amlnormal, amlbinomial, alaplace} \).

**Examples**

```r
set.seed(1234)
mydat <- data.frame(x = sort(runif(nn <- 200)))
mydat <- transform(mydat, y = rpois(nn, exp(0 + sin(8*x))))
(fit <- vgam(y ~ s(x), fam = amlpoisson(w.a.ml = c(0.02, 0.2, 1, 5, 50)),
            mydat, trace = TRUE))
fit@extra
```
AR1

*Autoregressive Process with Order-1 Family Function*

**Description**

Maximum likelihood estimation of the three-parameter AR-1 model

**Usage**

```r
AR1(ldrift = "identitylink", lsd = "loge", lvar = "loge",
lrho = "rhobit", idrift = NULL,
isd = NULL, ivar = NULL, irho = NULL,
ishrinkage = 0.9, type.likelihood = c("exact", "conditional"),
var.arg = FALSE, almost = 0.99, zero = c(-2, -3))
```

**Arguments**

- `ldrift, lsd, lvar, lrho`
  - Link functions applied to the scaled mean, standard deviation or variance, and correlation parameters. The parameter drift is known as the *drift*, and it is a scaled mean. See **Links** for more choices.

- `idrift, isd, ivar, irho`
  - Optional initial values for the parameters. If failure to converge occurs then try different values and monitor convergence by using `trace = TRUE`. For a $S$-column response, these arguments can be of length $S$, and they are recycled by the columns first. A value `NULL` means an initial value for each response is computed internally.

- `ishrinkage, zero`
  - See **CommonVGAMffArguments** for more information.

- `var.arg`
  - Same meaning as `uninormal`.

- `type.likelihood`
  - What type of likelihood function is maximized. The first choice (default) is the sum of the marginal likelihood and the conditional likelihood. Choosing the conditional likelihood means that the first observation is effectively ignored (this is handled internally by setting the value of the first prior weight to be some small positive number, e.g., `1.0e-6`). See the note below.

- `almost`
  - A value close to 1 but slightly smaller. One of the off-diagonal EIM elements is multiplied by this, to ensure that the EIM is positive-definite.
Details

The AR-1 model implemented here has

\[ Y_1 \sim N(\mu, \sigma^2/(1 - \rho^2)), \]

and

\[ Y_i = \mu^* + \rho Y_{i-1} + e_i, \]

where the \( e_i \) are i.i.d. Normal(0, sd = \( \sigma \)) random variates.

Here are a few notes: 1. A test for stationarity might be to test whether \( \mu^* \) is intercept-only. 2. The mean of all the \( Y_i \) is \( \mu^*/(1 - \rho) \) and these are returned as the fitted values. 3. The correlation of all the \( Y_i \) with \( Y_{i-1} \) is \( \rho \). 4. The default link function ensures that \(-1 < \rho < 1\).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

Monitoring convergence is urged: set trace = TRUE.

Yet to do: add an argument that allows the scaled mean parameter to be deleted, i.e, a 2-parameter model is fitted. Yet to do: ARff (p. lag = 1) should hopefully be written soon.

Note

For type.likelihood = "conditional", the prior weight for the first observation is set to some small positive number, which has the effect of deleting that observation. However, \( n \) is still the original \( n \) so that statistics such as the residual degrees of freedom are unchanged (uncorrected possibly).

Multiple responses are handled. The mean is returned as the fitted values.

Practical experience has shown that half-stepping is a very good idea. The default options use step sizes that are about one third the usual step size. Consequently, maxit is increased to about 100, by default.

Author(s)

Thomas W. Yee and Victor Miranda

See Also

vglm.control, dAR1, uninormal, arima.sim.
Examples

```r
# Example 1: using arimia.sim() to generate a stationary time series
nn <- 100; set.seed(1)
tsdata <- data.frame(x2 = runif(nn))
tsdata <- transform(tsdata,
  index = 1:nn,
  TS1 = arimia.sim(nn, model = list(ar = -0.80),
    sd = exp(1.0)),
  TS2 = arimia.sim(nn, model = list(ar = 0.50),
    sd = exp(1.0 + 2 * x2)))
fit1a <- vglm(cbind(TS1, TS2) ~ x2, AR1(zero = c(1:4, 6)),
  data = tsdata, trace = TRUE)
rhobit(-0.8)
rhobit(0.5)
coef(fit1a, matrix = TRUE)
summary(fit1a)  # SEs are useful to know

# Example 2: another stationary time series
nn <- 1000
my.rho <- rhobit(-1.0, inverse = TRUE)
my.mu <- 2.5
my.sd <- exp(1)
tsdata <- data.frame(index = 1:nn, TS3 = runif(nn))
for (i in 2:nn)
  tsdata$TS3[i] <- my.mu + my.rho * tsdata$TS3[i-1] + rnorm(1, sd = my.sd)
tsdata <- tsdata[-(1:ceiling(nn/5))],  # Remove the burn-in data:
fit2a <- vglm(TS3 ~ 1, AR1(type.likelihood = "conditional"),
  data = tsdata, trace = TRUE)
coef(fit2a, matrix = TRUE)
summary(fit2a)  # SEs are useful to know
Coef(fit2a)["rho"]  # Estimate of rho for intercept-only models
my.rho
coef(fit2a)[1]  # drift
my.mu  # Should be the same
head(weights(fit2a, type = "prior"))  # First one is effectively deleted
head(weights(fit2a, type = "working"))  # Ditto
```

---

**auuc**

**Auckland University Undergraduate Counts Data**

**Description**

Undergraduate student enrolments at the University of Auckland in 1990.

**Usage**

data(auuc)
Format

A data frame with 4 observations on the following 5 variables.

- **Commerce** a numeric vector of counts.
- **Arts** a numeric vector of counts.
- **SciEng** a numeric vector of counts.
- **Law** a numeric vector of counts.
- **Medicine** a numeric vector of counts.

Details

Each student is cross-classified by their colleges (Science and Engineering have been combined) and the socio-economic status (SES) of their fathers (1 = highest, down to 4 = lowest).

Source

Dr Tony Morrison.

References


Examples

```r
auuc
## Not run:
round(fitted(grc(auuc)))
round(fitted(grc(auuc, Rank = 2)))

## End(Not run)
```

---

**aux.posbernoulli.t**

*Auxiliary Function for the Positive Bernoulli Family Function with Time Effects*

Description

Returns behavioural effects indicator variables from a capture history matrix.

Usage

```r
aux.posbernoulli.t(y, check.y = FALSE, rename = TRUE, name = "bei")
```
Arguments

y Capture history matrix. Rows are animals, columns are sampling occasions, and values should be 0s and 1s only.

check.y Logical, if TRUE then some basic checking is performed.

rename, name If rename = TRUE then the behavioural effects indicator are named using the value of name as the prefix. If FALSE then use the same column names as y.

Details

This function can help fit certain capture–recapture models (commonly known as $M_{tb}$ or $M_{tbh}$ (no prefix h means it is an intercept-only model) in the literature). See posbernoulli.t for details.

Value

A list with the following components.

cap.hist1 A matrix the same dimension as y. In any particular row there are 0s up to the first capture. Then there are 1s thereafter.

cap1 A vector specifying which time occasion the animal was first captured.

y0i Number of noncaptures before the first capture.

yr0i Number of noncaptures after the first capture.

yr1i Number of recaptures after the first capture.

See Also

posbernoulli.t, deermice.

Examples

# Fit a M_tbh model to the deermice data:
(pdata <- aux.posbernoulli.t(with(deermice, cbind(y1, y2, y3, y4, y5, y6))))

deer - data.frame(deermice,
  be - 0,  # Add this
  pdata$cap hist1)  # Incorporate these
head(deermice)  # Augmented with behavioural effect indicator variables
tail(deermice)
Description

Data from a study of patients suffering from back pain. Prognostic variables were recorded at presentation and progress was categorised three weeks after treatment.

Usage

data(backPain)

Format

A data frame with 101 observations on the following 4 variables.

- x1 length of previous attack.
- x2 pain change.
- x3 lordosis.

- pain an ordered factor describing the progress of each patient with levels worse < same < slight.improvement < moderate.improvement < marked.improvement < complete.relief.

Source

http://ideas.repec.org/c/boc/bocode/s419001.html

The data set and this help file was copied from gnm so that a vignette in VGAM could be run; the analysis is described in Yee (2010).

References


Examples

summary(backPain)
Description

Purchasing of bacon and eggs.

Usage

data(beggs)

Format

Data frame of a two way table.

b0, b1, b2, b3, b4  The b refers to bacon. The number of times bacon was purchased was 0, 1, 2, 3, or 4.

e0, e1, e2, e3, e4  The e refers to eggs. The number of times eggs was purchased was 0, 1, 2, 3, or 4.

Details

The data is from Information Resources, Inc., a consumer panel based in a large US city [see Bell and Lattin (1998) for further details]. Starting in June 1991, the purchases in the bacon and fresh eggs product categories for a sample of 548 households over four consecutive store trips was tracked. Only those grocery shopping trips with a total basket value of at least five dollars was considered. For each household, the total number of bacon purchases in their four eligible shopping trips and the total number of egg purchases (usually a package of eggs) for the same trips, were counted.

Source


References


See Also

`rrvglm`, `rcim`, `grc`.

Examples

beggs
colSums(beggs)
rowSums(beggs)
Benford’s Distribution

Description

Density, distribution function, quantile function, and random generation for Benford’s distribution.

Usage

dbenf(x, ndigits = 1, log = FALSE)
pbenf(q, ndigits = 1, lower.tail = TRUE, log.p = FALSE)
qbenf(p, ndigits = 1, lower.tail = TRUE, log.p = FALSE)
rbenf(n, ndigits = 1)

Arguments

x, q Vector of quantiles. See ndigits.
p vector of probabilities.
n number of observations. A single positive integer. Else if length(n) > 1 then the length is taken to be the number required.
ndigits Number of leading digits, either 1 or 2. If 1 then the support of the distribution is \( \{1, \ldots, 9\} \), else \( \{10, \ldots, 99\} \).
log, log.p Logical. If \( \log.p = \text{TRUE} \) then all probabilities \( p \) are given as \( \log(p) \).
lower.tail Same meaning as in pnorm or qnorm.

Details

Benford’s Law (aka the significant-digit law) is the empirical observation that in many naturally occurring tables of numerical data, the leading significant (nonzero) digit is not uniformly distributed in \( \{1, 2, \ldots, 9\} \). Instead, the leading significant digit \( = D \), say) obeys the law

\[
P(D = d) = \log_{10} \left( 1 + \frac{1}{d} \right)
\]

for \( d = 1, \ldots, 9 \). This means the probability the first significant digit is 1 is approximately 0.301, etc.

Benford’s Law was apparently first discovered in 1881 by astronomer/mathematician S. Newcombe. It started by the observation that the pages of a book of logarithms were dirtiest at the beginning and progressively cleaner throughout. In 1938, a General Electric physicist called F. Benford rediscovered the law on this same observation. Over several years he collected data from different sources as different as atomic weights, baseball statistics, numerical data from Reader’s Digest, and drainage areas of rivers.

Applications of Benford’s Law has been as diverse as to the area of fraud detection in accounting and the design computers.
Benini

Value

dbenf gives the density, pbenf gives the distribution function, and qbenf gives the quantile function, and rbenf generates random deviates.

Author(s)

T. W. Yee and Kai Huang

References


Examples

dbenf(x <- c(0:10, NA, NaN, -Inf, Inf))
pbenf(x)

## Not run:
x <- 1:9
barplot(dbenf(x), col = "lightblue", las = 1, xlab = "Leading digit",
        ylab = "Probability", names.arg = as.character(x),
        main = paste("Benford's distribution", sep = ""))

hist(rbenf(n = 1000), border = "blue", prob = TRUE,
     main = "1000 random variates from Benford's distribution",
     xlab = "Leading digit", sub="Red is the true probability",
     breaks = 0:9 + 0.5, ylim = c(0, 0.35), xlim = c(0, 10.0))
lines(x, dbenf(x), col = "red", type = "h")
points(x, dbenf(x), col = "red")

## End(Not run)

Benini

The Benini Distribution

Description

Density, distribution function, quantile function and random generation for the Benini distribution with parameter shape.

Usage

dbenini(x, y0, shape, log = FALSE)
pbenini(q, y0, shape, lower.tail = TRUE, log.p = FALSE)
qbenini(p, y0, shape, lower.tail = TRUE, log.p = FALSE)
rbenini(n, y0, shape)
Arguments

- \( x, q \) vector of quantiles.
- \( p \) vector of probabilities.
- \( n \) number of observations. Same as \texttt{runif}.
- \( y_0 \) the scale parameter \( y_0 \).
- \( \text{shape} \) the shape parameter \( b \).
- \( \log \) Logical. If \( \log = \text{TRUE} \) then the logarithm of the density is returned.
- \( \text{lower.tail}, \log.p \) Same meaning as in \texttt{pnorm} or \texttt{qnorm}.

Details

See \texttt{benini}, the \texttt{VGAM} family function for estimating the parameter \( s \) by maximum likelihood estimation, for the formula of the probability density function and other details.

Value

dbenini gives the density, pbenini gives the distribution function, qbenini gives the quantile function, and rbenini generates random deviates.

Author(s)

T. W. Yee and Kai Huang

References


See Also

\texttt{benini}.

Examples

```r
## Not run:
y0 <- 1; shape <- exp(1)
xx <- seq(0.0, 4, len = 101)
plot(xx, dbenini(xx, y0 = y0, shape = shape), type = "l", col = "blue",
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Purple lines are the 10,20,....,90 percentiles", ylim = 0:1,
     las = 1, ylab = "", xlab = "x")
abline(h = 0, col = "blue", lty = 2)
lines(xx, pbenini(xx, y0 = y0, shape = shape), col = "orange")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qbenini(probs, y0 = y0, shape = shape)
lines(Q, dbenini(Q, y0 = y0, shape = shape), col = "purple", lty = 3, type = "h")
pbenini(Q, y0 = y0, shape = shape) - probs  # Should be all zero
```
Benini Distribution Family Function

Description

Estimating the 1-parameter Benini distribution by maximum likelihood estimation.

Usage

benini1(y0 = stop("argument 'y0' must be specified"), lshape = "loge", ishape = NULL, imethod = 1, zero = NULL)

Arguments

- **y0**: Positive scale parameter.
- **lshape**: Parameter link function and extra argument of the parameter $b$, which is the shape parameter. See Links for more choices. A log link is the default because $b$ is positive.
- **ishape**: Optional initial value for the shape parameter. The default is to compute the value internally.
- **imethod, zero**: Details at CommonVGAMffArguments.

Details

The Benini distribution has a probability density function that can be written

$$f(y) = 2s \exp(-s[(\log(y/y_0))^2]) \log(y/y_0)/y$$

for $0 < y_0 < y$, and shape $s > 0$. The cumulative distribution function for $Y$ is

$$F(y) = 1 - \exp(-s[(\log(y/y_0))^2]).$$

Here, Newton-Raphson and Fisher scoring coincide. The median of $Y$ is now returned as the fitted values, by default. This VGAM family function can handle a multiple responses, which is inputted as a matrix.

On fitting, the extra slot has a component called $y0$ which contains the value of the $y0$ argument.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

Yet to do: the 2-parameter Benini distribution estimates another shape parameter $a$ too. Hence, the code may change in the future.
The Beta-Binomial Distribution

Description

Density, distribution function, and random generation for the beta-binomial distribution.

Usage

\[
\begin{align*}
\text{dbetabinom}(x, \text{size}, \text{prob}, \text{rho} = 0, \text{log} = \text{FALSE}) \\
\text{pbetabinom}(q, \text{size}, \text{prob}, \text{rho}, \text{log.p} = \text{FALSE}) \\
\text{rbetabinom}(n, \text{size}, \text{prob}, \text{rho} = 0) \\
\text{dbetabinom.ab}(x, \text{size}, \text{shape1}, \text{shape2}, \text{log} = \text{FALSE}, \text{.dontuse.prob} = \text{NULL}) \\
\text{pbetabinom.ab}(q, \text{size}, \text{shape1}, \text{shape2}, \text{log.p} = \text{FALSE}) \\
\text{rbetabinom.ab}(n, \text{size}, \text{shape1}, \text{shape2}, \text{.dontuse.prob} = \text{NULL})
\end{align*}
\]

Arguments

- \(x, q\): vector of quantiles.
- \text{size}\: number of trials.
- \text{n}\: number of observations. Same as \text{runif}.
- \text{prob}\: the probability of success \(\mu\). Must be in the unit closed interval \([0, 1]\).
- \text{rho}\: the correlation parameter \(\rho\). Usually must be in the unit open interval \((0, 1)\), however, the value 0 is sometimes supported (if so then it corresponds to the usual binomial distribution).
shape1, shape2  the two (positive) shape parameters of the standard beta distribution. They are
called a and b in beta respectively.

log, log.p  Logical. If TRUE then all probabilities p are given as log(p).
.dontuse.prob  An argument that should be ignored and unused.

Details

The beta-binomial distribution is a binomial distribution whose probability of success is not a con-
stant but it is generated from a beta distribution with parameters shape1 and shape2. Note that the
mean of this beta distribution is \( \mu = \text{shape1}/(\text{shape1} + \text{shape2}) \), which therefore is the mean or
the probability of success.

See `betabinomial` and `betabinomialff`, the VGAM family functions for estimating the param-
eters, for the formula of the probability density function and other details.

Value

dbetabinom and dbetabinom.ab give the density, pbetabinom and pbetabinom.ab give the dis-
tribution function, and rbetabinom and rbetabinom.ab generate random deviates.

Note

pbetabinom and pbetabinom.ab can be particularly slow. The functions here ending in .ab are
called from those functions which don't. The simple transformations \( \mu = \alpha/(\alpha + \beta) \) and \( \rho =
1/(1 + \alpha + \beta) \) are used, where \( \alpha \) and \( \beta \) are the two shape parameters.

Author(s)

T. W. Yee

See Also

`betabinomial`, `betabinomialff`.

Examples

```r
set.seed(1); rbetabinom(10, 100, prob = 0.5)
set.seed(1); rbinom(10, 100, prob = 0.5)  # The same since rho = 0

## Not run: N <- 9; xx <- 0:N; s1 <- 2; s2 <- 3
dy <- dbetabinom.ab(xx, size = N, shape1 = s1, shape2 = s2)
barplot(rbind(dy, dbinom(xx, size = N, prob = s1 / (s1+s2)) ),
beside = TRUE, col = c("blue","green"), las = 1,
main = paste("Beta-binomial (size","N"," shape1"," s1,
" shape2"," s2"," ) (blue) vs\n",
" Binomial(size"," N"," prob="", s1/(s1+s2), ") (green)" , sep = ""),
names.arg = as.character(xx), cex.main = 0.8)
sum(dy * xx)  # Check expected values are equal
sum(dbinom(xx, size = N, prob = s1 / (s1+s2)) * xx)
cumsum(dy) - pbetabinom.ab(xx, N, shape1 = s1, shape2 = s2)  # Should be all 0
```
betabinomial  

Beta-binomial Distribution Family Function

Description

Fits a beta-binomial distribution by maximum likelihood estimation. The two parameters here are the mean and correlation coefficient.

Usage

betabinomial(lmu = "logit", lrho = "logit", irho = NULL, imethod = 1,  
is shrinkage = 0.95, nsimEIM = NULL, zero = 2)

Arguments

lmu, lrho  
  Link functions applied to the two parameters. See Links for more choices. The defaults ensure the parameters remain in (0, 1), however, see the warning below.

irho  
  Optional initial value for the correlation parameter. If given, it must be in (0, 1), and is recycled to the necessary length. Assign this argument a value if a convergence failure occurs. Having irho = NULL means an initial value is obtained internally, though this can give unsatisfactory results.

imethod  
  An integer with value 1 or 2 or ..., which specifies the initialization method for mu. If failure to converge occurs try the another value and/or else specify a value for irho.

zero  
  An integer specifying which linear/additive predictor is to be modelled as an intercept only. If assigned, the single value should be either 1 or 2. The default is to have a single correlation parameter. To model both parameters as functions of the covariates assign zero = NULL. See CommonVGAMffArguments for more information.

ishrinkage, nsimEIM  
  See CommonVGAMffArguments for more information. The argument is shrinkage is used only if imethod = 2. Using the argument nsimEIM may offer large advantages for large values of N and/or large data sets.
Details

There are several parameterizations of the beta-binomial distribution. This family function directly models the mean and correlation parameter, i.e., the probability of success. The model can be written $T|P = p \sim \text{Binomial}(N, p)$ where $P$ has a beta distribution with shape parameters $\alpha$ and $\beta$. Here, $N$ is the number of trials (e.g., litter size), $T = NY$ is the number of successes, and $p$ is the probability of a success (e.g., a malformation). That is, $Y$ is the proportion of successes. Like binomialff, the fitted values are the estimated probability of success (i.e., $E[Y]$ and not $E[T]$) and the prior weights $N$ are attached separately on the object in a slot.

The probability function is

$$P(T = t) = \binom{N}{t} \frac{B(\alpha + t, \beta + N - t)}{B(\alpha, \beta)}$$

where $t = 0, 1, \ldots, N$, and $B$ is the beta function with shape parameters $\alpha$ and $\beta$. Recall $Y = T/N$ is the real response being modelled.

The default model is $\eta_1 = \logit(\mu)$ and $\eta_2 = \logit(\rho)$ because both parameters lie between 0 and 1. The mean (of $Y$) is $p = \mu = \alpha/(\alpha + \beta)$ and the variance (of $Y$) is $\mu(1 - \mu)(1 + (N - 1)\rho)/N$. Here, the correlation $\rho$ is given by $1/(1 + \alpha + \beta)$ and is the correlation between the $N$ individuals within a litter. A litter effect is typically reflected by a positive value of $\rho$. It is known as the over-dispersion parameter.

This family function uses Fisher scoring. Elements of the second-order expected derivatives with respect to $\alpha$ and $\beta$ are computed numerically, which may fail for large $\alpha$, $\beta$, $N$ or else take a long time.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm.

Suppose fit is a fitted beta-binomial model. Then fit$@y$ contains the sample proportions $y$, fitted(fit) returns estimates of $E(Y)$, and weights(fit, type="prior") returns the number of trials $N$.

Warning

If the estimated rho parameter is close to zero then it pays to try lrho = "rhobit". One day this may become the default link function.

This family function is prone to numerical difficulties due to the expected information matrices not being positive-definite or ill-conditioned over some regions of the parameter space. If problems occur try setting irho to some numerical value, nsimEIM = 100, say, or else use etastart argument of vglm, etc.

Note

This function processes the input in the same way as binomialff. But it does not handle the case $N = 1$ very well because there are two parameters to estimate, not one, for each row of the input. Cases where $N = 1$ can be omitted via the subset argument of vglm.
The extended beta-binomial distribution of Prentice (1986) is currently not implemented in the VGAM package as it has range-restrictions for the correlation parameter that are currently too difficult to handle in this package. However, try \( \text{lrho} = \text{"rhobit"} \).

**Author(s)**

T. W. Yee

**References**


**See Also**

`betabinomialff`, `Betabinom`, `binomialff`, `betaff`, `dirmultinomial`, `lirat`, `simulate.vlm`.

**Examples**

```r
# Example 1
bdata <- data.frame(N = 10, mu = 0.5, rho = 0.8)
bdata <- transform(bdata, 
y = rbetabinom(n = 100, size = N, prob = mu, rho = rho))
fit <- vglm(cbind(y, N-y) ~ 1, betabinomial, data = bdata, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
head(cbind(depvar(fit), weights(fit, type = "prior")))

# Example 2
fit <- vglm(cbind(R, N-R) ~ 1, betabinomial, lirat, 
    trace = TRUE, subset = N > 1)
coef(fit, matrix = TRUE)
Coef(fit)
t(fitted(fit))
t(depvar(fit))
t(weights(fit, type = "prior"))

# Example 3, which is more complicated
lirat <- transform(lirat, fgrp = factor(grp))
summary(lirat) # Only 5 litters in group 3
fit2 <- vglm(cbind(R, N-R) ~ fgrp + hb, betabinomial(zero = 2), 
    data = lirat, trace = TRUE, subset = N > 1)
coef(fit2, matrix = TRUE)
## Not run: with(lirat, plot(hb[N > 1], fit2@misc$rho, 
    xlab = "Hemoglobin", ylab = "Estimated rho", 
    pch = as.character(grp[N > 1]), col = grp[N > 1]))
## End(Not run)
```
betabinomialff

**Description**

Fits a beta-binomial distribution by maximum likelihood estimation. The two parameters here are the shape parameters of the underlying beta distribution.

**Usage**

```r
betabinomialff(lshape1 = "loge", lshape2 = "loge", ishape1 = 1,
               ishape2 = NULL, imethod = 1, ishrinkage = 0.95,
               nsimEIM = NULL, zero = NULL)
```

**Arguments**

- `lshape1, lshape2`
  - Link functions for the two (positive) shape parameters of the beta distribution. See [Links](#) for more choices.

- `ishape1, ishape2`
  - Initial value for the shape parameters. The first must be positive, and is recycled to the necessary length. The second is optional. If a failure to converge occurs, try assigning a different value to `ishape1` and/or using `ishape2`.

- `zero`
  - An integer specifying which linear/additive predictor is to be modelled as an intercept only. If assigned, the single value should be either 1 or 2. The default is to model both shape parameters as functions of the covariates. If a failure to converge occurs, try `zero = 2`.

- `ishrinkage, nsimEIM, imethod`
  - See [CommonVGAMffArguments](#) for more information. The argument `ishrinkage` is used only if `imethod = 2`. Using the argument `nsimEIM` may offer large advantages for large values of `N` and/or large data sets.
Details

There are several parameterizations of the beta-binomial distribution. This family function directly
models the two shape parameters of the associated beta distribution rather than the probability of
success (however, see Note below). The model can be written $T|P = p \sim \text{Binomial}(N, p)$ where
$P$ has a beta distribution with shape parameters $\alpha$ and $\beta$. Here, $N$ is the number of trials (e.g.,
litter size), $T = NY$ is the number of successes, and $p$ is the probability of a success (e.g., a
malformation). That is, $Y$ is the proportion of successes. Like binomialff, the fitted values are
the estimated probability of success (i.e., $E[Y]$ and not $E[T]$) and the prior weights $N$ are attached
separately on the object in a slot.

The probability function is

$$P(T = t) = \binom{N}{t} \frac{B(\alpha + t, \beta + N - t)}{B(\alpha, \beta)}$$

where $t = 0, 1, \ldots, N$, and $B$ is the beta function with shape parameters $\alpha$ and $\beta$. Recall $Y = T/N$
is the real response being modelled.

The default model is $\eta_1 = \log(\alpha)$ and $\eta_2 = \log(\beta)$ because both parameters are positive. The mean
(of $Y$) is $p = \mu = \alpha/(\alpha + \beta)$ and the variance (of $Y$) is $\mu(1 - \mu)(1 + (N - 1)\rho)/N$. Here, the
correlation $\rho$ is given by $1/(1 + \alpha + \beta)$ and is the correlation between the $N$ individuals within a
litter. A litter effect is typically reflected by a positive value of $\rho$. It is known as the over-dispersion
parameter.

This family function uses Fisher scoring. The two diagonal elements of the second-order expected
derivatives with respect to $\alpha$ and $\beta$ are computed numerically, which may fail for large $\alpha$, $\beta$, $N$ or
else take a long time.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm.

Suppose fit is a fitted beta-binomial model. Then fit@y (better: depvar(fit)) contains the sample
proportions $y$, fitted(fit) returns estimates of $E(Y)$, and weights(fit, type = "prior")
returns the number of trials $N$.

Warning

This family function is prone to numerical difficulties due to the expected information matrices not
being positive-definite or ill-conditioned over some regions of the parameter space. If problems
occur try setting ishape1 to be some other positive value, using ishape2 and/or setting zero = 2.

This family function may be renamed in the future. See the warnings in betabinomial.

Note

This function processes the input in the same way as binomialff. But it does not handle the case
$N = 1$ very well because there are two parameters to estimate, not one, for each row of the input.
Cases where $N = 1$ can be omitted via the subset argument of vglm.

Although the two linear/additive predictors given above are in terms of $\alpha$ and $\beta$, basic algebra shows
that the default amounts to fitting a logit link to the probability of success; subtracting the second
linear/additive predictor from the first gives that logistic regression linear/additive predictor. That is, \( \text{logit}(\mu) = \eta_1 - \eta_2 \). This is illustrated in one of the examples below.

The extended beta-binomial distribution of Prentice (1986) is currently not implemented in the VGAM package as it has range-restrictions for the correlation parameter that are currently too difficult to handle in this package.

Author(s)

T. W. Yee

References


See Also

`betabinomial`, `Betabinom`, `binomialff`, `betaff`, `dirmultinomial`, `lirat`, `simulate.vlm`.

Examples

```r
# Example 1
N <- 10; s1 <- exp(1); s2 <- exp(2)
y <- rbetabinom.ab(n = 100, size = N, shape1 = s1, shape2 = s2)
fit <- vglm(cbind(y, N-y) ~ 1, betabinomialff, trace = TRUE)
coef(fit, matrix = TRUE)
Coeff(fit)
head(fit@misc$rho)  # The correlation parameter
head(cbind(depvar(fit), weights(fit, type = "prior")))

# Example 2
fit <- vglm(cbind(R, N-R) ~ 1, betabinomialff, data = lirat, 
            trace = TRUE, subset = N > 1)
coef(fit, matrix = TRUE)
Coeff(fit)
fit@misc$rho  # The correlation parameter
t(fitted(fit))
t(depvar(fit))
t(weights(fit, type = "prior"))
# A "log" link for the 2 shape parameters is a logistic regression:
all.equal(c(fitted(fit)),
         as.vector(logit(predict(fit)[, 1] - predict(fit)[, 2], inverse = TRUE)))

# Example 3, which is more complicated
lirat <- transform(lirat, fgrp = factor(grp))
```

betaff

The Two-parameter Beta Distribution Family Function

Description

Estimation of the mean and precision parameters of the beta distribution.

Usage

betaff(A = 0, B = 1, lmu = "logit", lphi = "loge",
            imu = NULL, iphi = NULL, imethod = 1, zero = NULL)

Arguments

A, B
  Lower and upper limits of the distribution. The defaults correspond to the standard beta distribution where the response lies between 0 and 1.

lmu, lphi
  Link function for the mean and precision parameters. The values A and B are extracted from the min and max arguments of extlogit. Consequently, only extlogit is allowed.

imu, iphi
  Optional initial value for the mean and precision parameters respectively. A NULL value means a value is obtained in the initialize slot.

imethod, zero
  See CommonVGAMffArguments for more information.
Details

The two-parameter beta distribution can be written
\[ f(y) = \frac{(y - A)^{\mu_1 \phi - 1} \times (B - y)^{(1 - \mu_1) \phi - 1}}{\text{beta}(\mu_1 \phi, (1 - \mu_1) \phi) \times (B - A)^{\phi - 1}} \]
for \( A < y < B \), and \( \text{beta}(\ldots) \) is the beta function (see \texttt{beta}). The parameter \( \mu_1 \) satisfies \( \mu_1 = (\mu - A)/(B - A) \) where \( \mu \) is the mean of \( Y \). That is, \( \mu_1 \) is the mean of a standard beta distribution: \( E(Y) = A + (B - A) \times \mu_1 \), and these are the fitted values of the object. Also, \( \phi \) is positive and \( A < \mu < B \). Here, the limits \( A \) and \( B \) are known.

Another parameterization of the beta distribution involving the raw shape parameters is implemented in \texttt{betaR}.

For general \( A \) and \( B \), the variance of \( Y \) is \( (B - A)^2 \times \mu_1 \times (1 - \mu_1)/(1 + \phi) \). Then \( \phi \) can be interpreted as a \emph{precision} parameter in the sense that, for fixed \( \mu \), the larger the value of \( \phi \), the smaller the variance of \( Y \). Also, \( \mu_1 = \text{shape1}/(\text{shape1} + \text{shape2}) \) and \( \phi = \text{shape1} + \text{shape2} \). Fisher scoring is implemented.

Value

An object of class "\texttt{vglmff}" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, and \texttt{vgam}.

Note

The response must have values in the interval \((A, B)\). The user currently needs to manually choose \( lmu \) to match the input of arguments \( A \) and \( B \), e.g., with \texttt{extlogit}; see the example below.

Author(s)

Thomas W. Yee

References


See Also

\texttt{betaR}, \texttt{Beta}, \texttt{genbetaII}, \texttt{betaII}, \texttt{betabinomialff}, \texttt{betageometric}, \texttt{betaprime}, \texttt{rbetageom}, \texttt{rbetanorm}, \texttt{kumar}, \texttt{extlogit}, \texttt{simulate.vlm}.

Examples

```r
bdata <- data.frame(y = rbeta(nn <- 1000, shape1 = exp(0), shape2 = exp(1)))
fit1 <- vglm(y ~ 1, betaff, data = bdata, trace = TRUE)
coef(fit1, matrix = TRUE)
Coeff(fit1)  # Useful for intercept-only models

# General A and B, and with a covariate
bdata <- transform(bdata, x2 = runif(nn))
redata <- transform(bdata, mu = logit(0.5 - x2, inverse = TRUE),
```

```r
```

The Beta-Geometric Distribution

Description
Density, distribution function, and random generation for the beta-geometric distribution.

Usage

\[
\begin{align*}
\text{dbetageom}(x, \text{shape1}, \text{shape2}, \log = \text{FALSE}) \\
\text{pbetageom}(q, \text{shape1}, \text{shape2}, \log.p = \text{FALSE}) \\
\text{rbetageom}(n, \text{shape1}, \text{shape2})
\end{align*}
\]

Arguments

- \(x, q\) vector of quantiles.
- \(n\) number of observations. Same as \text{runif}.
- \text{shape1}, \text{shape2} the two (positive) shape parameters of the standard beta distribution. They are called \(a\) and \(b\) in \text{beta} respectively.
- \log, \log.p Logical. If \text{TRUE} then all probabilities \(p\) are given as \(\log(p)\).

Details
The beta-geometric distribution is a geometric distribution whose probability of success is not a constant but it is generated from a beta distribution with parameters \(\text{shape1}\) and \(\text{shape2}\). Note that the mean of this beta distribution is \(\frac{\text{shape1}}{\text{shape1} + \text{shape2}}\), which therefore is the mean of the probability of success.

Value
\text{dbetageom} gives the density, \text{pbetageom} gives the distribution function, and \text{rbetageom} generates random deviates.

Note
\text{pbetageom} can be particularly slow.
Author(s)
T. W. Yee

See Also
geometric, betaff, Beta.

Examples
## Not run:
shape1 <- 1; shape2 <- 2; y <- 0:30
proby <- dbetageom(y, shape1, shape2, log = FALSE)
plot(y, proby, type = "h", col = "blue", ylab = "P[Y=y]", main = paste("Y ~ Beta-geometric(\text{shape1}=\", shape1,"\text{, shape2=\", shape2,"\text{)"}, sep = "")
sum(proby)

## End(Not run)

---

betageometric  Beta-geometric Distribution Family Function

Description
Maximum likelihood estimation for the beta-geometric distribution.

Usage
betageometric(lprob = "logit", lshape = "loge",
  iprob = NULL, ishape = 0.1,
  moreSummation = c(2, 100), tolerance = 1.0e-10, zero = NULL)

Arguments
lprob, lshape  Parameter link functions applied to the parameters \( p \) and \( \phi \) (called prob and shape below). The former lies in the unit interval and the latter is positive. See Links for more choices.

iprob, ishape  Numeric. Initial values for the two parameters. A NULL means a value is computed internally.

moreSummation  Integer, of length 2. When computing the expected information matrix a series summation from 0 to moreSummation[1]\*max(y)+moreSummation[2] is made, in which the upper limit is an approximation to infinity. Here, \( y \) is the response.

tolerance  Positive numeric. When all terms are less than this then the series is deemed to have converged.

zero  An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. If used, the value must be from the set \{1,2\}. 
Details

A random variable \( Y \) has a 2-parameter beta-geometric distribution if \( P(Y = y) = p(1 - p)^y \) for \( y = 0, 1, 2, \ldots \) where \( p \) are generated from a standard beta distribution with shape parameters \( \text{shape1} \) and \( \text{shape2} \). The parameterization here is to focus on the parameters \( p \) and \( \phi = 1/(\text{shape1} + \text{shape2}) \), where \( \phi \) is shape. The default link functions for these ensure that the appropriate range of the parameters is maintained. The mean of \( Y \) is \( E(Y) = \text{shape2}/(\text{shape1} - 1) = (1 - p)/(p - \phi) \) if \( \text{shape1} > 1 \), and if so, then this is returned as the fitted values.

The geometric distribution is a special case of the beta-geometric distribution with \( \phi = 0 \) (see \texttt{geometric}). However, fitting data from a geometric distribution may result in numerical problems because the estimate of \( \log(\phi) \) will 'converge' to \(-\infty\).

Value

An object of class "\texttt{vglmff}" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, and \texttt{vgam}.

Note

The first iteration may be very slow; if practical, it is best for the \texttt{weights} argument of \texttt{vglm} etc. to be used rather than inputting a very long vector as the response, i.e., \texttt{vglm(y ~ 1, \ldots, weights = wts)} is to be preferred over \texttt{vglm(rep(y, wts) ~ 1, \ldots)}. If convergence problems occur try inputting some values of argument \texttt{ishape}.

If an intercept-only model is fitted then the \texttt{misc} slot of the fitted object has list components \texttt{shape1} and \texttt{shape2}.

Author(s)

T. W. Yee

References


See Also

\texttt{geometric}, \texttt{betaff}, \texttt{rbetageom}.

Examples

\begin{verbatim}
  bdata <- data.frame(y = 0:11, wts = c(227,123,72,42,21,31,11,14,6,4,7,28))  
  fitb <- vglm(y ~ 1, betageometric, data = bdata, weight = wts, trace = TRUE)  
  fitg <- vglm(y ~ 1, geometric, data = bdata, weight = wts, trace = TRUE)  
  coef(fitb, matrix = TRUE)  
  Coef(fitb)  
  sqrt(diag(vcov(fitb, untransform = TRUE)))  
  fitb@misc$shape1  
  fitb@misc$shape2  
  # Very strong evidence of a beta-geometric:  
  pchisq(2 * (logLik(fitb) - logLik(fitg)), df = 1, lower.tail = FALSE)
\end{verbatim}
betaII

Beta Distribution of the Second Kind

Description

Maximum likelihood estimation of the 3-parameter beta II distribution.

Usage

betaII(lscale = "loge", lshape2.p = "loge", lshape3.q = "loge",
       iscale = NULL, ishape2.p = NULL, ishape3.q = NULL, imethod = 1,
       gscale = exp(-5:5), gshape2.p = exp(-5:5), gshape3.q = exp(-5:5),
       probs.y = c(0.25, 0.5, 0.75), zero = -(2:3))

Arguments

lscale, lshape2.p, lshape3.q
   Parameter link functions applied to the (positive) parameters scale, p and q.
   See Links for more choices.

iscale, ishape2.p, ishape3.q, imethod, zero
   See CommonVGAMffArguments for information.

gscale, gshape2.p, gshape3.q
   See CommonVGAMffArguments for information.

probs.y
   See CommonVGAMffArguments for information.

Details

The 3-parameter beta II is the 4-parameter generalized beta II distribution with shape parameter \( a = 1 \). It is also known as the Pearson VI distribution. Other distributions which are special cases of the 3-parameter beta II include the Lomax \((p = 1)\) and inverse Lomax \((q = 1)\). More details can be found in Kleiber and Kotz (2003).

The beta II distribution has density

\[
f(y) = y^{p-1} / \left[ b^p B(p, q) \{1 + y/b\}^{p+q} \right]
\]

for \( b > 0, p > 0, q > 0, y \geq 0 \). Here, \( b \) is the scale parameter scale, and the others are shape parameters. The mean is

\[
E(Y) = b \Gamma(p+1) \Gamma(q-1) / (\Gamma(p) \Gamma(q))
\]

provided \( q > 1 \); these are returned as the fitted values. This family function handles multiple responses.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.
Note
See the notes in genbetaII.

Author(s)
T. W. Yee

References

See Also
betaff, genbetaII, dagum, sinmad, fisk, inv.lomax, lomax, paralogistic, inv.paralogistic.

Examples
bdata <- data.frame(y = rsinmad(2000, shape1.a = 1, shape3.q = exp(2),
                     scale = exp(1)))  # Not genuine data!
fit <- vglm(y ~ 1, betaII, data = bdata, trace = TRUE)
fit <- vglm(y ~ 1, betaII(ishape2.p = 0.7, ishape3.q = 0.7),
            data = bdata, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)

Betanorm
The Beta-Normal Distribution

Description
Density, distribution function, quantile function and random generation for the univariate beta-normal distribution.

Usage
dbetanorm(x, shape1, shape2, mean = 0, sd = 1, log = FALSE)
pbetanorm(q, shape1, shape2, mean = 0, sd = 1,
         lower.tail = TRUE, log.p = FALSE)
qbetanorm(p, shape1, shape2, mean = 0, sd = 1,
         lower.tail = TRUE, log.p = FALSE)
rbetanorm(n, shape1, shape2, mean = 0, sd = 1)
Arguments

- \( x, q \) vector of quantiles.
- \( p \) vector of probabilities.
- \( n \) number of observations. Same as \texttt{runif}.
- \( \text{shape1, shape2} \) the two (positive) shape parameters of the standard beta distribution. They are called \( a \) and \( b \) respectively in \texttt{beta}.
- \( \text{mean, sd} \) the mean and standard deviation of the univariate normal distribution (\texttt{Normal}).
- \( \text{log, log.p} \) Logical. If \texttt{TRUE} then all probabilities \( p \) are given as \( \log(p) \).
- \( \text{lower.tail} \) Logical. If \texttt{TRUE} then the upper tail is returned, i.e., one minus the usual answer.

Details

The function \texttt{betauninormal}, the \texttt{VGAM} family function for estimating the parameters, has not yet been written.

Value

dbetanorm gives the density, pbetanorm gives the distribution function, qbetanorm gives the quantile function, and rbetanorm generates random deviates.

Author(s)

T. W. Yee

References


Examples

```r
## Not run:
shape1 <- 0.1; shape2 <- 4; m <- 1
x <- seq(-10, 2, len = 501)
plot(x, dbetanorm(x, shape1, shape2, m = m), type = "l", ylim = 0:1, las = 1,
     ylab = paste("betanorm","","",shape1,"","",shape2,"","",m="","",sd=1""), sep = ""),
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Gray lines are the 10,20,...,90 percentiles", col = "blue")
lines(x, pbetanorm(x, shape1, shape2, m = m), col = "orange")
abline(h = 0, col = "black")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qbetanorm(probs, shape1, shape2, m = m)
lines(Q, dbetanorm(Q, shape1, shape2, m = m), col = "gray50", lty = 2, type = "h")
lines(Q, pbetanorm(Q, shape1, shape2, m = m), col = "gray50", lty = 2, type = "h")
abline(h = probs, col = "gray50", lty = 2)
pbetanorm(Q, shape1, shape2, m = m) - probs # Should be all 0

## End(Not run)
```
The Beta-Prime Distribution

Description

Estimation of the two shape parameters of the beta-prime distribution by maximum likelihood estimation.

Usage

betaprime(link = "loge", i1 = 2, i2 = NULL, zero = NULL)

Arguments

- **link**: Parameter link function applied to the two (positive) shape parameters. See **Links** for more choices.
- **i1, i2**: Initial values for the first and second shape parameters. A NULL value means it is obtained in the initialize slot. Note that i2 is obtained using i1.
- **zero**: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The value must be from the set \{1,2\} corresponding respectively to shape1 and shape2 respectively. If zero=NULL then both parameters are modelled with the explanatory variables.

Details

The beta-prime distribution is given by

\[
f(y) = y^{\text{shape}1-1}(1 + y)^{-\text{shape}1-\text{shape}2}/B(\text{shape}1, \text{shape}2)
\]

for \( y > 0 \). The shape parameters are positive, and here, \( B \) is the beta function. The mean of \( Y \) is \( \text{shape}1/(\text{shape}2 - 1) \) provided \( \text{shape}2 > 1 \); these are returned as the fitted values.

If \( Y \) has a \( \text{Beta} (\text{shape}1, \text{shape}2) \) distribution then \( Y/(1-Y) \) and \( (1-Y)/Y \) have a \( \text{Betaprime} (\text{shape}1, \text{shape}2) \) and \( \text{Betaprime} (\text{shape}2, \text{shape}1) \) distribution respectively. Also, if \( Y_1 \) has a \( \text{gamma} (\text{shape}1) \) distribution and \( Y_2 \) has a \( \text{gamma} (\text{shape}2) \) distribution then \( Y_1/Y_2 \) has a \( \text{Betaprime} (\text{shape}1, \text{shape}2) \) distribution.

Value

An object of class "vglmff" (see **vglmff-class**). The object is used by modelling functions such as **vglm**, **rrvglm** and **vgam**.

Note

The response must have positive values only.

The beta-prime distribution is also known as the **beta distribution of the second kind** or the **inverted beta distribution**.
Author(s)
Thomas W. Yee

References

See Also
betaff, Beta.

Examples

```r
nn <- 1000
data <- data.frame(shape1 = exp(1), shape2 = exp(3))
bdeta <- transform(bdata, yb = rbeta(nn, shape1, shape2))
bdeta <- transform(bdata, y1 = (1-yb) / yb,
y2 = yb / (1-yb),
y3 = rgamma(nn, exp(3)) / rgamma(nn, exp(2)))

fit1 <- vglm(y1 ~ 1, betapprme, data = bdata, trace = TRUE)
coef(fit1, matrix = TRUE)

fit2 <- vglm(y2 ~ 1, betapprme, data = bdata, trace = TRUE)
coef(fit2, matrix = TRUE)

fit3 <- vglm(y3 ~ 1, betapprme, data = bdata, trace = TRUE)
coef(fit3, matrix = TRUE)

# Compare the fitted values
with(bdata, mean(y3))
head(fitted(fit3))
Coef(fit3) # Useful for intercept-only models
```

Description
Estimation of the shape parameters of the two-parameter beta distribution.

Usage

```r
betaR(lshape1 = "loge", lshape2 = "loge",
     i1 = NULL, i2 = NULL, trim = 0.05,
     A = 0, B = 1, parallel = FALSE, zero = NULL)
```
Arguments

lshape1, lshape2, il, i2

Details at CommonVGAMffArguments. See Links for more choices.

trim

An argument which is fed into mean(); it is the fraction (0 to 0.5) of observations to be trimmed from each end of the response y before the mean is computed. This is used when computing initial values, and guards against outliers.

A, B

Lower and upper limits of the distribution. The defaults correspond to the standard beta distribution where the response lies between 0 and 1.

parallel, zero

See CommonVGAMffArguments for more information.

Details

The two-parameter beta distribution is given by

\[ f(y) = (y - A)^{shape1-1} \times (B - y)^{shape2-1} / \left[ Beta(shape1, shape2) \times (B - A)^{shape1+shape2-1} \right] \]

for \( A < y < B \), and Beta(\( \ldots \)) is the beta function (see beta). The shape parameters are positive, and here, the limits \( A \) and \( B \) are known. The mean of \( Y \) is \( E(Y) = A + (B - A) \times \frac{shape1}{(shape1 + shape2)} \), and these are the fitted values of the object.

For the standard beta distribution the variance of \( Y \) is \( \frac{shape1 \times shape2}{[(1 + shape1 + shape2) \times (shape1 + shape2)^2]} \). If \( \sigma^2 = \frac{1}{1 + shape1 + shape2} \) then the variance of \( Y \) can be written \( \sigma^2 \mu(1 - \mu) \) where \( \mu = \frac{shape1}{(shape1 + shape2)} \) is the mean of \( Y \).

Another parameterization of the beta distribution involving the mean and a precision parameter is implemented in betaff.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Note

The response must have values in the interval (\( A, B \)). VGAM 0.7-4 and prior called this function betaff.

Author(s)

Thomas W. Yee

References


See Also

`betaff, Beta, genbetaII, betaII, betabinomialff, betageometric, betaprime, rbetageom, rbetanorm, kumar, simulate.vlm`.

Examples

```r
bdata <- data.frame(y = rbeta(n = 1000, shape1 = exp(0), shape2 = exp(1)))
fit <- vglm(y ~ 1, betaR(1shape1 = "identitylink", 1shape2 = "identitylink"),
           data = bdata, trace = TRUE, crit = "coef")
fit <- vglm(y ~ 1, betaR, data = bdata, trace = TRUE, crit = "coef")
coef(fit, matrix = TRUE)
Coef(fit) # Useful for intercept-only models

bdata <- transform(bdata, Y = 5 + 8 * y) # From 5 to 13, not 0 to 1
fit <- vglm(Y ~ 1, betaR(A = 5, B = 13), data = bdata, trace = TRUE)
Coef(fit)
c(meanY = with(bdata, mean(Y)), head(fitted(fit),2))
```

---

**Description**

Density, distribution function, and random generation for the (one parameter) bivariate Ali-Mikhail-Haq distribution.

**Usage**

```r
dbiamhcop(x1, x2, apar, log = FALSE)
pbiamhcop(q1, q2, apar)
rbiamhcop(n, apar)
```

**Arguments**

- `x1, x2, q1, q2` vector of quantiles.
- `n` number of observations. Same as `runif`
- `apar` the association parameter.
- `log` Logical. If `TRUE` then the logarithm is returned.

**Details**

See `biamhcop`, the VGAM family functions for estimating the parameter by maximum likelihood estimation, for the formula of the cumulative distribution function and other details.

**Value**

`dbiamhcop` gives the density, `pbiamhcop` gives the distribution function, and `rbiamhcop` generates random deviates (a two-column matrix).
biamhcop

Description

Estimate the association parameter of Ali-Mikhail-Haq’s bivariate distribution by maximum likelihood estimation.

Usage

biamhcop(lapar = "rhobit", iapar = NULL, imethod = 1, nsimEIM = 250)

Arguments

- **lapar**
  Link function applied to the association parameter \( \alpha \), which is real and \(-1 < \alpha < 1\). See [Links](#) for more choices.

- **iapar**
  Numeric. Optional initial value for \( \alpha \). By default, an initial value is chosen internally. If a convergence failure occurs try assigning a different value. Assigning a value will override the argument imethod.

- **imethod**
  An integer with value 1 or 2 which specifies the initialization method. If failure to converge occurs try the other value, or else specify a value for iapar.

- **nsimEIM**
  See [CommonVGAMffArguments](#) for more information.
Details

The cumulative distribution function is

\[ P(Y_1 \leq y_1, Y_2 \leq y_2) = \frac{y_1 y_2}{(1 - \alpha(1 - y_1)(1 - y_2))} \]

for \(-1 < \alpha < 1\). The support of the function is the unit square. The marginal distributions are the standard uniform distributions. When \(\alpha = 0\) the random variables are independent. This is an Archimedean copula.

Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

The response must be a two-column matrix. Currently, the fitted value is a matrix with two columns and values equal to 0.5. This is because each marginal distribution corresponds to a standard uniform distribution.

Author(s)

T. W. Yee and C. S. Chee

References


See Also

`rbiamhcop, bifgmcop, bigumbelIexp, rbilogis, simulate.vlm`.

Examples

```r
ymat <- rbiamhcop(1000, apar = rhobit(2, inverse = TRUE))
fit <- vglm(ymat ~ 1, biamhcop, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
```
Description

Density and random generation for the (one parameter) bivariate Clayton copula distribution.

Usage

dbiclaytoncop(x1, x2, apar = 0, log = FALSE)
rbiclaytoncop(n, apar = 0)

Arguments

- x1, x2: vector of quantiles. The x1 and x2 should both be in the interval (0, 1).
- n: number of observations. Same as rnorm.
- apar: the association parameter. Should be in the interval [0, ∞). The default corresponds to independence.
- log: Logical. If TRUE then the logarithm is returned.

Details

See biclaytoncop, the VGAM family functions for estimating the parameter by maximum likelihood estimation, for the formula of the cumulative distribution function and other details.

Value

dbiclaytoncop gives the density at point (x1,x2), rbicltyoncop generates random deviates (a two-column matrix).

Note

dbiclaytoncop() does not yet handle x1 = 0 and/or x2 = 0.

Author(s)

R. Feyter and T. W. Yee

References


See Also

biclaytoncop, binormalcop, binormal.
Examples

```r
## Not run: edge <- 0.01 # A small positive value
N <- 10^1; x <- seq(edge, 1.0 - edge, len = N); Rho <- 0.7
ox <- expand.grid(x, x)
zedd <- dbiclaytoncop(ox[, 1], ox[, 2], apar = Rho, log = TRUE)
par(mfrow = c(1, 2))
contour(x, x, matrix(zedd, N, N), col = "blue", labcex = 1.5, las = 1)
plot(biclaytoncop(1000, 2), col = "blue", las = 1)
## End(Not run)
```

biclaytoncop  Claydon Copula (Bivariate) Family Function

Description

Estimate the correlation parameter of the (bivariate) Clayton copula distribution by maximum likelihood estimation.

Usage

```r
biclaytoncop(lapar = "loge", iapar = NULL, imethod = 1,
            parallel = FALSE, zero = NULL)
```

Arguments

- `lapar`, `iapar`, `imethod`
  Details at `CommonVGAMffArguments`. See `Links` for more link function choices.
- `parallel`, `zero`
  Details at `CommonVGAMffArguments`. If `parallel = TRUE` then the constraint is also applied to the intercept.

Details

The cumulative distribution function is

\[
P(u_1, u_2; \alpha) = \left( u_1^{-\alpha} + u_2^{-\alpha} - 1 \right)^{-1/\alpha}
\]

for \(0 \leq \alpha\). Here, \(\alpha\) is the association parameter. The support of the function is the interior of the unit square; however, values of 0 and/or 1 are not allowed (currently). The marginal distributions are the standard uniform distributions. When \(\alpha = 0\) the random variables are independent.

This `VGAM` family function can handle multiple responses, for example, a six-column matrix where the first 2 columns is the first out of three responses, the next 2 columns being the next response, etc.

Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`. 
Note

The response matrix must have a multiple of two-columns. Currently, the fitted value is a matrix with the same number of columns and values equal to 0.5. This is because each marginal distribution corresponds to a standard uniform distribution.

This VGAM family function is fragile; each response must be in the interior of the unit square.

Author(s)

R. Feyter and T. W. Yee

References


See Also

rbiclaytoncop, dbiclaytoncop, kendall.tau.

Examples

```r
ymat <- rbiclaytoncop(n = (nn <- 1000), apar = exp(2))
bdata <- data.frame(y1 = ymat[, 1],
                    y2 = ymat[, 2],
                    y3 = ymat[, 1],
                    y4 = ymat[, 2],
                    x2 = runif(nn))

summary(bdata)
## Not run: plot(ymat, col = "blue")
fit1 <- vglm(cbind(y1, y2, y3, y4) ~ 1, # 2 responses, e.g., (y1,y2) is the first
             biclaytoncop, data = bdata,
             trace = TRUE, crit = "coef") # Sometimes a good idea

coef(fit1, matrix = TRUE)
Coeff(fit1)
head(fitted(fit1))
summary(fit1)

# Another example; apar is a function of x2
bdata <- transform(bdata, apar = exp(-0.5 + x2))
ymat <- rbiclaytoncop(n = nn, apar = with(bdata, apar))
bdata <- transform(bdata, y5 = ymat[, 1],
                   y6 = ymat[, 2])
fit2 <- vgam(cbind(y5, y6) ~ s(x2), data = bdata,
             biclaytoncop(lapar = "loge"), trace = TRUE)
## Not run: plot(fit2, lcol = "blue", scol = "orange", se = TRUE, las = 1)
```
BICvlm

Bayesian Information Criterion

Description

Calculates the Bayesian information criterion (BIC) for a fitted model object for which a log-likelihood value has been obtained.

Usage

```
BICvlm(object, ..., k = log(nobs(object)))
```

Arguments

- `object, ...` Same as `aicvlm`.
- `k` Numeric, the penalty per parameter to be used; the default is \( \log(n) \) where \( n \) is the number of observations).

Details

The so-called BIC or SBC (Schwarz’s Bayesian criterion) can be computed by calling `aicvlm` with a different \( k \) argument. See `aicvlm` for information and caveats.

Value

Returns a numeric value with the corresponding BIC, or \( \ldots \), depending on \( k \).

Warning

Like `aicvlm`, this code has not been double-checked. The general applicability of BIC for the VGLM/VGAM classes has not been developed fully. In particular, BIC should not be run on some VGAM family functions because of violation of certain regularity conditions, etc.

Many VGAM family functions such as `cumulative` can have the number of observations absorbed into the prior weights argument (e.g., weights in `vglm`), either before or after fitting. Almost all VGAM family functions can have the number of observations defined by the weights argument, e.g., as an observed frequency. BIC simply uses the number of rows of the model matrix, say, as defining \( n \), hence the user must be very careful of this possible error. Use at your own risk!!

Note

BIC, AIC and other ICs can have have many additive constants added to them. The important thing are the differences since the minimum value corresponds to the best model. Preliminary testing shows absolute differences with some VGAM family functions such as `gaussianff`, however, they should agree with non-normal families.

BIC has not been defined for QRR-VGLMs yet.
Farlie-Gumbel-Morgenstern’s Bivariate Distribution

Density, distribution function, and random generation for the (one parameter) bivariate Farlie-Gumbel-Morgenstern’s distribution.

Usage

\begin{verbatim}
dbifgmcop(x1, x2, apar, log = FALSE)
pbifgmcop(q1, q2, apar)
rbifgmcop(n, apar)
\end{verbatim}
Arguments

- x1, x2, q1, q2: vector of quantiles.
- n: number of observations. Same as in `runif`.
- apar: the association parameter.
- log: Logical. If TRUE then the logarithm is returned.

Details

See `bifgmcop`, the VGAM family functions for estimating the parameter by maximum likelihood estimation, for the formula of the cumulative distribution function and other details.

Value

dbifgmcop gives the density, pbifgmcop gives the distribution function, and rbifgmcop generates random deviates (a two-column matrix).

Author(s)

T. W. Yee

See Also

`bifgmcop`.

Examples

```r
## Not run: N <- 101; x <- seq(0.0, 1.0, len = N); apar <- 0.7
ox <- expand.grid(x, x)
zedd <- dbifgmcop(ox[, 1], ox[, 2], apar = apar)
contour(x, x, matrix(zedd, N, N), col = "blue")
zedd <- pbifgmcop(ox[, 1], ox[, 2], apar = apar)
contour(x, x, matrix(zedd, N, N), col = "blue")

plot(r <- rbifgmcop(n = 3000, apar = apar), col = "blue")
par(mfrow = c(1, 2))
hist(r[, 1]) # Should be uniform
hist(r[, 2]) # Should be uniform

## End(Not run)
```
bifgmcop

Farlie-Gumbel-Morgenstern’s Bivariate Distribution Family Function

Description
Estimate the association parameter of Farlie-Gumbel-Morgenstern’s bivariate distribution by maximum likelihood estimation.

Usage
bifgmcop(lapar = "rhobit", iapar = NULL, imethod = 1)

Arguments
lapar, iapar, imethod
Details at CommonVGAMffArguments. See Links for more link function choices.

Details
The cumulative distribution function is
\[ P(Y_1 \leq y_1, Y_2 \leq y_2) = y_1 y_2 (1 + \alpha (1 - y_1)(1 - y_2)) \]
for \(-1 < \alpha < 1\). The support of the function is the unit square. The marginal distributions are the standard uniform distributions. When \(\alpha = 0\) the random variables are independent.

Value
An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note
The response must be a two-column matrix. Currently, the fitted value is a matrix with two columns and values equal to 0.5. This is because each marginal distribution corresponds to a standard uniform distribution.

Author(s)
T. W. Yee

References
**Description**

Estimate the association parameter of FGM bivariate exponential distribution by maximum likelihood estimation.

**Usage**

```r
bifgexp(lapar = "rhobit", iapar = NULL, tola0 = 0.01, imethod = 1)
```

**Arguments**

- `lapar` Link function for the association parameter \( \alpha \), which lies between \(-1\) and \(1\). See [Links](#) for more choices and other information.
- `iapar` Numeric. Optional initial value for \( \alpha \). By default, an initial value is chosen internally. If a convergence failure occurs try assigning a different value. Assigning a value will override the argument `imethod`.
- `tola0` Positive numeric. If the estimate of \( \alpha \) has an absolute value less than this then it is replaced by this value. This is an attempt to fix a numerical problem when the estimate is too close to zero.
- `imethod` An integer with value 1 or 2 which specifies the initialization method. If failure to converge occurs try the other value, or else specify a value for `ia`.

**Details**

The cumulative distribution function is

\[
P(Y_1 \leq y_1, Y_2 \leq y_2) = e^{-y_1-y_2}(1 + \alpha[1 - e^{-y_1}][1 - e^{-y_2}]) + 1 - e^{-y_1} - e^{-y_2}
\]

for \( \alpha \) between \(-1\) and \(1\). The support of the function is for \( y_1 > 0 \) and \( y_2 > 0 \). The marginal distributions are an exponential distribution with unit mean. When \( \alpha = 0 \) then the random variables

---

### Examples

```r
ymat <- rbifgmcop(n = 1000, apar = rhobit(3, inverse = TRUE))
## Not run: plot(ymat, col = "blue")
fit <- vglm(ymat ~ 1, fam = bifgmcop, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
head(fitted(fit))
```
are independent, and this causes some problems in the estimation process since the distribution no longer depends on the parameter.

A variant of Newton-Raphson is used, which only seems to work for an intercept model. It is a very good idea to set `trace = TRUE`.

**Value**

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

**Note**

The response must be a two-column matrix. Currently, the fitted value is a matrix with two columns and values equal to 1. This is because each marginal distribution corresponds to an exponential distribution with unit mean.

This VGAM family function should be used with caution.

**Author(s)**

T. W. Yee

**References**


**See Also**

`bifgmcop`, `bigumbeliexp`.

**Examples**

```r
N <- 1000; mdata <- data.frame(y1 = rexp(N), y2 = rexp(N))
## Not run: plot(ymat)
fit <- vglm(cbind(y1, y2) ~ 1, bifgmexp, data = mdata, trace = TRUE)
fit <- vglm(cbind(y1, y2) ~ 1, bifgmexp, data = mdata, # This may fail
           trace = TRUE, crit = "coef")
coef(fit, matrix = TRUE)
Coef(fit)
head(fitted(fit))
```
Frank's Bivariate Distribution Family Function

Description

Estimate the association parameter of Frank's bivariate distribution by maximum likelihood estimation.

Usage

bifrankcop(lapar = "loge", iapar = 2, nsimEIM = 250)

Arguments

- **lapar**: Link function applied to the (positive) association parameter $\alpha$. See Links for more choices.
- **iapar**: Numeric. Initial value for $\alpha$. If a convergence failure occurs try assigning a different value.
- **nsimEIM**: See CommonVGAMffArguments.

Details

The cumulative distribution function is

$$P(Y_1 \leq y_1, Y_2 \leq y_2) = H_\alpha(y_1, y_2) = \log_\alpha[1 + (\alpha^{y_1} - 1)(\alpha^{y_2} - 1)/(\alpha - 1)]$$

for $\alpha \neq 1$. Note the logarithm here is to base $\alpha$. The support of the function is the unit square.

When $0 < \alpha < 1$ the probability density function $h_\alpha(y_1, y_2)$ is symmetric with respect to the lines $y_2 = y_1$ and $y_2 = 1 - y_1$. When $\alpha > 1$ then $h_\alpha(y_1, y_2) = h_{1/\alpha}(1 - y_2, y_2)$.

If $\alpha = 1$ then $H(y_1, y_2) = y_1y_2$, i.e., uniform on the unit square. As $\alpha$ approaches 0 then $H(y_1, y_2) = \min(y_1, y_2)$. As $\alpha$ approaches infinity then $H(y_1, y_2) = \max(0, y_1 + y_2 - 1)$.

The default is to use Fisher scoring implemented using rbifrankcop. For intercept-only models an alternative is to set nsimEIM=NULL so that a variant of Newton-Raphson is used.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

The response must be a two-column matrix. Currently, the fitted value is a matrix with two columns and values equal to a half. This is because the marginal distributions correspond to a standard uniform distribution.
**bigamma.mckay**

**Author(s)**

T. W. Yee

**References**


**See Also**

`rbifrankcop, bifgmcop, simulate.vlm`.

**Examples**

```r
## Not run:
ymat <- rbifrankcop(n = 2000, apar = exp(4))
plot(ymat, col = "blue")
fit <- vglm(ymat ~ 1, fam = bifrankcop, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
vcov(fit)
head(fitted(fit))
summary(fit)

## End(Not run)
```

---

**bigamma.mckay**  
*Bivariate Gamma: McKay’s Distribution*

**Description**

Estimate the three parameters of McKay’s bivariate gamma distribution by maximum likelihood estimation.

**Usage**

```r
bigamma.mckay(lscale = "loge", lshape1 = "loge", lshape2 = "loge",
iscale = NULL, ishape1 = NULL, ishape2 = NULL,
imethod = 1, zero = 2:3)
```

**Arguments**

- `lscale`, `lshape1`, `lshape2`  
  Link functions applied to the (positive) parameters $a$, $p$ and $q$ respectively. See `Links` for more choices.
- `iscale`, `ishape1`, `ishape2`  
  Optional initial values for $a$, $p$ and $q$ respectively. The default is to compute them internally.
- `imethod, zero`  
  See `CommonVGAMffArguments`.
One of the earliest forms of the bivariate gamma distribution has a joint probability density function given by

\[
f(y_1, y_2; a, p, q) = \frac{1}{a^{p+q}y_1^{p-1}y_2^{q-1}} \exp(-y_2/a)/[\Gamma(p)\Gamma(q)]
\]

for \( a > 0, \ p > 0, \ q > 0 \) and \( 0 < y_1 < y_2 \) (Mckay, 1934). Here, \( \Gamma \) is the gamma function, as in \texttt{gamma}. By default, the linear/additive predictors are \( \eta_1 = \log(a), \ \eta_2 = \log(p), \ \eta_3 = \log(q) \).

The marginal distributions are gamma, with shape parameters \( p \) and \( p + q \) respectively, but they have a common scale parameter \( a \). Pearson’s product-moment correlation coefficient of \( y_1 \) and \( y_2 \) is \( \sqrt{p/(p + q)} \). This distribution is also known as the bivariate Pearson type III distribution. Also, \( Y_2 - y_1 \), conditional on \( Y_1 = y_1 \), has a gamma distribution with shape parameter \( q \).

Value

An object of class "\texttt{vglmff}" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm} and \texttt{vgam}.

Note

The response must be a two column matrix where the first column is \( y_1 \) and the second \( y_2 \). It is necessary that each element of the vectors \( y_1 \) and \( y_2 - y_1 \) be positive. Currently, the fitted value is a matrix with two columns; the first column has values \( ap \) for the marginal mean of \( y_1 \), while the second column has values \( a(p + q) \) for the marginal mean of \( y_2 \) (all evaluated at the final iteration).

Author(s)

T. W. Yee

References


See Also

\texttt{gamma2}.

Examples

```r
shape1 <- exp(1); shape2 <- exp(2); scalepar <- exp(3)
ndata <- data.frame(y1 = rgamma(nn <- 1000, shape = shape1, scale = scalepar))
ndata <- transform(ndata, zedd = rgamma(nn, shape = shape2, scale = scalepar))
ndata <- transform(ndata, y2 = y1 + zedd)  # Z is defined as Y2-y1|Y1=y1
fit <- vglm(cbind(y1, y2) ~ 1, bigamma.mckay, data = ndata, trace = TRUE)
```
bigumbellexp

Description

Estimate the association parameter of Gumbel’s Type I bivariate distribution by maximum likelihood estimation.

Usage

bigumbellexp(lapar = "identitylink", iapar = NULL, imethod = 1)

Arguments

lapar Link function applied to the association parameter \( \alpha \). See Links for more choices.

iapar Numeric. Optional initial value for \( \alpha \). By default, an initial value is chosen internally. If a convergence failure occurs try assigning a different value. Assigning a value will override the argument imethod.

imethod An integer with value 1 or 2 which specifies the initialization method. If failure to converge occurs try the other value, or else specify a value for ia.

Details

The cumulative distribution function is

\[
P(Y_1 \leq y_1, Y_2 \leq y_2) = e^{-y_1 - y_2 + \alpha y_1 y_2} + 1 - e^{-y_1} - e^{-y_2}
\]

for real \( \alpha \). The support of the function is for \( y_1 > 0 \) and \( y_2 > 0 \). The marginal distributions are an exponential distribution with unit mean.

A variant of Newton-Raphson is used, which only seems to work for an intercept model. It is a very good idea to set trace=TRUE.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.
Note

The response must be a two-column matrix. Currently, the fitted value is a matrix with two columns and values equal to 1. This is because each marginal distribution corresponds to an exponential distribution with unit mean.

This VGAM family function should be used with caution.

Author(s)

T. W. Yee

References


See Also

bifgexp.

Examples

```r
nn <- 1000
gdata <- data.frame(y1 = rexp(nn), y2 = rexp(nn))
## Not run: with(gdata, plot(cbind(y1, y2)))
fit <- vglm(cbind(y1, y2) ~ 1, bigumbellexp, data = gdata, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
head(fitted(fit))
```

---

**bilogis**

*Bivariate Logistic Distribution*

Description

Density, distribution function, quantile function and random generation for the 4-parameter bivariate logistic distribution.

Usage

```r
dbilogis(x1, x2, loc1 = 0, scale1 = 1, loc2 = 0, scale2 = 1, log = FALSE)
pbilogis(q1, q2, loc1 = 0, scale1 = 1, loc2 = 0, scale2 = 1)
rbilogis(n, loc1 = 0, scale1 = 1, loc2 = 0, scale2 = 1)
```
Arguments

x1, x2, q1, q2 vector of quantiles.
n number of observations. Same as rlogis.
loc1, loc2 the location parameters l1 and l2.
scale1, scale2 the scale parameters s1 and s2.
log Logical. If log = TRUE then the logarithm of the density is returned.

Details

See bilogis, the VGAM family function for estimating the four parameters by maximum likelihood estimation, for the formula of the cumulative distribution function and other details.

Value
dbilogis gives the density, pbilogis gives the distribution function, and rbilogis generates random deviates (a two-column matrix).

Note

Gumbel (1961) proposed two bivariate logistic distributions with logistic distribution marginals, which he called Type I and Type II. The Type I is this one. The Type II belongs to the Morgenstern type. The biamhcop distribution has, as a special case, this distribution, which is when the random variables are independent.

Author(s)

T. W. Yee

References


See Also

bilogistic, biamhcop.

Examples

```R
## Not run: par(mfrow = c(1, 3))
ymat <- rbilogis(n = 2000, loc1 = 5, loc2 = 7, scale2 = exp(1))
myxlim <- c(-2, 15); myylim <- c(-10, 30)
plot(ymat, xlim = myxlim, ylim = myylim)

N <- 100
x1 <- seq(myxlim[1], myxlim[2], len = N)
x2 <- seq(myylim[1], myylim[2], len = N)
ox <- expand.grid(x1, x2)
z <- dbilogis(x[,1], x[,2], loc1 = 5, loc2 = 7, scale2 = exp(1))
```
bilogistic

Bivariate Logistic Distribution Family Function

Description

Estimates the four parameters of the bivariate logistic distribution by maximum likelihood estimation.

Usage

bilogistic(location = "identitylink", scale = "loge",
iloc = NULL, iscale1 = NULL, iloc2 = NULL, iscale2 = NULL,
imethod = 1, zero = NULL)

Arguments

location Link function applied to both location parameters \( l_1 \) and \( l_2 \). See Links for more choices.
scale Parameter link function applied to both (positive) scale parameters \( s_1 \) and \( s_2 \). See Links for more choices.
iloc, iloc2 Initial values for the location parameters. By default, initial values are chosen internally using imethod. Assigning values here will override the argument imethod.
iscale1, iscale2 Initial values for the scale parameters. By default, initial values are chosen internally using imethod. Assigning values here will override the argument imethod.
imethod An integer with value 1 or 2 which specifies the initialization method. If failure to converge occurs try the other value.
zero An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The default is none of them. If used, choose values from the set \{1,2,3,4\}.

Details

The four-parameter bivariate logistic distribution has a density that can be written as

\[
f(y_1, y_2; l_1, s_1, l_2, s_2) = \frac{2 \exp\left[-(y_1 - l_1)/s_1 - (y_2 - l_2)/s_2\right]}{s_1 s_2 (1 + \exp[-(y_1 - l_1)/s_1] + \exp[-(y_2 - l_2)/s_2])^3}
\]
where \( s_1 > 0, s_2 > 0 \) are the scale parameters, and \( l_1 \) and \( l_2 \) are the location parameters. Each of the two responses are unbounded, i.e., \(-\infty < y_j < \infty\). The mean of \( Y_j \) is \( l_j \) etc. The fitted values are returned in a 2-column matrix. The cumulative distribution function is

\[
F(y_1, y_2; l_1, s_1, l_2, s_2) = (1 + \exp[-(y_1 - l_1)/s_1] + \exp[-(y_2 - l_2)/s_2])^{-1}
\]

The marginal distribution of \( Y_1 \) is

\[
P(Y_1 \leq y_1) = F(y_1; l_1, s_1) = (1 + \exp[-(y_1 - l_1)/s_1])^{-1}.
\]

By default, \( \eta_1 = l_1, \eta_2 = \log(s_1), \eta_3 = l_2, \eta_4 = \log(s_2) \) are the linear/additive predictors.

Value

An object of class "vglmff" (see \code{vglmff-class}). The object is used by modelling functions such as \code{vglm}, \code{rrvglm} and \code{vgam}.

Note

This family function uses the BFGS quasi-Newton update formula for the working weight matrices. Consequently the estimated variance-covariance matrix may be inaccurate or simply wrong! The standard errors must be therefore treated with caution; these are computed in functions such as \code{vcov()} and \code{summary()}.

Author(s)

T. W. Yee

References


See Also

\code{logistic}, \code{rbilogis}.

Examples

ymat <- rbilogis(n <- 1000, loc1 = 5, loc2 = 7, scale2 = exp(1))
## Not run: plot(ymat)
fit <- vglm(ymat ~ 1, fam = bilogistic, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
head(fitted(fit))
vcov(fit)
head(weights(fit, type = "work"))
summary(fit)
Binom2.or  Bivariate Binary Regression with an Odds Ratio

Description

Density and random generation for a bivariate binary regression model using an odds ratio as the measure of dependency.

Usage

\[
\text{rbinom2.or}(n, \mu_1, \\
\mu_2 = \text{if (exchangeable) } \mu_1 \text{ else stop("argument 'mu2' not specified")}, \\
\text{oratio} = 1, \text{exchangeable} = \text{FALSE}, \text{tol} = 0.001, \text{twoCols} = \text{TRUE}, \\
\text{colnames} = \text{if (twoCols) c("y1","y2") else c("00", "01", "10", "11"), ErrorCheck = TRUE}
\]

\[
\text{dbinom2.or}(\mu_1, \\
\mu_2 = \text{if (exchangeable) } \mu_1 \text{ else stop("'mu2' not specified")),} \\
\text{oratio} = 1, \text{exchangeable} = \text{FALSE}, \text{tol} = 0.001, \\
\text{colnames} = \text{c("00", "01", "10", "11"), ErrorCheck = TRUE}
\]

Arguments

\(n\)  number of observations. Same as in \text{runif}. The arguments \(\mu_1, \mu_2, \text{oratio}\) are recycled to this value.

\(\mu_1, \mu_2\)  The marginal probabilities. Only \(\mu_1\) is needed if \text{exchangeable} = \text{TRUE}. Values should be between 0 and 1.

\(\text{oratio}\)  Odds ratio. Must be numeric and positive. The default value of unity means the responses are statistically independent.

\(\text{exchangeable}\)  Logical. If \text{TRUE}, the two marginal probabilities are constrained to be equal.

\(\text{twoCols}\)  Logical. If \text{TRUE}, then a \(n \times 2\) matrix of 1s and 0s is returned. If \text{FALSE}, then a \(n \times 4\) matrix of 1s and 0s is returned.

\(\text{colnames}\)  The \text{dimnames} argument of \text{matrix} is assigned \text{list(NULL, colnames)}.

\(\text{tol}\)  Tolerance for testing independence. Should be some small positive numerical value.

\(\text{ErrorCheck}\)  Logical. Do some error checking of the input parameters?

Details

The function \text{rbinom2.or} generates data coming from a bivariate binary response model. The data might be fitted with the \text{VGAM} family function \text{binom2.or}.

The function \text{dbinom2.or} does not really compute the density (because that does not make sense here) but rather returns the four joint probabilities.
Value

The function `rbinom2.or` returns either a 2 or 4 column matrix of 1s and 0s, depending on the argument `twoCols`.

The function `dbinom2.or` returns a 4 column matrix of joint probabilities; each row adds up to unity.

Author(s)

T. W. Yee

See Also

`binom2.or`.

Examples

```r
nn <- 2000  # Example 1
ymat <- rbinom2.or(n = nn, mu1 = 0.8, oratio = exp(2), exch = TRUE)
(mytab <- table(ymat[, 1], ymat[, 2], dnn = c("Y1", "Y2")))
(myor <- mytab["0","0"] * mytab["1","1"] / (mytab["0","0"] * mytab["0","1"]))
fit <- vglm(ymat - 1, binom2.or(exch = TRUE))
coef(fit, matrix = TRUE)

bdata <- data.frame(x2 = sort(runif(nn)))  # Example 2
bdata <- transform(bdata, mu1 = logit(-2 + 4*x2, inverse = TRUE),
                   mu2 = logit(-1 + 3*x2, inverse = TRUE))
dmat <- with(bdata, dbinom2.or(mu1 = mu1, mu2 = mu2, oratio = exp(2)))
ymat <- with(bdata, rbinom2.or(n = nn, mu1 = mu1, mu2 = mu2, oratio = exp(2)))
fit2 <- vglm(ymat ~ x2, binom2.or, data = bdata)
coef(fit2, matrix = TRUE)
## Not run:
matplot(with(bdata, x2), dmat, lty = 1:4, col = 1:4, type = "l",
        main = "Joint probabilities", ylim = 0:1, lwd = 2,
        ylab = "Probabilities", xlab = "x2", las = 1)
legend(x = 0, y = 0.5, lty = 1:4, col = 1:4, lwd = 2,
       legend = c("1 = (y1=0, y2=0)", "2 = (y1=0, y2=1)",
                  "3 = (y1=1, y2=0)", "4 = (y1=1, y2=1)"))
## End(Not run)
```

Description

Fits a Palmgren (bivariate odds-ratio model, or bivariate logistic regression) model to two binary responses. Actually, a bivariate logistic/probit/cloglog/cauchit model can be fitted. The odds ratio is used as a measure of dependency.
Usage

binom2.or(lmu = "logit", lm1 = lmu, lm2 = lmu, loratio = "loge",
        lmu1 = NULL, lmu2 = NULL, ioratio = NULL, zero = 3,
        exchangeable = FALSE, tol = 0.001, more.robust = FALSE)

Arguments

lmu Link function applied to the two marginal probabilities. See \texttt{Links} for more choices. See the note below.

lm1, lm2 Link function applied to the first and second of the two marginal probabilities.

loratio Link function applied to the odds ratio. See \texttt{Links} for more choices.

imu1, imu2, ioratio Optional initial values for the marginal probabilities and odds ratio. See \texttt{CommonVGAMfArguments} for more details. In general good initial values are often required so use these arguments if convergence failure occurs.

zero Which linear/additive predictor is modelled as an intercept only? A NULL means none.

exchangeable Logical. If TRUE, the two marginal probabilities are constrained to be equal.

tol Tolerance for testing independence. Should be some small positive numerical value.

more.robust Logical. If TRUE then some measures are taken to compute the derivatives and working weights more robustly, i.e., in an attempt to avoid numerical problems. Currently this feature is not debugged if set TRUE.

Details

Also known informally as the Palmgren model, the bivariate logistic model is a full-likelihood based model defined as two logistic regressions plus \( \log(\text{oratio}) = \eta_3 \) where \( \eta_3 \) is the third linear/additive predictor relating the odds ratio to explanatory variables. Explicitly, the default model is

\[
\text{logit}[P(Y_j = 1)] = \eta_j, \quad j = 1, 2
\]

for the marginals, and

\[
\log[P(Y_{00} = 1)P(Y_{11} = 1)/P(Y_{01} = 1)P(Y_{10} = 1)] = \eta_3,
\]

specifies the dependency between the two responses. Here, the responses equal 1 for a success and 0 for a failure, and the odds ratio is often written \( \psi = p_{00}p_{11}/(p_{10}p_{01}) \). The model is fitted by maximum likelihood estimation since the full likelihood is specified. The two binary responses are independent if and only if the odds ratio is unity, or equivalently, the log odds ratio is zero. Fisher scoring is implemented.

The default models \( \eta_3 \) as a single parameter only, i.e., an intercept-only model, but this can be circumvented by setting zero = NULL in order to model the odds ratio as a function of all the explanatory variables. The function \texttt{binom2.or()} can handle other probability link functions such as \texttt{probit}, \texttt{cloglog} and \texttt{cauchit} links as well, so is quite general. In fact, the two marginal probabilities can each have a different link function. A similar model is the \textit{bivariate probit model}
(binom2.rho), which is based on a standard bivariate normal distribution, but the bivariate probit model is less interpretable and flexible.

The exchangeable argument should be used when the error structure is exchangeable, e.g., with eyes or ears data.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

When fitted, the fitted.values slot of the object contains the four joint probabilities, labelled as $(Y_1, Y_2) = (0,0), (0,1), (1,0), (1,1)$, respectively. These estimated probabilities should be extracted with the fitted generic function.

Note

At present we call binom2.or families a bivariate odds-ratio model. The response should be either a 4-column matrix of counts (whose columns correspond to $(Y_1, Y_2) = (0,0), (0,1), (1,0), (1,1)$ respectively), or a two-column matrix where each column has two distinct values, or a factor with four levels. The function rbinom2.or may be used to generate such data. Successful convergence requires at least one case of each of the four possible outcomes.

By default, a constant odds ratio is fitted because zero = 3. Set zero = NULL if you want the odds ratio to be modelled as a function of the explanatory variables; however, numerical problems are more likely to occur.

The argument lmu, which is actually redundant, is used for convenience and for upward compatibility: specifying lmu only means the link function will be applied to lmu1 and lmu2. Users who want a different link function for each of the two marginal probabilities should use the lmu1 and lmu2 arguments, and the argument lmu is then ignored. It doesn’t make sense to specify exchangeable = TRUE and have different link functions for the two marginal probabilities.

Regarding Yee and Dirnbock (2009), the xij (see vglm.control) argument enables environmental variables with different values at the two time points to be entered into an exchangeable binom2.or model. See the author’s webpage for sample code.

Author(s)

Thomas W. Yee

References


See Also

rbinom2.or, binom2.rho, loglinb2, zipebcom, coalminers, binomialff, logit, probit, cloglog, cauchit.

Examples

# Fit the model in Table 6.7 in McCullagh and Nelder (1989)
coalminers <- transform(coalminers, Age = (age - 42) / 5)
fit <- vglm(cbind(nBnW, nBW, BnW, BW) ~ Age,
            binom2.or(zero = NULL), data = coalminers)
fitted(fit)
summary(fit)
coef(fit, matrix = TRUE)
c(weights(fit, type = "prior")) * fitted(fit) # Table 6.8

## Not run: with(coalminers, matplot(Age, fitted(fit), type = "l", las = 1,
                                    xlab = "(age - 42) / 5", lwd = 2))
with(coalminers, matpoints(Age, depvar(fit), col=1:4))
legend(x = -4, y = 0.5, lty = 1:4, col = 1:4, lwd = 2,
       legend = c("1 = (Breathlessness=0, Wheeze=0)",
                  "2 = (Breathlessness=0, Wheeze=1)",
                  "3 = (Breathlessness=1, Wheeze=0)",
                  "4 = (Breathlessness=1, Wheeze=1)"))
## End(Not run)

# Another model: pet ownership
## Not run: data(xs.nz, package = "VGAMdata")
# More homogeneous:
petdata <- subset(xs.nz, ethnicity == "European" & age < 70 & sex == "M")
petdata <- na.omit(petdata[, c("cat", "dog", "age")])
summary(petdata)
with(petdata, table(cat, dog)) # Can compute the odds ratio

fit <- vgam(cbind((1-cat) * (1-dog), (1-cat) * dog,
                   cat * (1-dog), cat * dog) ~ s(age, df = 5),
            binom2.or(zero = 3), data = petdata, trace = TRUE)
colSums(depvar(fit))
coef(fit, matrix = TRUE)

## Not run: # Plot the estimated probabilities
ooo <- order(with(petdata, age))
matplot(with(petdata, age)[ooo], fitted(fit)[ooo, ], type = "l",
        xlab = "Age", ylab = "Probability", main = "Pet ownership",
        ylim = c(0, max(fitted(fit))), las = 1, lwd = 1.5)
legend("topleft", col=1:4, lty = 1:4, leg = c("no cat or dog ",
                                                 "dog only", "cat only", "cat and dog"), lwd = 1.5)
## End(Not run)
Binom2.rho

Binom2.rho  Bivariate Probit Model

Description
Density and random generation for a bivariate probit model. The correlation parameter rho is the measure of dependency.

Usage
rbinom2.rho(n, mu1, mu2 = if (exchangeable) mu1 else stop("argument 'mu2' not specified"), rho = 0, exchangeable = FALSE, twoCols = TRUE, colnames = if (twoCols) c("y1", "y2") else c("00", "01", "10", "11"), ErrorCheck = TRUE)
dbinom2.rho(mu1, mu2 = if (exchangeable) mu1 else stop("'mu2' not specified"), rho = 0, exchangeable = FALSE, colnames = c("00", "01", "10", "11"), ErrorCheck = TRUE)

Arguments
n  number of observations. Same as in runif. The arguments mu1, mu2, rho are recycled to this value.
mu1, mu2  The marginal probabilities. Only mu1 is needed if exchangeable = TRUE. Values should be between 0 and 1.
rho  The correlation parameter. Must be numeric and lie between −1 and 1. The default value of zero means the responses are uncorrelated.
exchangeable  Logical. If TRUE, the two marginal probabilities are constrained to be equal.
twoCols  Logical. If TRUE, then a n × 2 matrix of 1s and 0s is returned. If FALSE, then a n × 4 matrix of 1s and 0s is returned.
colnames  The dimnames argument of matrix is assigned list(NULL, colnames).
ErrorCheck  Logical. Do some error checking of the input parameters?

Details
The function rbinom2.rho generates data coming from a bivariate probit model. The data might be fitted with the VGAM family function binom2.rho.

The function dbinom2.rho does not really compute the density (because that does not make sense here) but rather returns the four joint probabilities.
The function `rbinom2.rho` returns either a 2 or 4 column matrix of 1s and 0s, depending on the argument `twocols`.

The function `dbinom2.rho` returns a 4 column matrix of joint probabilities; each row adds up to unity.

Author(s)

T. W. Yee

See Also

`binom2.rho`.

Examples

```r
(myrho <- rhobit(2, inverse = TRUE)) # Example 1
ymat <- rbinom2.rho(nn <- 2000, mu1 = 0.8, rho = myrho, exch = TRUE)
(mytab <- table(ymat[, 1], ymat[, 2], dnn = c("Y1", "Y2")))
fit <- vglm(ymat ~ 1, binom2.rho(exch = TRUE))
coef(fit, matrix = TRUE)

bdata <- data.frame(x2 = sort(runif(nn))) # Example 2
bdata <- transform(bdata, mu1 = probit(-2+4*x2, inverse = TRUE),
                   mu2 = probit(-1+3*x2, inverse = TRUE))
dmat <- with(bdata, dbinom2.rho(mu1, mu2, myrho))
ymat <- with(bdata, rbinom2.rho(nn, mu1, mu2, myrho))
fit2 <- vglm(ymat ~ x2, binom2.rho, data = bdata)
coef(fit2, matrix = TRUE)
## Not run: matplot(with(bdata, x2), dmat, lty = 1:4, col = 1:4,
                type = "l", main = "Joint probabilities",
                ylim = 0:1, lwd = 2, ylab = "Probability")
legend(x = 0.25, y = 0.9, lty = 1:4, col = 1:4, lwd = 2,
       legend = c("1 = (y1=0, y2=0)", "2 = (y1=0, y2=1)",
                "3 = (y1=1, y2=0)", "4 = (y1=1, y2=1)"))
## End(Not run)
```

Description

Fits a bivariate probit model to two binary responses. The correlation parameter rho is the measure of dependency.
Usage

```r
binom2.rho(lrho = "rhobit", lmu = "probit", imu1 = NULL, imu2 = NULL,
    irho = NULL, imethod = 1, zero = 3,
    exchangeable = FALSE, grho = seq(-0.95, 0.95, by = 0.05),
    nsimEIM = NULL)
binom2.Rho(rho = 0, imu1 = NULL, imu2 = NULL,
    exchangeable = FALSE, nsimEIM = NULL)
```

Arguments

- `lrho`: Link function applied to the \( \rho \) association parameter. See `Links` for more choices.
- `lmu`: Link function applied to the marginal probabilities. Should be left alone.
- `irho`: Optional initial value for \( \rho \). If given, this should lie between \(-1\) and \(1\). See below for more comments.
- `imu1`, `imu2`: Optional initial values for the two marginal probabilities. May be a vector.
- `zero`: Which linear/additive predictor is modelled as an intercept only? A NULL means none. Numerically, the \( \rho \) parameter is easiest modelled as an intercept only, hence the default.
- `exchangeable`: Logical. If `TRUE`, the two marginal probabilities are constrained to be equal.
- `imethod`, `nsimEIM`, `grho`: See `CommonVGAMffArguments` for more information. A value of at least 100 for `nsimEIM` is recommended; the larger the value the better.
- `rho`: Numeric vector. Values are recycled to the needed length, and ought to be in range.

Details

The bivariate probit model was one of the earliest regression models to handle two binary responses jointly. It has a probit link for each of the two marginal probabilities, and models the association between the responses by the \( \rho \) parameter of a standard bivariate normal distribution (with zero means and unit variances). One can think of the joint probabilities being \( \Phi(\eta_j, \eta_3; \rho) \) where \( \Phi \) is the cumulative distribution function of a standard bivariate normal distribution.

Explicitly, the default model is

\[
\text{probit}[P(Y_j = 1)] = \eta_j, \quad j = 1, 2
\]

for the marginals, and

\[
\text{rhobit}[\rho] = \eta_3.
\]

The joint probability \( P(Y_1 = 1, Y_2 = 1) = \Phi(\eta_1, \eta_2; \rho) \), and from these the other three joint probabilities are easily computed. The model is fitted by maximum likelihood estimation since the full likelihood is specified. Fisher scoring is implemented.

The default models \( \eta_3 \) as a single parameter only, i.e., an intercept-only model for rho, but this can be circumvented by setting `zero = NULL` in order to model rho as a function of all the explanatory variables.

The bivariate probit model should not be confused with a bivariate logit model with a probit link (see `binom2.or`). The latter uses the odds ratio to quantify the association. Actually, the bivariate
logit model is recommended over the bivariate probit model because the odds ratio is a more natural way of measuring the association between two binary responses.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

When fitted, the fitted.values slot of the object contains the four joint probabilities, labelled as \((Y_1, Y_2) = (0,0), (0,1), (1,0), (1,1)\), respectively.

Note

See binom2.or about the form of input the response should have.

By default, a constant \(\rho\) is fitted because zero = 3. Set zero = NULL if you want the \(\rho\) parameter to be modelled as a function of the explanatory variables. The value \(\rho\) lies in the interval \((-1, 1)\), therefore a rhobit link is default.

Converge problems can occur. If so, assign irho a range of values and monitor convergence (e.g., set trace = TRUE). Else try imethod. Practical experience shows that local solutions can occur, and that irho needs to be quite close to the (global) solution. Also, imu1 and imu2 may be used.

This help file is mainly about binom2.rho(). binom2.Rho() fits a bivariate probit model with known \(\rho\). The inputted rho is saved in the misc slot of the fitted object, with rho as the component name.

In some econometrics applications (e.g., Freedman 2010, Freedman and Sekhon 2010) one response is used as an explanatory variable, e.g., a recursive binomial probit model. Such will not work here. Historically, the bivariate probit model was the first VGAM I ever wrote, based on Ashford and Sowden (1970). I don’t think they ever thought of it either! Hence the criticisms raised go beyond the use of what was originally intended.

Author(s)

Thomas W. Yee

References


See Also

rbinom2.rho, rhobit, pnorm2, binom2.or, loglinb2, coalminers, binomialff, rhobit, fisherz.
Examples

```r
coalminers <- transform(coalminers, Age = (age - 42) / 5)
fit <- vglm(cbind(nBnW, nBW, BnW, BW) ~ Age,
           binom2.rho, data = coalminers, trace = TRUE)
summary(fit)
coef(fit, matrix = TRUE)
```

Description

Family function for fitting generalized linear models to binomial responses, where the dispersion parameter may be known or unknown.

Usage

```r
binomialff(link = "logit", dispersion = 1, multiple.responses = FALSE,
         onedpar = FALSE, multiple.responses, parallel = FALSE,
         zero = NULL, bred = FALSE, earg.link = FALSE)
```

Arguments

- **link**: Link function; see Links and CommonVGAMffArguments for more information.
- **dispersion**: Dispersion parameter. By default, maximum likelihood is used to estimate the model because it is known. However, the user can specify `dispersion = 0` to have it estimated, or else specify a known positive value (or values if `multiple.responses` is TRUE).
- **multiple.responses**: Multivariate response? If TRUE, then the response is interpreted as \( M \) independent binary responses, where \( M \) is the number of columns of the response matrix. In this case, the response matrix should have \( Q \) columns consisting of counts (successes), and the weights argument should have \( Q \) columns consisting of the number of trials (successes plus failures).
  
  If FALSE and the response is a (2-column) matrix, then the number of successes is given in the first column, and the second column is the number of failures.
- **onedpar**: One dispersion parameter? If `multiple.responses`, then a separate dispersion parameter will be computed for each response (column), by default. Setting `onedpar = TRUE` will pool them so that there is only one dispersion parameter to be estimated.
- **parallel**: A logical or formula. Used only if `multiple.responses` is TRUE. This argument allows for the parallelism assumption whereby the regression coefficients for a variable is constrained to be equal over the \( M \) linear/additive predictors. If `parallel = TRUE` then the constraint is not applied to the intercepts.
An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \{1,2,\ldots,M\}, where \(M\) is the number of columns of the matrix response. See CommonVGAMffArguments for more information.

earg.link

Details at CommonVGAMffArguments.

bred

Details at CommonVGAMffArguments. Setting \(\text{bred} = \text{TRUE}\) should work for multiple responses (\(\text{multiple.responses} = \text{TRUE}\)) and all VGAM link functions; it has been tested for logit only (and it gives similar results to brglm but not identical), and further testing is required. One result from fitting bias reduced binary regression is that finite regression coefficients occur when the data is separable (see example below).

Details

This function is largely to mimic binomial, however there are some differences.

If the dispersion parameter is unknown, then the resulting estimate is not fully a maximum likelihood estimate (see pp.124–8 of McCullagh and Nelder, 1989).

A dispersion parameter that is less/greater than unity corresponds to under-/over-dispersion relative to the binomial model. Over-dispersion is more common in practice.

Setting \(\text{multiple.responses} = \text{TRUE}\) is necessary when fitting a Quadratic RR-VGLM (see cqo) because the response is a matrix of \(M\) columns (e.g., one column per species). Then there will be \(M\) dispersion parameters (one per column of the response matrix).

When used with cqo and cao, it may be preferable to use the cloglog link.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, vgam, rrvglm, cqo, and cao.

Warning

With a multivariate response, assigning a known dispersion parameter for each response is not handled well yet. Currently, only a single known dispersion parameter is handled well.

See the above note regarding bred.

The maximum likelihood estimate will not exist if the data is completely separable or quasi-completely separable. See Chapter 10 of Altman et al. (2004) for more details, and safeBinaryRegression. Yet to do: add a sepcheck = TRUE, say, argument to detect this problem and give an appropriate warning.

Note

If \(\text{multiple.responses}\) is FALSE (default) then the response can be of one of two formats: a factor (first level taken as failure), or a 2-column matrix (first column = successes) of counts. The argument weights in the modelling function can also be specified as any vector of positive values. In general, 1 means success and 0 means failure (to check, see the \(y\) slot of the fitted object). Note that a general vector of proportions of success is no longer accepted.

The notation \(M\) is used to denote the number of linear/additive predictors.
If `multiple.responses` is `TRUE`, then the matrix response can only be of one format: a matrix of 1's and 0's (1 = success).

The call `binomialff(dispersion = 0, ...)` is equivalent to `quasibinomialff(...). The latter was written so that R users of `quasibinomial()` would only need to add a “ff” to the end of the family function name.

Regardless of whether the dispersion parameter is to be estimated or not, its value can be seen from the output from the `summary()` of the object.

Fisher scoring is used. This can sometimes fail to converge by oscillating between successive iterations (Ridout, 1990). See the example below.

**Author(s)**

Thomas W. Yee

**References**


**See Also**

`quasibinomialff`, `links`, `rrvglm`, `cqo`, `cao`, `betabinomial`, `posbinomial`, `zibinomial`, `double.expbinomial`, `seq2binomial`, `amlbinomial`, `simplex`, `binomial`, `simulate.vlm`, `safeBinaryRegression`.

**Examples**

```r
quasibinomialff()
quasibinomialff(link = "probit")

shunua <- hunua[sort.list(with(hunua, altitude)), ] # Sort by altitude
fit <- vglm(agaaus ~ poly(altitude, 2), binomialff(link = cloglog), shunua)
## Not run:
plot(agaaus ~ jitter(altitude), shunua, col = "blue", ylab = "P(Agaaus = 1)",
     main = "Presence/absence of Agathis australis", las = 1)
with(shunua, lines(altitude, fitted(fit), col = "orange", lwd = 2))
## End(Not run)

# Fit two species simultaneously
fit2 <- vgam(cbind(agaaus, kniexc) ~ s(altitude),
             binomialff(multiple.responses = TRUE), data = shunua)
## Not run:
with(shunua, matplot(altitude, fitted(fit2), type = "l",
                      main = "Two species response curves", las = 1))
## End(Not run)
```
# Shows that Fisher scoring can sometime fail. See Ridout (1990).
ridout <- data.frame(v = c(1000, 100, 10), r = c(4, 3, 3), n = c(5, 5, 5))
(ridout <- transform(ridout, logv = log(v)))
# The iterations oscillates between two local solutions:
glm.fail <- glm(r / n ~ offset(logv) + 1, weight = n,
    binomial(link = "cloglog"), ridout, trace = TRUE)
coef(glm.fail)
# vglm()'s half-stepping ensures the MLE of -5.4007 is obtained:
vglm.ok <- vglm(cbind(r, n-r) ~ offset(logv) + 1,
    binomialff(link = cloglog), ridout, trace = TRUE)
coef(vglm.ok)

# Separable data
set.seed(123)
threshold <- 0
bdata <- data.frame(x2 = sort(rnorm(nn <- 100)))
bdata <- transform(bdata, y1 = ifelse(x2 < threshold, 0, 1))
fit <- vglm(y1 ~ x2, binomialff(bred = TRUE),
    data = bdata, criter = "coef", trace = TRUE)
coef(fit, matrix = TRUE) # Finite!!
summary(fit)
## Not run: plot(devpvar(fit) ~ x2, data = bdata, col = "blue", las = 1)
lines(fitted(fit) ~ x2, data = bdata, col = "orange")
abline(v = threshold, col = "gray", lty = "dashed")
## End(Not run)

---

### Binorm

**Bivariate normal distribution cumulative distribution function**

---

**Description**

Density, cumulative distribution function and random generation for the bivariate normal distribution.

**Usage**

```r
dbinorm(x1, x2, mean1 = 0, mean2 = 0, var1 = 1, var2 = 1, cov12 = 0, log = FALSE)
pbinorm(q1, q2, mean1 = 0, mean2 = 0, var1 = 1, var2 = 1, cov12 = 0)
rbinorm(n, mean1 = 0, mean2 = 0, var1 = 1, var2 = 1, cov12 = 0)
pnorm2(x1, x2, mean1 = 0, mean2 = 0, var1 = 1, var2 = 1, cov12 = 0)
```

**Arguments**

- `x1, x2, q1, q2` vector of quantiles.
- `mean1, mean2, var1, var2, cov12`
  - vector of means, variances and the covariance.
Binorm

\[ n \] number of observations. Same as \texttt{rnorm}.

\[ \text{log} \] Logical. If \text{log} = TRUE then the logarithm of the density is returned.

\section*{Details}

The default arguments correspond to the standard bivariate normal distribution with correlation parameter \( \rho = 0 \). That is, two independent standard normal distributions. Let \( s_\text{d1} \) (say) be \( \sqrt{\text{var}_1} \) and written \( \sigma_1 \), etc. Then the general formula for the correlation coefficient is \( \rho = \text{cov} / (\sigma_1 \sigma_2) \) where \text{cov} is argument \text{cov}\text{QR}. Thus if arguments \text{var}\text{QR} and \text{var\text{R}} are left alone then \text{cov\text{QR}} can be inputted with \text{rho}.

One can think of this function as an extension of \texttt{pnorm} to two dimensions, however note that the argument names have been changed for \texttt{VGAM} 0.9-1 onwards.

\section*{Value}

\texttt{dbinorm} gives the density, \texttt{pbinorm} gives the cumulative distribution function, \texttt{rbinorm} generates random deviates (\( n \) by 2 matrix).

\section*{Warning}

Being based on an approximation, the results of \texttt{pbinorm}() may be negative! Also, \texttt{pnorm2()} should be withdrawn soon; use \texttt{pbinorm}() instead because it is identical.

\section*{Note}

For \texttt{rbinorm()}, if the \( i \)th variance-covariance matrix is not positive-definite then the \( i \)th row is all NAs.

\section*{References}

\texttt{pbinorm()} is based on Donnelly (1973), the code was translated from FORTRAN to ratfor using \texttt{struct}, and then from ratfor to C manually. The function was originally called \texttt{bivnor}, and TWY only wrote a wrapper function.


\section*{See Also}

\texttt{pnorm}, \texttt{binormal}, \texttt{uninormal}.

\section*{Examples}

\begin{verbatim}
yvec <- c(-5, -1.96, 0, 1.96, 5)
ymat <- expand.grid(yvec, yvec)
cbind(ymat, pbinorm(ymat[, 1], ymat[, 2]))

## Not run: rhovec <- seq(-0.95, 0.95, by = 0.01)
plot(rhovec, pbinorm(0, 0, cov2 = rhovec), type = "l", col = "blue", las = 1)
abline(v = 0, h = 0.25, col = "gray", lty = "dashed")
## End(Not run)
\end{verbatim}
binormal  

**Bivariate normal distribution family function**

**Description**

Maximum likelihood estimation of the five parameters of a bivariate normal distribution.

**Usage**

```r
binormal(lmean1 = "identitylink", lmean2 = "identitylink", 
lsd1 = "loge", lsd2 = "loge", 
lrho = "rhobit", 
imean1 = NULL, imean2 = NULL, 
isd1 = NULL, isd2 = NULL, 
irho = NULL, imethod = 1, 
eq.mean = FALSE, eq.sd = FALSE, 
zero = 3:5)
```

**Arguments**

- `lmean1`, `lmean2`, `lsd1`, `lsd2`, `lrho`
  - Link functions applied to the means, standard deviations and rho parameters. See [Links](#) for more choices. Being positive quantities, a log link is the default for the standard deviations.
- `imean1`, `imean2`, `isd1`, `isd2`, `irho`, `imethod`, `zero`
  - See [CommonVGAMffArguments](#) for more information.
- `eq.mean`, `eq.sd`
  - Logical or formula. Constrains the means or the standard deviations to be equal. Only one of these arguments may be assigned a value.

**Details**

For the bivariate normal distribution, this fits a linear model (LM) to the means, and by default, the other parameters are intercept-only. The response should be a two-column matrix. The correlation parameter is rho, which lies between $-1$ and $1$ (thus the `rhobit` link is a reasonable choice). The fitted means are returned as the fitted values, which is in the form of a two-column matrix. Fisher scoring is implemented.

**Value**

An object of class "vglmff" (see [vglmff-class](#)). The object is used by modelling functions such as `vglm`, and `vgam`.

**Warning**

This function may be renamed to `normal2()` or something like that at a later date.
binormalcop

Note

If both equal means and equal standard deviations are desired then use something like constraints = list("(Intercept)"") and maybe zero = NULL etc.

Author(s)

T. W. Yee

See Also

uninormal, gaussianff, pnorm2, bistudentt.

Examples

```r
set.seed(123); nn <- 1000
bdata <- data.frame(x2 = runif(nn), x3 = runif(nn))
bdata <- transform(bdata, y1 = rnorm(nn, 1 + 2 * x2),
                   y2 = rnorm(nn, 3 + 4 * x2))
fit1 <- vglm(cbind(y1, y2) ~ x2,
             binormal(eq.sd = TRUE), data = bdata, trace = TRUE)
coef(fit1, matrix = TRUE)
constraints(fit1)
summary(fit1)
```

# Estimated P(Y1 <= y1, Y2 <= y2) under the fitted model
var1 <- loge(2 * predict(fit1)[, "loge(sd1)"], inverse = TRUE)
var2 <- loge(2 * predict(fit1)[, "loge(sd2)"], inverse = TRUE)
cov12 <- rhobit(predict(fit1)[, "rhobit(rho)"], inverse = TRUE)
head(with(bdata, pnorm2(y1, y2,
                      mean1 = predict(fit1)[, "mean1"],
                      mean2 = predict(fit1)[, "mean2"],
                      var1 = var1, var2 = var2, cov12 = cov12)))

---

**binormalcop**

*Gaussian Copula (Bivariate) Family Function*

**Description**

Estimate the correlation parameter of the (bivariate) Gaussian copula distribution by maximum likelihood estimation.

**Usage**

```r
binormalcop(lrho = "rhobit", irho = NULL, imethod = 1,
            parallel = FALSE, zero = NULL)
```
Arguments

lrho, irho, imethod
Details at CommonVGAMffArguments. See Links for more link function choices.

parallel, zero
Details at CommonVGAMffArguments. If parallel = TRUE then the constraint is applied to the intercept too.

Details

The cumulative distribution function is

\[ P(Y_1 \leq y_1, Y_2 \leq y_2) = \Phi_2(\Phi^{-1}(y_1), \Phi^{-1}(y_2); \rho) \]

for \(-1 < \rho < 1\), \(\Phi_2\) is the cumulative distribution function of a standard bivariate normal (see pbinorm), and \(\Phi\) is the cumulative distribution function of a standard univariate normal (see pnorm).

The support of the function is the interior of the unit square; however, values of 0 and/or 1 are not allowed. The marginal distributions are the standard uniform distributions. When \(\rho = 0\) the random variables are independent.

This VGAM family function can handle multiple responses, for example, a six-column matrix where the first 2 columns is the first out of three responses, the next 2 columns being the next response, etc.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

The response matrix must have a multiple of two-columns. Currently, the fitted value is a matrix with the same number of columns and values equal to 0.5. This is because each marginal distribution corresponds to a standard uniform distribution.

This VGAM family function is fragile; each response must be in the interior of the unit square. Setting crit = "coef" is sometimes a good idea because inaccuracies in pbinorm might mean unnecessary half-stepping will occur near the solution.

Author(s)

T. W. Yee

References


See Also

rbinormcop, pnorm, kendall.tau.
Binormcop

Examples

```r
nn <- 1000
ymat <- rbinormcop(n = nn, rho = rhibit(-0.9, inverse = TRUE))
bdata <- data.frame(y1 = ymat[, 1],
y2 = ymat[, 2],
y3 = ymat[, 1],
y4 = ymat[, 2],
x2 = runif(nn))
summary(bdata)
## Not run: plot(ymat, col = "blue")
fit1 <- vglm(cbind(y1, y2, y3, y4) ~ 1, # 2 responses, e.g., (y1,y2) is the first
fam = binormalcop,
crit = "coef", # Sometimes a good idea
data = bdata, trace = TRUE)
coef(fit1, matrix = TRUE)
Coef(fit1)
head(fitted(fit1))
summary(fit1)

# Another example; rho is a linear function of x2
bdata <- transform(bdata, rho = -0.5 + x2)
ymat <- rbinormcop(n = nn, rho = with(bdata, rho))
bdata <- transform(bdata, y5 = ymat[, 1],
y6 = ymat[, 2])
fit2 <- vgam(cbind(y5, y6) ~ s(x2), data = bdata,
binormalcop(lrho = "identitylink"), trace = TRUE)
## Not run: plot(fit2, lcol = "blue", scol = "orange", se = TRUE, las = 1)
```

Binormcop

Gaussian Copula (Bivariate) Distribution

Description

Density, distribution function, and random generation for the (one parameter) bivariate Gaussian copula distribution.

Usage

```r
dbinormcop(x1, x2, rho = 0, log = FALSE)
pbinormcop(q1, q2, rho = 0)
rbinormcop(n, rho = 0)
```

Arguments

- `x1`, `x2`, `q1`, `q2` vector of quantiles. The `x1` and `x2` should be in the interval `(0, 1)`. Ditto for `q1` and `q2`.
- `n` number of observations. Same as `rnorm`.

`rho` the correlation parameter. Should be in the interval \((-1, 1)\).

`log` Logical. If `TRUE` then the logarithm is returned.

**Details**

See `binormalcop`, the VGAM family functions for estimating the parameter by maximum likelihood estimation, for the formula of the cumulative distribution function and other details.

**Value**

`dbinormcop` gives the density, `pbinormcop` gives the distribution function, and `rbinormcop` generates random deviates (a two-column matrix).

**Note**

Yet to do: allow \(x_1\) and/or \(x_2\) to have values 1, and to allow any values for \(x_1\) and/or \(x_2\) to be outside the unit square.

**Author(s)**

T. W. Yee

**See Also**

`binormalcop`, `binormal`.

**Examples**

```r
## Not run: edge <- 0.01  # A small positive value
N <- 101; x <- seq(edge, 1.0 - edge, len = N); Rho <- 0.7
ox <- expand.grid(x, x)
zedd <- dbinormcop(ox[, 1], ox[, 2], rho = Rho, log = TRUE)
contour(x, x, matrix(zedd, N, N), col = "blue", labcex = 1.5)
zedd <- pbinormcop(ox[, 1], ox[, 2], rho = Rho)
contour(x, x, matrix(zedd, N, N), col = "blue", labcex = 1.5)

## End(Not run)
```

---

**Biplackett**  
*Plackett’s Bivariate Copula*

**Description**

Density, distribution function, and random generation for the (one parameter) bivariate Plackett copula.
Usage

dbiplackcop(x1, x2, oratio, log = FALSE)
pbiplackcop(q1, q2, oratio)
rbiplackcop(n, oratio)

Arguments

x1, x2, q1, q2 vector of quantiles.
n number of observations. Same as in runif.
oratio the positive odds ratio $\psi$.
log Logical. If TRUE then the logarithm is returned.

Details

See biplackettcop, the VGAM family functions for estimating the parameter by maximum likelihood estimation, for the formula of the cumulative distribution function and other details.

Value

dbiplackcop gives the density, pbiplackcop gives the distribution function, and rbiplackcop generates random deviates (a two-column matrix).

Author(s)

T. W. Yee

References


See Also

biplackettcop, bifrankcop.

Examples

```r
# Not run: N <- 101; oratio <- exp(1)
x <- seq(0.0, 1.0, len = N)
ox <- expand.grid(x, x)
zedd <- dbiplackcop(ox[, 1], ox[, 2], oratio = oratio)
contour(x, x, matrix(zedd, N, N), col = "blue")
zedd <- pbiplackcop(ox[, 1], ox[, 2], oratio = oratio)
contour(x, x, matrix(zedd, N, N), col = "blue")
plot(rr <- rbiplackcop(n = 3000, oratio = oratio))
par(mfrow = c(1, 2))
hist(rr[, 1])  # Should be uniform
hist(rr[, 2])  # Should be uniform
```
Plackett’s Bivariate Copula Family Function

Description

Estimate the association parameter of Plackett’s bivariate distribution (copula) by maximum likelihood estimation.

Usage

biplackettcop(link = "loge", ioratio = NULL, imethod = 1, nsimEIM = 200)

Arguments

link Link function applied to the (positive) odds ratio \( \psi \). See \texttt{Links} for more choices and information.

ioratio Numeric. Optional initial value for \( \psi \). If a convergence failure occurs try assigning a value or a different value.

imethod, nsimEIM See \texttt{CommonVGAMffArguments}.

Details

The defining equation is

\[
\psi = H \times (1 - y_1 - y_2 + H)/((y_1 - H) \times (y_2 - H))
\]

where \( P(Y_1 \leq y_1, Y_2 \leq y_2) = H \psi(y_1, y_2) \) is the cumulative distribution function. The density function is \( h_\psi(y_1, y_2) = \)

\[
\psi[1 + (\psi - 1)(y_1 + y_2 - 2y_1y_2)]/([1 + (\psi - 1)(y_1 + y_2)]^2 - 4\psi(\psi - 1)y_1y_2)^{3/2}
\]

for \( \psi > 0 \). Some writers call \( \psi \) the cross product ratio but it is called the odds ratio here. The support of the function is the unit square. The marginal distributions here are the standard uniform although it is commonly generalized to other distributions.

If \( \psi = 1 \) then \( h_\psi(y_1, y_2) = y_1y_2 \), i.e., independence. As the odds ratio tends to infinity one has \( y_1 = y_2 \). As the odds ratio tends to 0 one has \( y_2 = 1 - y_1 \).

Fisher scoring is implemented using \texttt{rbiplackcop}. Convergence is often quite slow.

Value

An object of class "vglmff" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm} and \texttt{vgam}. 
Note

The response must be a two-column matrix. Currently, the fitted value is a 2-column matrix with 0.5 values because the marginal distributions correspond to a standard uniform distribution.

Author(s)

T. W. Yee

References


See Also

*rbiplackcop, bifrankcop.*

Examples

```r
## Not run:
ymat <- rbiplackcop(n = 2000, oratio = exp(2))
plot(ymat, col = "blue")
fit <- vglm(ymat ~ 1, fam = biplackettcop, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
vcov(fit)
head(fitted(fit))
summary(fit)

## End(Not run)
```

Biplot of Constrained Regression Models

Description

biplot is a generic function applied to RR-VGLMs and QRR-VGLMs etc. These apply to rank-1 and rank-2 models of these only. For RR-VGLMs these plot the second latent variable scores against the first latent variable scores.

Methods

x  The object from which the latent variables are extracted and/or plotted.

Note

See lvplot which is very much related to biplots.
The Birnbaum-Saunders Distribution

Description

Density, distribution function, and random generation for the Birnbaum-Saunders distribution.

Usage

\[ \text{dbisa}(x, \text{scale} = 1, \text{shape}, \text{log} = \text{FALSE}) \]
\[ \text{pbisa}(q, \text{scale} = 1, \text{shape}, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE}) \]
\[ \text{qbisa}(p, \text{scale} = 1, \text{shape}, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE}) \]
\[ \text{rbisa}(n, \text{scale} = 1, \text{shape}) \]

Arguments

- \(x, q\) vector of quantiles.
- \(p\) vector of probabilities.
- \(n\) Same as in \text{runif}.
- \(\text{scale}, \text{shape}\) the (positive) scale and shape parameters.
- \(\text{log}\) Logical. If \text{TRUE} then the logarithm of the density is returned.
- \(\text{lower.tail}, \text{log.p}\) Same meaning as in \text{pnorm} or \text{qnorm}.

Details

The Birnbaum-Saunders distribution is a distribution which is used in survival analysis. See \text{bisa}, the \text{VGAM} family function for estimating the parameters, for more details.

Value

\text{dbisa} gives the density, \text{pbisa} gives the distribution function, and \text{qbisa} gives the quantile function, and \text{rbisa} generates random deviates.

Author(s)

T. W. Yee and Kai Huang

See Also

\text{bisa}.
Examples

```r
## Not run:
x <- seq(0, 6, len = 400)
plot(x, dbisa(x, shape = 1), type = "l", col = "blue",
     ylab = "Density", lwd = 2, ylim = c(0,1.3), lty = 3,
     main = "X ~ Birnbaum-Saunders(shape, scale = 1)"
lines(x, dbisa(x, shape = 2), col = "orange", lty = 2, lwd = 2)
lines(x, dbisa(x, shape = 0.5), col = "green", lty = 1, lwd = 2)
legend(x = 3, y = 0.9, legend = paste("shape = ",c(0.5, 1,2)),
      col = c("green","blue","orange"), lty = 1:3, lwd = 2)

shape <- 1; x <- seq(0, 4, len = 401)
plot(x, dbisa(x, shape = shape), type = "l", col = "blue", las = 1, ylab = "",
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Purple lines are the 10,20,...,90 percentiles", ylim = 0:1)
abline(h = 0, col = "blue", lty = 2)
lines(x, pbisa(x, shape = shape), col = "orange")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qbsa(probs, shape = shape)
lines(Q, dbisa(Q, shape = shape), col = "purple", lty = 3, type = "h")
pbsa(Q, shape = shape) - probs # Should be all zero
abline(h = probs, col = "purple", lty = 3)
lines(Q, pbisa(Q, shape = shape), col = "purple", lty = 3, type = "h")
```
## End(Not run)

---

**bisa**

*Birnbaum-Saunders Distribution Family Function*

**Description**

Estimates the shape and scale parameters of the Birnbaum-Saunders distribution by maximum likelihood estimation.

**Usage**

```r
bisa(ls = \"loge\", ish = \"loge\",
    iscale = 1, ishape = NULL, imethod = 1, zero = NULL, nowarning = FALSE)
```

**Arguments**

- `nowarning`: Logical. Suppress a warning? Ignored for VGAM 0.9-7 and higher.
- `lscale`, `ishape`: Parameter link functions applied to the shape and scale parameters ($a$ and $b$ below). See `Links` for more choices. A log link is the default for both because they are positive.
- `iscale`, `ishape`: Initial values for $a$ and $b$. A NULL means an initial value is chosen internally using `imethod`.

imethod  An integer with value 1 or 2 or 3 which specifies the initialization method. If failure to converge occurs try the other value, or else specify a value for ishape and/or iscale.

zero  An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The default is none of them. If used, choose one value from the set \{1,2\}.

Details

The (two-parameter) Birnbaum-Saunders distribution has a cumulative distribution function that can be written as

$$F(y; a, b) = \Phi[\xi(y/b)/a]$$

where \(\Phi(\cdot)\) is the cumulative distribution function of a standard normal (see `pnorm`), \(\xi(t) = \sqrt{t} - 1/\sqrt{t}, y > 0, a > 0\) is the shape parameter, \(b > 0\) is the scale parameter. The mean of \(Y\) (which is the fitted value) is \(b(1 + a^2/2)\), and the variance is \(a^2b^2(1 + \frac{5}{4}a^2)\). By default, \(\eta_1 = \log(a)\) and \(\eta_2 = \log(b)\) for this family function.

Note that \(a\) and \(b\) are orthogonal, i.e., the Fisher information matrix is diagonal. This family function implements Fisher scoring, and it is unnecessary to compute any integrals numerically.

Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Author(s)

T. W. Yee

References


See Also

`pbisa`, `inv.gaussianff`.
Examples

```r
bdata1 <- data.frame(x2 = runif(nn <- 1000))
bdata1 <- transform(bdata1, shape = exp(-0.5 + x2), scale = exp(1.5))
bdata1 <- transform(bdata1, y = rbisa(nn, scale, shape))
fit1 <- vglm(y ~ x2, bisa(zero = 1), data = bdata1, trace = TRUE)
coef(fit1, matrix = TRUE)
```

```r
## Not run:
bdata2 <- data.frame(shape = exp(-0.5), scale = exp(0.5))
bdata2 <- transform(bdata2, y = rbisa(nn, scale, shape))
fit <- vglm(y ~ 1, bisa, data = bdata2, trace = TRUE)
with(bdata2, hist(y, prob = TRUE, ylim = c(0, 0.5), col = "lightblue"))
coef(fit, matrix = TRUE)
with(bdata2, mean(y))
head(fitted(fit))
x <- with(bdata2, seq(0, max(y), len = 200))
lines(dbisa(x, Coef(fit)[1], Coef(fit)[2]) ~ x, data = bdata2,
     col = "orange", lwd = 2)
## End(Not run)
```

---

**Bistudentt**

*Bivariate Student-t distribution cumulative distribution function*

**Description**

Density for the bivariate Student-t distribution distribution.

**Usage**

```r
dbistudentt(x1, x2, df, rho = 0, log = FALSE)
```

**Arguments**

- `x1, x2` vector of quantiles.
- `df, rho` vector of degrees of freedom and correlation parameter. For `df`, a value `Inf` is currently not working.
- `log` Logical. If `log` = `TRUE` then the logarithm of the density is returned.

**Details**

One can think of this function as an extension of `dt` to two dimensions. See `bistudentt` for more information.

**Value**

`dbistudentt` gives the density.
References


See Also

`bistudentt`, `dt`.

Examples

```r
## Not run: N <- 101; x <- seq(-4, 4, len = N); Rho <- 0.7; mydf <- 10
ox <- expand.grid(x, x)
zedd <- dbistudentt(ox[, 1], ox[, 2], df = mydf, rho = Rho, log = TRUE)
contour(x, x, matrix(zedd, N, N), col = "blue", labcex = 1.5)
## End(Not run)
```

bistudentt  
**Bivariate Student-t Family Function**

Description

Estimate the degrees of freedom and correlation parameters of the (bivariate) Student-t distribution by maximum likelihood estimation.

Usage

```r
bistudentt(ldf = "loglog", lrho = "rhobit",
           idf = NULL, irho = NULL, imethod = 1,
             parallel = FALSE, zero = -1)
```

Arguments

- `ldf`, `lrho`, `idf`, `irho`, `imethod`
  - Details at `CommonVGAMffArguments`. See `Links` for more link function choices.
  - `parallel`, `zero` Details at `CommonVGAMffArguments`.

Details

The density function is

\[ f(y_1, y_2; \nu, \rho) = \frac{1}{2\pi\sqrt{1 - \rho^2}} (1 + y_1^2 + y_2^2 - 2\rho y_1 y_2) / (\nu(1 - \rho^2))^{(\nu + 2)/2} \]

for \(-1 < \rho < 1\), and real \(y_1\) and \(y_2\).

This VGAM family function can handle multiple responses, for example, a six-column matrix where the first 2 columns is the first out of three responses, the next 2 columns being the next response, etc.
Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

The working weight matrices have not been fully checked.

Note

The response matrix must have a multiple of two-columns. Currently, the fitted value is a matrix with the same number of columns and values equal to 0.0.

Author(s)

T. W. Yee, with help from Thibault Vatter.

References


See Also

dbistudentt, binormal, pt.

Examples

nn <- 1000
mydof <- loglog(1, inverse = TRUE)
ymat <- cbind(rt(nn, df = mydof), rt(nn, df = mydof))
bdata <- data.frame(y1 = ymat[, 1], y2 = ymat[, 2],
                    y3 = ymat[, 1], y4 = ymat[, 2], x2 = runif(nn))
summary(bdata)
## Not run: plot(ymat, col = "blue")
fit1 <- vglm(cbind(y1, y2, y3, y4) ~ 1, # 2 responses, e.g., (y1,y2) is the 1st
             fam = bistudentt, # crit = "coef", # Sometimes a good idea
data = bdata, trace = TRUE)

coef(fit1, matrix = TRUE)
Coef(fit1)
head(fitted(fit1))
summary(fit1)
Description

The body mass indexes and ages from an approximate random sample of 700 New Zealand adults.

Usage

data(bmi.nz)

Format

A data frame with 700 observations on the following 2 variables.

- **age**: a numeric vector; their age (years).
- **BMI**: a numeric vector; their body mass indexes, which is their weight divided by the square of their height (kg / m^2).

Details

They are a random sample from the Fletcher Challenge/Auckland Heart and Health survey conducted in the early 1990s.

There are some outliers in the data set.

A variable `gender` would be useful, and may be added later.

Source

Clinical Trials Research Unit, University of Auckland, New Zealand, [http://www.ctru.auckland.ac.nz](http://www.ctru.auckland.ac.nz).

References


Examples

```r
## Not run: with(bmi.nz, plot(age, BMI, col = "blue"))
fit <- vgam(BMI ~ s(age, df = c(2, 4, 2)), lms.yjn, data = bmi.nz, trace = TRUE)
qtplot(fit, pcol = "blue", tcol = "brown", lcol = "brown")
## End(Not run)
```
Description

Estimates the parameter of a Borel-Tanner distribution by maximum likelihood estimation.

Usage

```r
borel.tanner(Qsize = 1, link = "logit", imethod = 1)
```

Arguments

- `Qsize` A positive integer. It is called $Q$ below and is the initial queue size. The default value $Q = 1$ corresponds to the Borel distribution.
- `link` Link function for the parameter; see `links` for more choices and for general information.
- `imethod` See CommonVGAMArguments. Valid values are 1, 2, 3 or 4.

Details

The Borel-Tanner distribution (Tanner, 1953) describes the distribution of the total number of customers served before a queue vanishes given a single queue with random arrival times of customers (at a constant rate $r$ per unit time, and each customer taking a constant time $b$ to be served). Initially the queue has $Q$ people and the first one starts to be served. The two parameters appear in the density only in the form of the product $rb$, therefore we use $a = rb$, say, to denote the single parameter to be estimated. The density function is

$$f(y; a) = \frac{Q}{(y - Q)!} y^{y-Q-1} a^{y-Q} \exp(-ay)$$

where $y = Q, Q + 1, Q + 2, \ldots$. The case $Q = 1$ corresponds to the Borel distribution (Borel, 1942). For the $Q = 1$ case it is necessary for $0 < a < 1$ for the distribution to be proper. The Borel distribution is a basic Lagrangian distribution of the first kind. The Borel-Tanner distribution is an $Q$-fold convolution of the Borel distribution.

The mean is $Q/(1 - a)$ (returned as the fitted values) and the variance is $Qa/(1 - a)^3$. The distribution has a very long tail unless $a$ is small. Fisher scoring is implemented.

Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Author(s)

T. W. Yee
References


See Also

`rbort, poissonff, felix`.

Examples

```r
bdata <- data.frame(y = rbort(n = 200))
fit <- vglm(y ~ 1, borel.tanner, data = bdata, trace = TRUE, crit = "e")
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)
```

---

*Bort The Borel-Tanner Distribution*

Description

Density and random generation for the Borel-Tanner distribution.

Usage

```r
dbort(x, Qsize = 1, a = 0.5, log = FALSE)
rbort(n, Qsize = 1, a = 0.5)
```

Arguments

- `x` vector of quantiles.
- `n` number of observations. Must be a positive integer of length 1.
- `Qsize`, `a` See `borel.tanner`.
- `log` Logical. If `log = TRUE` then the logarithm of the density is returned.

Details

See `borel.tanner`, the *VGAM* family function for estimating the parameter, for the formula of the probability density function and other details.
Value

dbort gives the density, rbort generates random deviates.

Warning

Looping is used for rbort, therefore values of a close to 1 will result in long (or infinite!) computational times. The default value of a is subjective.

Author(s)

T. W. Yee

See Also

borel.tanner.

Examples

```r
## Not run:  qsiz[ <- 1; a <- 0.5; x <- qsiz[:(qsiz[+10)
plot(x, dbort(x, qsiz[, a), type = "h", las = 1, col = "blue",
    ylab = paste("fbort(qsiz[, qsiz[, "", a="", a, "])", log = "y",
    main = "Borel-Tanner density function")
## End(Not run)
```

Brat

**Inputting Data to fit a Bradley Terry Model**

Description

Takes in a square matrix of counts and outputs them in a form that is accessible to the brat and bratt family functions.

Usage

`Brat(mat, ties = 0 * mat, string = c(">", "="), whitespace = FALSE)`

Arguments

- `mat`: Matrix of counts, which is considered $M$ by $M$ in dimension when there are ties, and $M + 1$ by $M + 1$ when there are no ties. The rows are winners and the columns are losers, e.g., the 2-1 element is now many times Competitor 2 has beaten Competitor 1. The matrices are best labelled with the competitors’ names.
- `ties`: Matrix of counts. This should be the same dimension as `mat`. By default, there are no ties. The matrix must be symmetric, and the diagonal should contain NAs.
- `string`: Character. The matrices are labelled with the first value of the descriptor, e.g., "NZ > OZ" ‘means’ NZ beats Australia in rugby. Suggested alternatives include " beats " or " wins against ". The second value is used to handle ties.
whitespace Logical. If TRUE then a white space is added before and after string; it generally enhances readability. See CommonVGAMffArguments for some similar-type information.

Details

In the VGAM package it is necessary for each matrix to be represented as a single row of data by brat and bratt. Hence the non-diagonal elements of the $M + 1$ by $M + 1$ matrix are concatenated into $M(M + 1)$ values (no ties), while if there are ties, the non-diagonal elements of the $M$ by $M$ matrix are concatenated into $M(M - 1)$ values.

Value

A matrix with 1 row and either $M(M + 1)$ or $M(M - 1)$ columns.

Note

This is a data preprocessing function for brat and bratt.

Yet to do: merge InverseBrat into brat.

Author(s)

T. W. Yee

References


See Also

brat, bratt, InverseBrat.

Examples

```r
journal <- c("Biometrika", "Comm Statist", "JASA", "JRSS-B")
mat <- matrix(c(NA, 33, 320, 284, 730, NA, 813, 276, 498, 68, NA, 325, 221, 17, 142, NA), 4, 4)
dimnames(mat) <- list(winner = journal, loser = journal)
Brat(mat) # Less readable
Brat(mat, whitespace = TRUE) # More readable
vglm(Brat(mat, whitespace = TRUE) ~ 1, brat, trace = TRUE)
```
brat

Bradley Terry Model

Description
Fits a Bradley Terry model (intercept-only model) by maximum likelihood estimation.

Usage
brat(refgp = "last", refvalue = 1, ialpha = 1)

Arguments
refgp Integer whose value must be from the set \{1, \ldots, M + 1\}, where there are \(M + 1\) competitors. The default value indicates the last competitor is used—but don’t input a character string, in general.

refvalue Numeric. A positive value for the reference group.

ialpha Initial values for the \(\alpha\)s. These are recycled to the appropriate length.

Details
The Bradley Terry model involves \(M + 1\) competitors who either win or lose against each other (no draws/ties allowed in this implementation—see bratt if there are ties). The probability that Competitor \(i\) beats Competitor \(j\) is \(\alpha_i / (\alpha_i + \alpha_j)\), where all the \(\alpha\)s are positive. Loosely, the \(\alpha\)s can be thought of as the competitors’ ‘abilities’. For identifiability, one of the \(\alpha_i\) is set to a known value \(\text{refvalue}\), e.g., 1. By default, this function chooses the last competitor to have this reference value.

The data can be represented in the form of a \(M + 1\) by \(M + 1\) matrix of counts, where winners are the rows and losers are the columns. However, this is not the way the data should be inputted (see below).

Excluding the reference value/group, this function chooses \(\log(\alpha_j)\) as the \(M\) linear predictors. The log link ensures that the \(\alpha\)s are positive.

The Bradley Terry model can be fitted by logistic regression, but this approach is not taken here. The Bradley Terry model can be fitted with covariates, e.g., a home advantage variable, but unfortunately, this lies outside the VGLM theoretical framework and therefore cannot be handled with this code.

Value
An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm.

Warning
Presently, the residuals are wrong, and the prior weights are not handled correctly. Ideally, the total number of counts should be the prior weights, after the response has been converted to proportions. This would make it similar to family functions such as multinomial and binomialff.
**Note**

The function `Brat` is useful for coercing a $M + 1$ by $M + 1$ matrix of counts into a one-row matrix suitable for `brat`. Diagonal elements are skipped, and the usual S order of `c(a.matrix)` of elements is used. There should be no missing values apart from the diagonal elements of the square matrix. The matrix should have winners as the rows, and losers as the columns. In general, the response should be a 1-row matrix with $M(M + 1)$ columns.

Only an intercept model is recommended with `brat`. It doesn’t make sense really to include covariates because of the limited VGLM framework.

Notationally, note that the **VGAM** family function `brat` has $M + 1$ contestants, while `bratt` has $M$ contestants.

**Author(s)**

T. W. Yee

**References**


The **BradleyTerry2** package has more comprehensive capabilities than this function.

**See Also**

`bratt`, `Brat`, `multinomial`, `binomialff`.

**Examples**

```r
citation statistics being cited is a 'win'; citing is a 'loss'
journal <- c("Biometrika", "Comm.Statis", "JASA", "JRSS-B")
mat <- matrix(c( NA, 33, 320, 284,
                 730, NA, 813, 276,
                 498, 68, NA, 325,
                 221, 17, 142, NA), 4, 4)
dimnames(mat) <- list(winner = journal, loser = journal)
fit <- vglm(Brat(mat) ~ 1, brat(refgp = 1), trace = TRUE)
fit <- vglm(Brat(mat) ~ 1, brat(refgp = 1), trace = TRUE, crit = "coef")
summary(fit)
c(0, coef(fit))  # Log-abilities (in order of "journal")
c(1, Coef(fit))  # Abilities (in order of "journal")
fitted(fit)      # Probabilities of winning in awkward form
(check <- inverseBrat(fitted(fit)))  # Probabilities of winning
check + t(check)  # Should be 1's in the off-diagonals
```
**bratt**

**Bradley Terry Model With Ties**

### Description

Fits a Bradley Terry model with ties (intercept-only model) by maximum likelihood estimation.

### Usage

```r
bratt(refgp = "last", refvalue = 1, ialpha = 1, i0 = 0.01)
```

### Arguments

- `refgp` : Integer whose value must be from the set \{1, \ldots, M\}, where there are \(M\) competitors. The default value indicates the last competitor is used—but don’t input a character string, in general.
- `refvalue` : Numeric. A positive value for the reference group.
- `ialpha` : Initial values for the \(\alpha_s\). These are recycled to the appropriate length.
- `i0` : Initial value for \(\alpha_0\). If convergence fails, try another positive value.

### Details

There are several models that extend the ordinary Bradley Terry model to handle ties. This family function implements one of these models. It involves \(M\) competitors who either win or lose or tie against each other. (If there are no draws/ties then use `brat`). The probability that Competitor \(i\) beats Competitor \(j\) is \(\alpha_i/(\alpha_i + \alpha_j + \alpha_0)\), where all the \(\alpha_s\) are positive. The probability that Competitor \(i\) ties with Competitor \(j\) is \(\alpha_0/(\alpha_i + \alpha_j + \alpha_0)\). Loosely, the \(\alpha_s\) can be thought of as the competitors’ ‘abilities’, and \(\alpha_0\) is an added parameter to model ties. For identifiability, one of the \(\alpha_i\) is set to a known value `refvalue`, e.g., 1. By default, this function chooses the last competitor to have this reference value. The data can be represented in the form of a \(M\) by \(M\) matrix of counts, where winners are the rows and losers are the columns. However, this is not the way the data should be inputted (see below).

Excluding the reference value/group, this function chooses \(\log(\alpha_j)\) as the first \(M - 1\) linear predictors. The log link ensures that the \(\alpha_s\) are positive. The last linear predictor is \(\log(\alpha_0)\).

The Bradley Terry model can be fitted with covariates, e.g., a home advantage variable, but unfortunately, this lies outside the VGLM theoretical framework and therefore cannot be handled with this code.

### Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`. 
Note

The function `brat` is useful for coercing a $M$ by $M$ matrix of counts into a one-row matrix suitable for `bratt`. Diagonal elements are skipped, and the usual S order of `c(a.matrix)` of elements is used. There should be no missing values apart from the diagonal elements of the square matrix. The matrix should have winners as the rows, and losers as the columns. In general, the response should be a matrix with $M(M-1)$ columns.

Also, a symmetric matrix of ties should be passed into `brat`. The diagonal of this matrix should be all NAs.

Only an intercept model is recommended with `bratt`. It doesn’t make sense really to include covariates because of the limited VGLM framework.

Notationally, note that the `VGAM` family function `brat` has $M + 1$ contestants, while `bratt` has $M$ contestants.

Author(s)

T. W. Yee

References


See Also

`brat`, `Brat`, `binomialff`.

Examples

```r
# citation statistics: being cited is a 'win'; citing is a 'loss'
journal <- c("Biometrika", "Comm.Statist", "JASA", "JRSS-B")
mat <- matrix(c(NA, 33, 320, 284,
                730, NA, 813, 276,
                498, 68, NA, 325,
                221, 17, 142, NA), 4, 4)
dimnames(mat) <- list(winner = journal, loser = journal)

# Add some ties. This is fictitional data.
ties <- 5 + 0 * mat
ties[2, 1] <- ties[1, 2] <- 9

# Now fit the model
fit <- vglm(Brat(mat, ties) ~ 1, bratt(refgp = 1), trace = TRUE)
fit <- vglm(Brat(mat, ties) ~ 1, bratt(refgp = 1), trace = TRUE, crit = "coef")

summary(fit)
c(0, coef(fit))  # Log-abilities (in order of "journal"); last is log(alpha0)
c(1, Coef(fit))  # Abilities (in order of "journal"); last is alpha0
fit@misc$alpha  # alpha_1,...,alpha_M
```
calibrate

Model Calibrations

Description

calibrate is a generic function used to produce calibrations from various model fitting functions. The function invokes particular ‘methods’ which depend on the ‘class’ of the first argument.

Usage

calibrate(object, ...)

Arguments

object An object for which a calibration is desired.
...
Additional arguments affecting the calibration produced. Usually the most important argument in ... is newdata which, for calibrate, contains new response data, Y, say.

Details

Given a regression model with explanatory variables X and response Y, calibration involves estimating X from Y using the regression model. It can be loosely thought of as the opposite of predict (which takes an X and returns a Y.)

Value

In general, given a new response Y, the explanatory variables X are returned. However, for constrained ordination models such as CQO and CAO models, it is usually not possible to return X, so the latent variables are returned instead (they are linear combinations of the X). See the specific calibrate methods functions to see what they return.

Note

This function was not called predictx because of the inability of constrained ordination models to return X; they can only return the latent variable values (site scores) instead.

Author(s)

T. W. Yee
See Also

predict.calibrate.qrrvglm.

Examples

```r
## Not run:
hs育人[,1:6] <- scale(hspider[,1:6])  # Standardized environmental vars
set.seed(123)
pl <- cao(cbind(Pardlugu, Pardmont, Pardnigr, Pardpull, Zoraspin) ~
  WaterCon + BareSand + FallTwig +
  CoveMoss + CoveHerb + RefLlux,
  family = poissonff, data = hspider, Rank = 1,
  df1.nl = c(Zoraspin = 2, 1.9),
  Bestof = 3, Crowlpositive = TRUE)

siteNos <- 1:2  # Calibrate these sites
cri <- calibrate(pl, new = data.frame(depvar(pl)[siteNos, ]), trace = TRUE)

# Graphically compare the actual site scores with their calibrated values
persp(pl, main = "Solid=actual, dashed=calibrated site scores",
     label = TRUE, col = "blue", las = 1)
# Actual site scores:
abline(v = latvar(pl)[siteNos], lty = 1, col = 1:length(siteNos))
abline(v = cri, lty = 2, col = 1:length(siteNos))  # Calibrated values
```

### calibration-methods

**Calibration for Constrained Regression Models**

**Description**

calibrate is a generic function applied to QRR-VGLMs and RR-VGAMs etc.

**Methods**

- **object** The object from which the calibration is performed.

### calibrate.qrrvglm

**Calibration for CQO and CAO models**

**Description**

Performs maximum likelihood calibration for constrained and unconstrained quadratic and additive ordination models (CQO and CAO models are better known as QRR-VGLMs and RR-VGAMs respectively).
Usage

calibrate.qrrvglm(object, newdata = NULL,
    type = c("latvar", "predictors", "response", "vcov", "all3or4"),
    initial vals = NULL, ...)

Arguments

object The fitted CQO/CAO model.

newdata A data frame with new response data (usually new species data). The default is to use the original data used to fit the model; however, the calibration may take a long time to compute because the computations are expensive.

type What type of result is to be returned. The first are the calibrated latent variables or site scores. This must be computed always. The "predictors" are the linear/quadratic or additive predictors evaluated at the calibrated latent variables or site scores. The "response" are the fitted means evaluated at the calibrated latent variables or site scores. The "vcov" are the estimated variance-covariance matrices of the calibrated latent variables or site scores. The "all3or4" is for all of them, i.e., all types. For CAO models, "vcov" is unavailable, so all 3 are returned. For CQO models, "vcov" is available, so all 4 are returned.

initial vals Initial values for the search. For rank-1 models, this should be a vector of length \( nrow(\text{newdata}) \), and for rank 2 models this should be a two column matrix with the number of rows equalling the number of rows in \( \text{newdata} \). The default is a grid defined by arguments in \text{calibrate.qrrvglm.control}.

... Arguments that are fed into \text{calibrate.qrrvglm.control}.

Details

Given a fitted regression CQO/CAO model, maximum likelihood calibration is theoretically easy and elegant. However, the method assumes that all species are independent, which is not really true in practice. More details and references are given in Yee (2012).

The function \text{optim} is used to search for the maximum likelihood solution. Good initial values are needed, and \text{calibrate.qrrvglm.control} allows the user some control over the choice of these.

Value

The argument type determines what is returned. If type = "all3or4" then all the type values are returned in a list, with the following components. Each component has length \( nrow(\text{newdata}) \).

\begin{itemize}
  \item \text{latvar} Calibrated latent variables or site scores.
  \item \text{predictors} linear/quadratic or additive predictors. For example, for Poisson families, this will be on a log scale, and for binomial families, this will be on a logit scale.
  \item \text{response} Fitted values of the response, evaluated at the calibrated latent variables or site scores.
  \item \text{vcov} Estimated variance-covariance matrix of the calibrated latent variables or site scores. Actually, these are stored in an array whose last dimension is \( nrow(\text{newdata}) \).
\end{itemize}
Warning

This function is computationally expensive. Setting `trace = TRUE` to get a running log is a good idea.

Note

Despite the name of this function, CAO models are handled as well.

Author(s)

T. W. Yee

References


See Also

calibrate.qrrvglm.control, calibrate, cqa, cao.

Examples

```r
## Not run:
hsppr[,1:6] <- scale(hsppr[, 1:6])  # Standardize the environmental variables
set.seed(123)
p1 <- cqa(cbind(Pardlugu, Pardmont, Pardnigr, Pardpull, Zoraspin) ~
          WaterCon + BareSand + FallTwig +
          CoveMoss + CoveHerb + ReflLux,
          family = poissonff, data = hsppr, Rank = 1,
          I.toler = TRUE, Crowlpositive = TRUE)

siteNos <- 3:4  # Calibrate these sites
cp1 <- calibrate(p1, new = data.frame(depvar(p1)[siteNos, ]), trace = TRUE)
## End(Not run)

## Not run:
# Graphically compare the actual site scores with their calibrated values
persp(p1, main = "Site scores: solid=actual, dashed=calibrated",
      label = TRUE, col = "blue", las = 1)
# Actual site scores:
abline(v = latvar(p1)[siteNos], lty = 1, col = 1:length(siteNos))
abline(v = cp1, lty = 2, col = 1:length(siteNos))  # Calibrated values
## End(Not run)
```
Control function for CQO/CAO calibration

**Description**

Algorithmic constants and parameters for running `calibrate.qrrvglm` are set using this function.

**Usage**

```r
calibrate.qrrvglm.control(object, trace = FALSE, Method.optim = "BFGS",
                          gridSize = if (Rank == 1) 9 else 5, varI.latvar = FALSE, ...)
```

**Arguments**

- `object` The fitted CQO/CAO model. The user should ignore this argument.
- `trace` Logical indicating if output should be produced for each iteration. It is a good idea to set this argument to be `TRUE` since the computations are expensive.
- `gridSize` Numeric, recycled to length `Rank`. Controls the resolution of the grid used for initial values. For each latent variable, an equally spaced grid of length `gridSize` is cast from the smallest site score to the largest site score. Then the likelihood function is evaluated on the grid, and the best fit is chosen as the initial value. Thus increasing the value of `gridSize` increases the chance of obtaining the global solution, however, the computing time increases proportionately.
- `varI.latvar` Logical. For CQO objects only, this argument is fed into `coef.qrrvglm`.
- `...` Avoids an error message for extraneous arguments.

**Details**

Most CQO/CAO users will only need to make use of `trace` and `gridSize`. These arguments should be used inside their call to `calibrate.qrrvglm`, not this function directly.

**Value**

A list which with the following components.

- `trace` Numeric (even though the input can be logical).
- `gridSize` Positive integer.
- `varI.latvar` Logical.

**Note**

Despite the name of this function, CAO models are handled as well.
Description

A constrained additive ordination (CAO) model is fitted using the reduced-rank vector generalized additive model (RR-VGAM) framework.
Usage

cao(formula, family, data = list(),
    weights = NULL, subset = NULL, na.action = na.fail,
    etastart = NULL, mustart = NULL, coefstart = NULL,
    control = cao.control(...), offset = NULL,
    method = "cao.fit", model = FALSE, x.arg = TRUE, y.arg = TRUE,
    contrasts = NULL, constraints = NULL,
    extra = NULL, qr.arg = FALSE, smart = TRUE, ...)

Arguments

  formula  a symbolic description of the model to be fit. The RHS of the formula is used to construct the latent variables, upon which the smooths are applied. All the variables in the formula are used for the construction of latent variables except for those specified by the argument norrr, which is itself a formula. The LHS of the formula contains the response variables, which should be a matrix with each column being a response (species).

  family    a function of class "vglmmf" (see vglmff-class) describing what statistical model is to be fitted. This is called a "VGAM family function". See commonvgamffarguments for general information about many types of arguments found in this type of function. See cao for a list of those presently implemented.

  data      an optional data frame containing the variables in the model. By default the variables are taken from environment(formula), typically the environment from which cao is called.

  weights   an optional vector or matrix of (prior) weights to be used in the fitting process. For cao, this argument currently should not be used.

  subset    an optional logical vector specifying a subset of observations to be used in the fitting process.

  na.action a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The “factory-fresh” default is na.omit.

  etastart  starting values for the linear predictors. It is a $M$-column matrix. If $M = 1$ then it may be a vector. For cao, this argument currently should not be used.

  mustart   starting values for the fitted values. It can be a vector or a matrix. Some family functions do not make use of this argument. For cao, this argument currently should not be used.

  coefstart starting values for the coefficient vector. For cao, this argument currently should not be used.

  control   a list of parameters for controlling the fitting process. See cao.control for details.

  offset    a vector or $M$-column matrix of offset values. These are a priori known and are added to the linear predictors during fitting. For cao, this argument currently should not be used.

  method    the method to be used in fitting the model. The default (and presently only) method cao.fit uses iteratively reweighted least squares (IRLS) within FORTRAN code called from optim.
model  a logical value indicating whether the model frame should be assigned in the model slot.

x.arg, y.arg  logical values indicating whether the model matrix and response vector/matrix used in the fitting process should be assigned in the x and y slots. Note the model matrix is the linear model (LM) matrix.

contrasts  an optional list. See the contrasts.arg of model.matrix.default.

constraints  an optional list of constraint matrices. For cao, this argument currently should not be used. The components of the list must be named with the term it corresponds to (and it must match in character format). Each constraint matrix must have $M$ rows, and be of full-column rank. By default, constraint matrices are the $M$ by $M$ identity matrix unless arguments in the family function itself override these values. If constraints is used it must contain all the terms; an incomplete list is not accepted.

extra  an optional list with any extra information that might be needed by the family function. For cao, this argument currently should not be used.

qr.arg  For cao, this argument currently should not be used.

smart  logical value indicating whether smart prediction (smartpred) will be used.

...  further arguments passed into cao.control.

Details

The arguments of cao are a mixture of those from vgam and cqo, but with some extras in cao.control. Currently, not all of the arguments work properly.

CAO can be loosely be thought of as the result of fitting generalized additive models (GAMs) to several responses (e.g., species) against a very small number of latent variables. Each latent variable is a linear combination of the explanatory variables; the coefficients $C$ (called $C$ below) are called constrained coefficients or canonical coefficients, and are interpreted as weights or loadings. The $C$ are estimated by maximum likelihood estimation. It is often a good idea to apply scale to each explanatory variable first.

For each response (e.g., species), each latent variable is smoothed by a cubic smoothing spline, thus CAO is data-driven. If each smooth were a quadratic then CAO would simplify to constrained quadratic ordination (CQO; formerly called canonical Gaussian ordination or CGO). If each smooth were linear then CAO would simplify to constrained linear ordination (CLO). CLO can theoretically be fitted with cao by specifying df1.nl=0, however it is more efficient to use rrvglm.

Currently, only Rank=1 is implemented, and only noRRR = ~1 models are handled.

With binomial data, the default formula is

$$\text{logit}(P[Y_s = 1]) = \eta_s = f_s(\nu), \quad s = 1, 2, \ldots, S$$

where $x_2$ is a vector of environmental variables, and $\nu = C^T x_2$ is a $R$-vector of latent variables. The $\eta_s$ is an additive predictor for species $s$, and it models the probabilities of presence as an additive model on the logit scale. The matrix $C$ is estimated from the data, as well as the smooth functions $f_s$. The argument noRRR = ~1 specifies that the vector $x_1$, defined for RR-VGLMs and QRR-VGLMs, is simply a 1 for an intercept. Here, the intercept in the model is absorbed into the functions. A cloglog link may be preferable over a logit link.
With Poisson count data, the formula is
\[
\log(E[Y_s]) = \eta_s = f_s(\nu)
\]
which models the mean response as an additive models on the log scale.

The fitted latent variables (site scores) are scaled to have unit variance. The concept of a tolerance is undefined for CAO models, but the optimums and maximums are defined. The generic functions \texttt{Max} and \texttt{Opt} should work for CAO objects, but note that if the maximum occurs at the boundary then \texttt{Max} will return a \texttt{NA}. Inference for CAO models is currently undeveloped.

**Value**

An object of class "cao" (this may change to "rrvgam" in the future). Several generic functions can be applied to the object, e.g., \texttt{coef}, \texttt{concoef}, \texttt{lvplot}, \texttt{summary}.

**Warning**

CAO is very costly to compute. With version 0.7-8 it took 28 minutes on a fast machine. I hope to look at ways of speeding things up in the future.

Use \texttt{set.seed} just prior to calling \texttt{cao()} to make your results reproducible. The reason for this is finding the optimal CAO model presents a difficult optimization problem, partly because the log-likelihood function contains many local solutions. To obtain the (global) solution the user is advised to try \textit{many} initial values. This can be done by setting \texttt{bestof} some appropriate value (see \texttt{cao.control}). Trying many initial values becomes progressively more important as the nonlinear degrees of freedom of the smooths increase.

Currently the dispersion parameter for a \texttt{gaussian} CAO model is estimated slightly differently and may be slightly biassed downwards (usually a little too small).

**Note**

CAO models are computationally expensive, therefore setting \texttt{trace = TRUE} is a good idea, as well as running it on a simple random sample of the data set instead.

Sometimes the IRLS algorithm does not converge within the FORTRAN code. This results in warnings being issued. In particular, if an error code of 3 is issued, then this indicates the IRLS algorithm has not converged. One possible remedy is to increase or decrease the nonlinear degrees of freedom so that the curves become more or less flexible, respectively.

**Author(s)**

T. W. Yee

**References**


**See Also**

\texttt{cao.control}, \texttt{Coef.cao}, \texttt{cqo}, \texttt{latvar}, \texttt{Opt}, \texttt{Max}, \texttt{persp.cao}, \texttt{poissonff}, \texttt{binomialff}, \texttt{negbinomial}, \texttt{gamma2}, \texttt{gaussianff}, \texttt{set.seed}, \texttt{gam}, \texttt{trap0}.  

Examples

```r
## Not run:
hsider[, 1:6] <- scale(hsider[, 1:6]) # Standardized environmental vars
set.seed(149) # For reproducible results
ap1 <- cao(cbind(Pardlug, Pardmont, Pardnigr, Pardpull) ~
  WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + RefiLux,
  family = poissonff, data = hsider, Rank = 1,
  df1.nl = c(Pardpull = 2.7, 2.5),
  Bestof = 7, Crowlpositive = FALSE)
sort(deviance(ap1, history = TRUE)) # A history of all the iterations

Coeff(ap1)
concoeff(ap1)

par(mfrow = c(2, 2))
plot(ap1) # All the curves are unimodal; some quite symmetric

par(mfrow = c(1, 1), las = 1)
index <- 1:ncol(depvar(ap1))
lvplot(ap1, lcol = index, pcol = index, y = TRUE)

trplot(ap1, label = TRUE, col = index)
abline(a = 0, b = 1, lty = 2)

trplot(ap1, label = TRUE, col = "blue", log = "xy", which.sp = c(1, 3))
abline(a = 0, b = 1, lty = 2)

persp(ap1, col = index, lwd = 2, label = TRUE)
abline(v = Opt(ap1), lty = 2, col = index)
abline(h = Max(ap1), lty = 2, col = index)

## End(Not run)
```

cao.control

Control Function for RR-VGAMs (CAO)

Description

Algorithmic constants and parameters for a constrained additive ordination (CAO), by fitting a reduced-rank vector generalized additive model (RR-VGAM), are set using this function. This is the control function for cao.

Usage

```r
cao.control(Rank = 1, all.knots = FALSE, criterion = "deviance", Cinit = NULL,
  Crowlpositive = TRUE, epsilon = 1.0e-05, Etamat.colmax = 10,
  GradientFunction = FALSE, iKvector = 0.1, iShape = 0.1,
  norrrr = ~ 1, Norrrr = NA,
  SmallNo = 5.0e-13, Use.Init.Poisson.Q0 = TRUE,
```
Bestof = if (length(Cinit)) 1 else 10, maxitl = 10,
imethod = 1, bf. epsilon = 1.0e-7, bf. maxit = 10,
Maxit. optim = 250, optim. maxit = 20, sd. sitescores = 1.0,
 sd. Cinit = 0.02, suppress. warnings = TRUE,
trace = TRUE, df1.nl = 2.5, df2.nl = 2.5,
spar1 = 0, spar2 = 0, ...)

Arguments

Rank
The numerical rank $R$ of the model, i.e., the number of latent variables. Currently only Rank = 1 is implemented.

all.knots
Logical indicating if all distinct points of the smoothing variables are to be used as knots. Assigning the value FALSE means fewer knots are chosen when the number of distinct points is large, meaning less computational expense. See vgam.control for details.

criterion
Convergence criterion. Currently, only one is supported: the deviance is minimized.

cinit
Optional initial C matrix which may speed up convergence.

crow1positive
Logical vector of length Rank (recycled if necessary): are the elements of the first row of C positive? For example, if Rank is 4, then specifying crow1positive = c(FALSE, TRUE) will force C[1,1] and C[1,3] to be negative, and C[1,2] and C[1,4] to be positive.

epsilon
Positive numeric. Used to test for convergence for GLMs fitted in FORTRAN. Larger values mean a loosening of the convergence criterion.

etamat.colmax
Positive integer, no smaller than Rank. Controls the amount of memory used by Init.Poisson.QO(). It is the maximum number of columns allowed for the pseudo-response and its weights. In general, the larger the value, the better the initial value. Used only if Use.Init.Poisson.QO = TRUE.

GradientFunction
Logical. Whether optim's argument gr is used or not, i.e., to compute gradient values. Used only if fastalgorithm is TRUE. Currently, this argument must be set to FALSE.

iKvector, iShape
See qrrvglm.control.

noRRR
Formula giving terms that are not to be included in the reduced-rank regression (or formation of the latent variables). The default is to omit the intercept term from the latent variables. Currently, only noRRR = ~ 1 is implemented.

Norr
Defunct. Please use noRRR. Use of Norrr will become an error soon.

SmallNo
Positive numeric between .Machine$double.eps and 0.0001. Used to avoid under- or over-flow in the IRLS algorithm.

Use.Init.Poisson.QO
Logical. If TRUE then the function Init.Poisson.QO is used to obtain initial values for the canonical coefficients C. If FALSE then random numbers are used instead.

Bestof
Integer. The best of Bestof models fitted is returned. This argument helps guard against local solutions by (hopefully) finding the global solution from many fits.
The argument works only when the function generates its own initial value for \( C \), i.e., when \( C \) are not passed in as initial values. The default is only a convenient minimal number and users are urged to increase this value.

**maxitl**
Positive integer. Maximum number of Newton-Raphson/Fisher-scoring/local-scoring iterations allowed.

**imethod**
See `qrrvglm.control`.

**bf.epsilon**
Positive numeric. Tolerance used by the modified vector backfitting algorithm for testing convergence.

**bf.maxit**
Positive integer. Number of backfitting iterations allowed in the compiled code.

**Maxit.optim**
Positive integer. Number of iterations given to the function `optim` at each of the `optim.maxit` iterations.

**optim.maxit**
Positive integer. Number of times `optim` is invoked.

**sd.sitescores**
Numeric. Standard deviation of the initial values of the site scores, which are generated from a normal distribution. Used when `Use.Init.Poisson.QO` is FALSE.

**sd.Cinit**
Standard deviation of the initial values for the elements of \( C \). These are normally distributed with mean zero. This argument is used only if `Use.Init.Poisson.QO` = FALSE.

**suppress.warnings**
Logical. Suppress warnings?

**trace**
Logical indicating if output should be produced for each iteration. Having the value `TRUE` is a good idea for large data sets.

**df1.nl, df2.nl**
Numeric and non-negative, recycled to length \( S \). Nonlinear degrees of freedom for smooths of the first and second latent variables. A value of 0 means the smooth is linear. Roughly, a value between 1.0 and 2.0 often has the approximate flexibility of a quadratic. The user should not assign too large a value to this argument, e.g., the value 4.0 is probably too high. The argument `df1.nl` is ignored if `spar1` is assigned a positive value or values. Ditto for `df2.nl`.

**spar1, spar2**
Numeric and non-negative, recycled to length \( S \). Smoothing parameters of the smooths of the first and second latent variables. The larger the value, the more smooth (less wiggly) the fitted curves. These arguments are an alternative to specifying `df1.nl` and `df2.nl`. A value 0 (the default) for `spar1` means that `df1.nl` is used. Ditto for `spar2`. The values are on a scaled version of the latent variables. See Green and Silverman (1994) for more information.

... Ignored at present.

**Details**
Many of these arguments are identical to `qrrvglm.control`. Here, \( R \) is the Rank, \( M \) is the number of additive predictors, and \( S \) is the number of responses (species). Thus \( M = S \) for binomial and Poisson responses, and \( M = 2S \) for the negative binomial and 2-parameter gamma distributions.

Allowing the smooths too much flexibility means the CAO optimization problem becomes more difficult to solve. This is because the number of local solutions increases as the nonlinearity of the smooths increases. In situations of high nonlinearity, many initial values should be used, so that `Bestof` should be assigned a larger value. In general, there should be a reasonable value of `df1.nl` somewhere between 0 and about 3 for most data sets.
Value

A list with the components corresponding to its arguments, after some basic error checking.

Note

The argument df1.nl can be inputted in the format c(spp1 = 2, spp2 = 3, 2.5), say, meaning the default value is 2.5, but two species have alternative values.

If spar1 = 0 and df1.nl = 0 then this represents fitting linear functions (CLO). Currently, this is handled in the awkward manner of setting df1.nl to be a small positive value, so that the smooth is almost linear but not quite. A proper fix to this special case should be done in the short future.

Author(s)

T. W. Yee

References


See Also

cao.

Examples

```r
## Not run:
hsphrase[,1:6] <- scale(hsphrase[,1:6])  # Standardized environmental vars
set.seed(123)
ap1 <- cao(cbind(Pardlugu, Pardmont, Pardnigr, Pardpull, Zoraspin) ~
  WaterCon + BareSand + FallTwig +
  CoveMoss + CoveHerb + RefLlux,
  family = poissonff, data = hspider,
  df1.nl = c(Zoraspin = 2.3, 2.1),
  Bestof = 10, Crowglyph = FALSE)
sort(deviance(ap1, history = TRUE))  # A history of all the iterations

Coef(ap1)

par(mfrow = c(2, 3))  # All or most of the curves are unimodal; some are
plot(ap1, lcol = "blue")  # quite symmetric. Hence a CQO model should be ok
par(mfrow = c(1, 1), las = 1)
index <- 1:ncol(depvar(ap1))  # lplot is jagged because only 28 sites
lplot(ap1, lcol = index, pcol = index, y = TRUE)

trplot(ap1, label = TRUE, col = index)
abline(a = 0, b = 1, lty = 2)
```
Card

Cardioid Distribution

Description
Density, distribution function, quantile function and random generation for the cardioid distribution.

Usage

dcard(x, mu, rho, log = FALSE)
pcard(q, mu, rho, lower.tail = TRUE, log.p = FALSE)
qcard(p, mu, rho, tolerance = 1e-07, maxits = 500,
    lower.tail = TRUE, log.p = FALSE)
rcard(n, mu, rho, ...)  

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. Same as in runif.
mu, rho See cardioid for more information.
tolerance, maxits, ... The first two are control parameters for the algorithm used to solve for the roots of a nonlinear system of equations; tolerance controls for the accuracy and maxits is the maximum number of iterations. rcard calls qcard so the ... can be used to vary the two arguments.

log Logical. If log = TRUE then the logarithm of the density is returned.
lower.tail, log.p Same meaning as in pnorm or qnorm.

Details
See cardioid, the VGAM family function for estimating the two parameters by maximum likelihood estimation, for the formula of the probability density function and other details.

Value
dcard gives the density, pcard gives the distribution function, qcard gives the quantile function, and rcard generates random deviates.

Note
Convergence problems might occur with rcard.
cardioid

Author(s)

Thomas W. Yee and Kai Huang

See Also

cardioid.

Examples

```r
## Not run
mu <- 4; rho <- 0.4; x <- seq(0, 2*pi, len = 501)
plot(x, dcard(x, mu, rho), type = "l", las = 1, ylim = c(0, 1), col = "blue",
     ylab = paste("[dp]card(mu="", mu, ", rho="", rho, ")"),
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Purple lines are the 10,20,...,90 percentiles")
lines(x, pcard(x, mu, rho), col = "orange")

probs <- seq(0.1, 0.9, by = 0.1)
Q <- qcard(probs, mu, rho)
lines(Q, dcard(Q, mu, rho), col = "purple", lty = 3, type = "h")
lines(Q, pcard(Q, mu, rho), col = "purple", lty = 3, type = "h")
abline(h = c(0, probs, 1), v = c(0, 2*pi), col = "purple", lty = 3)
max(abs(pcard(Q, mu, rho) - probs)) # Should be 0

## End(Not run)
```

cardioid  

Cardioid Distribution Family Function

Description

Estimates the two parameters of the cardioid distribution by maximum likelihood estimation.

Usage

```r
cardioid(lmu = extlogit(min = 0, max = 2*pi),
    lrho = extlogit(min = -0.5, max = 0.5),
    imu = NULL, irho = 0.3, nsimEIM = 100, zero = NULL)
```

Arguments

- `lmu`, `lrho`  Parameter link functions applied to the $\mu$ and $\rho$ parameters, respectively. See Links for more choices.
- `imu`, `irho`  Initial values. A NULL means an initial value is chosen internally. See CommonVGAMffArguments for more information.
- `nsimEIM`, `zero`  See CommonVGAMffArguments for more information.
Details

The two-parameter cardioid distribution has a density that can be written as

\[
f(y; \mu, \rho) = \frac{1}{2\pi} (1 + 2\rho \cos(y - \mu))
\]

where \(0 < y < 2\pi\), \(0 < \mu < 2\pi\), and \(-0.5 < \rho < 0.5\) is the concentration parameter. The default link functions enforce the range constraints of the parameters.

For positive \(\rho\) the distribution is unimodal and symmetric about \(\mu\). The mean of \(Y\) (which make up the fitted values) is \(\pi + (\rho/\pi)((2\pi - \mu) \sin(2\pi - \mu) + \cos(2\pi - \mu) - \mu \sin(\mu) - \cos(\mu)).\)

Value

An object of class "vglmff" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, \texttt{rrvglm} and \texttt{vgam}.

Warning

Numerically, this distribution can be difficult to fit because of a log-likelihood having multiple maximums. The user is therefore encouraged to try different starting values, i.e., make use of \texttt{imu} and \texttt{irho}.

Note

Fisher scoring using simulation is used.

Author(s)

T. W. Yee

References


See Also

\texttt{rcard}, \texttt{extlogit}, \texttt{vonmises}.

\texttt{CircStats} and \texttt{circular} currently have a lot more R functions for circular data than the \texttt{VGAM} package.

Examples

```r
## Not run:
cdata <- data.frame(y = rcard(n = 1000, mu = 4, rho = 0.45))
fit <- vglm(y ~ 1, cardioid, data = cdata, trace = TRUE)
coef(fit, matrix=TRUE)
Coef(fit)
c(with(cdata, mean(y)), head(fitted(fit), 1))
summary(fit)
```
cauchit

Cauchit Link Function

Description

Computes the cauchit (tangent) link transformation, including its inverse and the first two derivatives.

Usage

cauchit(theta, bvalue = .Machine$double.eps,
    inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)

Arguments

theta       Numeric or character. See below for further details.

bvalue     See Links.

inverse, deriv, short, tag     Details at Links.

Details

This link function is an alternative link function for parameters that lie in the unit interval. This type of link bears the same relation to the Cauchy distribution as the probit link bears to the Gaussian. One characteristic of this link function is that the tail is heavier relative to the other links (see examples below).

Numerical values of theta close to 0 or 1 or out of range result in Inf, -Inf, NA or NaN.

Value

For deriv = 0, the tangent of theta, i.e., tan(pi * (theta-0.5)) when inverse = FALSE, and if inverse = TRUE then 0.5 + atan(theta)/pi.

For deriv = 1, then the function returns d theta / d eta as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the reciprocal.

Note

Numerical instability may occur when theta is close to 1 or 0. One way of overcoming this is to use bvalue.

As mentioned above, in terms of the threshold approach with cumulative probabilities for an ordinal response this link function corresponds to the Cauchy distribution (see cauchy1).

Author(s)

Thomas W. Yee
References


See Also

logit, probit, cloglog, loge, cauchy, cauchy.

Examples

```r
p <- seq(0.01, 0.99, by=0.01)
cau *chit(p)
max(abs(cauchit(cauchit(p), inverse = TRUE) - p)) # Should be 0

p <- c(seq(-0.02, 0.02, by=0.01), seq(0.97, 1.02, by = 0.01))
cauchit(p) # Has no NAs

## Not run:
par(mfrow = c(2, 2), lwd = (mylwd <- 2))
y <- seq(-4, 4, length = 100)
p <- seq(0.01, 0.99, by = 0.01)

for (d in 0:1) {
  matplot(p, cbind(logit(p, deriv = d), probit(p, deriv = d)),
          type = "n", col = "purple", ylab = "transformation",
          las = 1, main = if (d == 0) "Some probability link functions"
          else "First derivative")
  lines(p, logit(p, deriv = d), col = "limegreen")
  lines(p, probit(p, deriv = d), col = "purple")
  lines(p, cloglog(p, deriv = d), col = "chocolate")
  lines(p, cauchit(p, deriv = d), col = "tan")
  if (d == 0) {
    abline(v = 0.5, h = 0, lty = "dashed")
    legend(0, 4.5, c("logit", "probit", "cloglog", "cauchit"), lwd = mylwd,
           col = c("limegreen", "purple", "chocolate", "tan"))
  } else
    abline(v = 0.5, lty = "dashed")
}

for (d in 0) {
  matplot(y, cbind( logit(y, deriv = d, inverse = TRUE),
                  probit(y, deriv = d, inverse = TRUE)),
          type = "n", col = "purple", xlab = "transformation", ylab = "p",
          main = if (d == 0) "Some inverse probability link functions"
          else "First derivative", las=1)
  lines(y, logit(y, deriv = d, inverse = TRUE), col = "limegreen")
  lines(y, probit(y, deriv = d, inverse = TRUE), col = "purple")
  lines(y, cloglog(y, deriv = d, inverse = TRUE), col = "chocolate")
  lines(y, cauchit(y, deriv = d, inverse = TRUE), col = "tan")
  if (d == 0) {
    abline(h = 0.5, v = 0, lty = "dashed")
    legend(-4, 1, c("logit", "probit", "cloglog", "cauchit"), lwd = mylwd,
           col = c("limegreen", "purple", "chocolate", "tan"))
  } else
    abline(v = 0.5, lty = "dashed")
```

cauchit
cauchy

cauchy

Cauchy Distribution Family Function

Description

Estimates either the location parameter or both the location and scale parameters of the Cauchy distribution by maximum likelihood estimation.

Usage

cauchy(llocation = "identitylink", lscale = "loge",
llocation = NULL, lscale = NULL,
iprobs = seq(0.2, 0.8, by = 0.2),
imethod = 1, nsimEIM = NULL, zero = 2)
cauchyl(scale.arg = 1, llocation = "identitylink",
llocation = NULL, imethod = 1)

Arguments

llocation, lscale
Parameter link functions for the location parameter \( a \) and the scale parameter \( b \). See Links for more choices.

llocation, lscale
Optional initial value for \( a \) and \( b \). By default, an initial value is chosen internally for each.

imethod
Integer, either 1 or 2 or 3. Initial method, three algorithms are implemented. The user should try all possible values to help avoid converging to a local solution. Also, choose another value if convergence fails, or use llocation and/or lscale.

iprobs
Probabilities used to find the respective sample quantiles; used to compute lscale.

zero, nsimEIM See CommonVGAMffArguments for more information.

scale.arg
Known (positive) scale parameter, called \( b \) below.

Details

The Cauchy distribution has density function

\[
f(y; a, b) = \left\{ \pi b \left[ 1 + ((y - a)/b)^2 \right] \right\}^{-1}
\]
where \( y \) and \( a \) are real and finite, and \( b > 0 \). The distribution is symmetric about \( a \) and has a heavy tail. Its median and mode are \( a \), but the mean does not exist. The fitted values are the estimates of \( a \). Fisher scoring is the default but if \( nsmEIM \) is specified then Fisher scoring with simulation is used.

If the scale parameter is known (cauchy1) then there may be multiple local maximum likelihood solutions for the location parameter. However, if both location and scale parameters are to be estimated (cauchy) then there is a unique maximum likelihood solution provided \( n > 2 \) and less than half the data are located at any one point.

**Value**

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

**Warning**

It is well-known that the Cauchy distribution may have local maximums in its likelihood function; make full use of imethod, ilocation, iscale etc.

**Note**

Good initial values are needed. By default these VGAM family functions search for a starting value for \( a \) on a grid. It also pays to select a wide range of initial values via the ilocation and/or iscale and/or imethod arguments.

**Author(s)**

T. W. Yee

**References**


**See Also**

Cauchy, cauchit, studentt, simulate.vlm.
Examples

```r
# Both location and scale parameters unknown
set.seed(123)
cdata <- data.frame(x2 = runif(nn <- 1000))
cdata <- transform(cdata, loc = exp(1 + 0.5 * x2), scale = exp(1))
cdata <- transform(cdata, y2 = rcauchy(nn, loc, scale))
fit2 <- vglm(y2 ~ x2, cauchy(loc = "loge"), data = cdata, trace = TRUE)
coef(fit2, matrix = TRUE)
head(fitted(fit2))  # Location estimates
summary(fit2)

# Location parameter unknown
cdata <- transform(cdata, scale = 0.4)
cdata <- transform(cdata, y1 = rcauchy(nn, loc, scale))
fit1 <- vglm(y1 ~ x2, cauchy(scale = 0.4), data = cdata, trace = TRUE)
coef(fit1, matrix = TRUE)
```

---

cdf.lmscreg  

*Cumulative Distribution Function for LMS Quantile Regression*

### Description

Computes the cumulative distribution function (CDF) for observations, based on a LMS quantile regression.

### Usage

```r
cdf.lmscreg(object, newdata = NULL, ...)
```

### Arguments

- `object`  
  A **VGAM** quantile regression model, i.e., an object produced by modelling functions such as `vglm` and `vgam` with a family function beginning with "lm.s.".

- `newdata`  
  Data frame where the predictions are to be made. If missing, the original data is used.

- `...`  
  Parameters which are passed into functions such as `cdf.lms.yjn`.

### Details

The CDFs returned here are values lying in [0,1] giving the relative probabilities associated with the quantiles `newdata`. For example, a value near 0.75 means it is close to the upper quartile of the distribution.

### Value

A vector of CDF values lying in [0,1].
Note

The data are treated like quantiles, and the percentiles are returned. The opposite is performed by `qtplot.lmscreg`.

The CDF values of the model have been placed in `@post$cdf` when the model was fitted.

Author(s)

Thomas W. Yee

References


See Also

`deplot.lmscreg, qtplot.lmscreg, lms.bcn, lms.bcg, lms.yjn`.

Examples

```r
fit <- vgam(BMI ~ s(age, df=c(4, 2)), lms.bcn(zero = 1), data = bmi.nz)
head(fit@post$cdf)
head(cdf(fit))  # Same
head(depvar(fit))
head(fitted(fit))

cdf(fit, data.frame(age = c(31.5, 39), BMI = c(28.4, 24)))
```

---

cens.gumbel

**Censored Gumbel Distribution**

Description

Maximum likelihood estimation of the 2-parameter Gumbel distribution when there are censored observations. A matrix response is not allowed.

Usage

```r
cens.gumbel(llocation = "identitylink", lscale = "loge",
            iscale = NULL, mean = TRUE, percentiles = NULL, zero = 2)
```
Arguments

location, iscale
Character. Parameter link functions for the location and (positive) scale parameters. See Links for more choices.

iscale
Numeric and positive. Initial value for scale. Recycled to the appropriate length. In general, a larger value is better than a smaller value. The default is to choose the value internally.

mean
Logical. Return the mean? If TRUE then the mean is returned, otherwise percentiles given by the percentiles argument.

percentiles
Numeric with values between 0 and 100. If mean=FALSE then the fitted values are percentiles which must be specified by this argument.

zero
An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The value (possibly values) must be from the set {1,2} corresponding respectively to location and scale. If zero=NULL then all linear/additive predictors are modelled as a linear combination of the explanatory variables. The default is to fit the shape parameter as an intercept only.

Details

This VGAM family function is like gumbel but handles observations that are left-censored (so that the true value would be less than the observed value) else right-censored (so that the true value would be greater than the observed value). To indicate which type of censoring, input extra=list(leftcensored=vec1, rightcensored=vec2) where vec1 and vec2 are logical vectors the same length as the response. If the two components of this list are missing then the logical values are taken to be FALSE. The fitted object has these two components stored in the extra slot.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

Numerical problems may occur if the amount of censoring is excessive.

Note

See gumbel for details about the Gumbel distribution. The initial values are based on assuming all uncensored observations, therefore could be improved upon.

Author(s)

T. W. Yee

References

See Also
gumbel, egumbel, rgumbel, guplot, gev, venice.

Examples

# Example 1
ystar <- venice[["r1"]]
# Use the first order statistic as the response
nn <- length(ystar)
L <- runif(nn, 100, 104) # Lower censoring points
U <- runif(nn, 130, 135) # Upper censoring points
y <- pmax(L, ystar) # Left censored
y <- pmin(U, y) # Right censored
extra <- list(leftcensored = ystar < L, rightcensored = ystar > U)
fit <- vglm(y ~ scale(year), data = venice, trace = TRUE, extra = extra,
            cens.gumbel(mean = FALSE, perc = c(5, 25, 50, 75, 95)))
coef(fit, matrix = TRUE)
head(fitted(fit))
fit@extra

# Example 2: simulated data
nn <- 1000
ystar <- rgumbel(nn, loc = 1, scale = exp(0.5)) # The uncensored data
L <- runif(nn, -1, 1) # Lower censoring points
U <- runif(nn, 2, 5) # Upper censoring points
y <- pmax(L, ystar) # Left censored
y <- pmin(U, y) # Right censored
## Not run: par(mfrow = c(1, 2)); hist(ystar); hist(y);
extra <- list(leftcensored = ystar < L, rightcensored = ystar > U)
fit <- vglm(y ~ 1, trace = TRUE, extra = extra, cens.gumbel)
coef(fit, matrix = TRUE)

cens.normal

Censored Normal Distribution

Description

Maximum likelihood estimation for the normal distribution with left and right censoring.

Usage

cens.normal(lmu = "identitylink", lsd = "loge", imethod = 1, zero = 2)

Arguments

lmu, lsd Parameter link functions applied to the mean and standard deviation parameters. See Links for more choices. The standard deviation is a positive quantity, therefore a log link is the default.

imethod Initialization method. Either 1 or 2, this specifies two methods for obtaining initial values for the parameters.
zero

An integer vector, containing the value 1 or 2. If so, the mean or standard deviation respectively are modelled as an intercept only. Setting zero = NULL means both linear/additive predictors are modelled as functions of the explanatory variables.

Details

This function is like uninormal but handles observations that are left-censored (so that the true value would be less than the observed value) else right-censored (so that the true value would be greater than the observed value). To indicate which type of censoring, input extra = list(leftcensored = vec1, rightcensored = vec2) where vec1 and vec2 are logical vectors the same length as the response. If the two components of this list are missing then the logical values are taken to be FALSE. The fitted object has these two components stored in the extra slot.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

This function is an alternative to tobit but cannot handle a matrix response and uses different working weights. If there are no censored observations then uninormal is recommended instead.

Author(s)

T. W. Yee

See Also

tobit, uninormal, double.cens.normal.

Examples

```r
## Not run:
cdata <- data.frame(x2 = runif(nn <- 1000))  # ystar are true values
cdata <- transform(cdata, ystar = rnorm(nn, m = 100 + 15 * x2, sd = exp(3)))
with(cdata, hist(ystar))
cdata <- transform(cdata, L = runif(nn, 80, 90)), # Lower censoring points
    U = runif(nn, 130, 140))  # Upper censoring points
    cdata <- transform(cdata, y = pmax(L, ystar))) # Left censored
    cdata <- transform(cdata, y = pmin(U, y))   # Right censored
    with(cdata, hist(y))
    Extra <- list(leftcensored = with(cdata, ystar < L),
        rightcensored = with(cdata, ystar > U))
    fit1 <- vglm(y ~ x2, cens.normal, data = cdata, crit = "c", extra = Extra)
    fit2 <- vglm(y ~ x2, tobit(Lower = with(cdata, L), Upper = with(cdata, U)),
        data = cdata, crit = "c", trace = TRUE)
    coef(fit1, matrix = TRUE)
    max(abs(coef(fit1, matrix = TRUE) - coef(fit2, matrix = TRUE))) # Should be 0
    names(fit1@extra)
```
Censored Poisson Family Function

Description
Family function for a censored Poisson response.

Usage
cens.poisson(link = "log", imu = NULL)

Arguments
- link: Link function applied to the mean; see Links for more choices.
- imu: Optional initial value; see CommonVGAMArguments for more information.

Details
Often a table of Poisson counts has an entry $J+$ meaning $\geq J$. This family function is similar to poissonff but handles such censored data. The input requires SurvS4. Only a univariate response is allowed. The Newton-Raphson algorithm is used.

Value
An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning
As the response is discrete, care is required with Surv, especially with "interval" censored data because of the (start, end] format. See the examples below. The examples have $y < L$ as left censored and $y \geq U$ (formatted as $U+$) as right censored observations, therefore $L \leq y < U$ is for uncensored and/or interval censored observations. Consequently the input must be tweaked to conform to the (start, end] format.

Note
The function poissonff should be used when there are no censored observations. Also, NAs are not permitted with SurvS4, nor is type = "counting".

Author(s)
Thomas W. Yee
References

See survival for background.

See Also

SurvS4, poissonff, Links.

Examples

# Example 1: right censored data
set.seed(123); U <- 20
cdata <- data.frame(y = rpois(N <- 100, exp(3)))
cdata <- transform(cdata, cy = pmin(U, y),
                      rensored = (y >= U))
cdata <- transform(cdata, status = ifelse(rensored, 0, 1))
with(cdata, table(cy))
with(cdata, table(rensored))
with(cdata, table(ii <- print(SurvS4(cy, status))))) # Check; U+ means >= U
fit <- vglm(SurvS4(cy, status) ~ 1, cens.poisson, data = cdata, trace = TRUE)
coef(fit, matrix = TRUE)
table(print(depvar(fit))) # Another check; U+ means >= U

# Example 2: left censored data
L <- 15
cdata <- transform(cdata, cY = pmax(L, y),
                      lcensored = y < L) # Note y < L, not cY == L or y <= L
cdata <- transform(cdata, status = ifelse(lcensored, 0, 1))
with(cdata, table(cy))
with(cdata, table(lcensored))
with(cdata, table(ii <- print(SurvS4(cY, status, type = "left"))))) # Check
fit <- vglm(SurvS4(cY, status, type = "left") ~ 1, cens.poisson, data = cdata, trace = TRUE)
coef(fit, matrix = TRUE)

# Example 3: interval censored data
cdata <- transform(cdata, Lvec = rep(L, len = N),
                      Uvec = rep(U, len = N))
cdata <- transform(cdata, icensored = Lvec <= y & y < Uvec) # Not lcensored or rcensored
with(cdata, table(icensored))
cdata <- transform(cdata, status = rep(3, N)) # 3 means interval censored
cdata <- transform(cdata, status = ifelse(rcensored, 0, status)) # 0 means right censored
cdata <- transform(cdata, status = ifelse(lcensored, 2, status)) # 2 means left censored
# Have to adjust Lvec and Uvec because of the (start, end) format:
cdata$Lvec[with(cdata, icensored)] <- cdata$Lvec[with(cdata, icensored)] - 1
cdata$Uvec[with(cdata, icensored)] <- cdata$Uvec[with(cdata, icensored)] - 1
cdata$Lvec[with(cdata, lcensored)] <- cdata$Lvec[with(cdata, lcensored)]
cdata$Uvec[with(cdata, rcensored)] <- cdata$Uvec[with(cdata, rcensored)]
with(cdata, table(ii <- print(SurvS4(Lvec, Uvec, status, type = "interval"))))) # Check
fit <- vglm(SurvS4(Lvec, Uvec, status, type = "interval") ~ 1,
            cens.poisson, data = cdata, trace = TRUE)
Cystic Fibrosis Data

Description

This data frame concerns families data and cystic fibrosis.

Usage

data(cfibrosis)

Format

A data frame with 24 rows on the following 4 variables.

siblings, affected, ascertained, families Over ascertained families, the $k$th ascertained family has $s_k$ siblings of whom $r_k$ are affected and $a_k$ are ascertained.

Details

The data set allows a classical segregation analysis to be performed. In particular, to test Mendelian segregation ratios in nuclear family data. The likelihood has similarities with seq2binomial.

Source

The data is originally from Crow (1965) and appears as Table 2.3 of Lange (2002).
Examples

cfibrosis
summary(cfibrosis)

cgo

Redirects the user to cgo

Description

Redirects the user to the function cgo.

Usage

cgo(...)

Arguments

... Ignored.

Details

The former function cgo has been renamed cqo because CGO (for canonical Gaussian ordination) is a confusing and inaccurate name. CQO (for constrained quadratic ordination) is better. This new nomenclature described in Yee (2006).

Value

Nothing is returned; an error message is issued.

Warning

The code, therefore, in Yee (2004) will not run without changing the "g" to a "q".

Author(s)

Thomas W. Yee

References


See Also

cqo.
Examples

```r
## Not run:
cgo()

## End(Not run)
```

---

**chest.nz**

_Chest Pain in NZ Adults Data_

**Description**

Presence/absence of chest pain in 10186 New Zealand adults.

**Usage**

`data(chest.nz)`

**Format**

A data frame with 73 rows and the following 5 variables.

- `age` a numeric vector; age (years).
- `nolnor` a numeric vector of counts; no pain on LHS or RHS.
- `nolr` a numeric vector of counts; no pain on LHS but pain on RHS.
- `lnor` a numeric vector of counts; no pain on RHS but pain on LHS.
- `lr` a numeric vector of counts; pain on LHS and RHS of chest.

**Details**

Each adult was asked their age and whether they experienced any pain or discomfort in their chest over the last six months. If yes, they indicated whether it was on their LHS and/or RHS of their chest.

**Source**

Examples

```r
# Not run:
fit <- vgam(cbind(nolnor, nolr, lnor, lr) ~ s(age, c(4, 3)),
            binom2.or(exchan = TRUE, zero = NULL), data = chest.nz)
coef(fit, matrix = TRUE)

# Not run
# Not run: plot(fit, which.cf = 2, se = TRUE)
```

---

**chinese.nz**

**Chinese Population in New Zealand 1867–2001 Data**

---

**Description**

The Chinese population in New Zealand from 1867 to 2001, along with the whole of the New Zealand population.

**Usage**

```r
data(chinese.nz)
```

**Format**

A data frame with 27 observations on the following 4 variables.

- `year`  Year.
- `male`  Number of Chinese males.
- `female`  Number of Chinese females.
- `nz`  Total number in the New Zealand population.

**Details**

Historically, there was a large exodus of Chinese from the Guangdong region starting in the mid-1800s to the gold fields of South Island of New Zealand, California, and southern Australia, etc. Discrimination then meant that only men were allowed entry, to hinder permanent settlement. In the case of New Zealand, the government relaxed its immigration laws after WWII to allow wives of Chinese already in NZ to join them because China had been among the Allied powers. Gradual relaxation in the immigration and an influx during the 1980s meant the Chinese population became increasingly demographically normal over time.

The NZ total for the years 1867 and 1871 exclude the Maori population. Three modifications have been made to the female column to make the data internally consistent with the original table.

**References**

Examples

```r
## Not run: par(mfrow = c(1, 2))
plot(female / (male + female) ~ year, chinese.nz, type = "b",
ylab = "Proportion", col = "blue", las = 1,
cex = 0.015 * sqrt(male + female),
#  cex = 0.10 * sqrt(male + female)^1.5 / sqrt(female) / sqrt(male)),
main = "Proportion of NZ Chinese that are female")
abline(h = 0.5, lty = "dashed", col = "gray")

fit1.cnz <- vglm(cbind(female, male) ~ year,
                 binomialff, data = chinese.nz)
fit2.cnz <- vglm(cbind(female, male) ~ sm.poly(year, 2), binomialff, data = chinese.nz)
fit4.cnz <- vglm(cbind(female, male) ~ sm.bs(year, 5), binomialff, data = chinese.nz)

lines(fitted(fit1.cnz) ~ year, chinese.nz, col = "purple", lty = 1)
lines(fitted(fit2.cnz) ~ year, chinese.nz, col = "green", lty = 2)
lines(fitted(fit4.cnz) ~ year, chinese.nz, col = "orange", lwd = 2, lty = 1)
legend("bottomright", col = c("purple", "green", "orange"),
       lty = c(1, 2, 1), lwd = c("linear", "quadratic", "B-spline"))

plot(100*(male+female)/nz ~ year, chinese.nz, type = "b",
ylab = "Percent",
     ylim = c(0, max(100*(male+female)/nz)), col = "blue", las = 1,
     main = "Percent of NZers that are Chinese")
abline(h = 0, lty = "dashed", col = "gray")
## End(Not run)
```

---

chisq  

Chi-squared Distribution

Description

Maximum likelihood estimation of the degrees of freedom for a chi-squared distribution.

Usage

```
chisq(link = "loge", zero = NULL)
```

Arguments

- `link`, `zero` See `CommonVGAMArguments` for information.

Details

The degrees of freedom is treated as a parameter to be estimated, and as real (not integer). Being positive, a log link is used by default. Fisher scoring is used.
Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Note

Multiple responses are permitted. There may be convergence problems if the degrees of freedom is very large or close to zero.

Author(s)

T. W. Yee

References


See Also

`chisquare`, `uninormal`.

Examples

cdata <- data.frame(x2 = runif(nn <- 1000))
cdata <- transform(cdata, y1 = rchisq(nn, df = exp(1 - 1 * x2)),
y2 = rchisq(nn, df = exp(2 - 2 * x2)))
fit <- vglm(cbind(y1, y2) ~ x2, chisq, data = cdata, trace = TRUE)
coef(fit, matrix = TRUE)

clo

Redirects the user to `rrvglm`

Description

Redirects the user to the function `rrvglm`.

Usage

clo(...)

Arguments

... Ignored.
Details

CLO stands for *constrained linear ordination*, and is fitted with a statistical class of models called *reduced-rank vector generalized linear models* (RR-VGLMs). It allows for generalized reduced-rank regression in that response types such as Poisson counts and presence/absence data can be handled.

Currently in the VGAM package, `rrvglm` is used to fit RR-VGLMs. However, the Author’s opinion is that linear responses to a latent variable (composite environmental gradient) is not as common as unimodal responses, therefore `cqo` is often more appropriate.

The new CLO/CQO/CAO nomenclature described in Yee (2006).

Value

Nothing is returned; an error message is issued.

Author(s)

Thomas W. Yee

References


See Also

`rrvglm`, `cqo`.

Examples

```r
## Not run:
clo()
## End(Not run)
```

cloglog  Complementary Log-log Link Function

Description

Computes the complementary log-log transformation, including its inverse and the first two derivatives.
**Usage**

```r
cloglog(theta, bvalue = NULL, inverse = FALSE, deriv = 0,
       short = TRUE, tag = FALSE)
```

**Arguments**

- `theta` Numeric or character. See below for further details.
- `bvalue` See [Links](#) for general information about links.
- `inverse, deriv, short, tag`
  
  Details at [Links](#).

**Details**

The complementary log-log link function is commonly used for parameters that lie in the unit interval. Numerical values of `theta` close to 0 or 1 or out of range result in `Inf`, `-Inf`, `NA` or `NaN`.

**Value**

For `deriv = 0`, the complimentary log-log of `theta`, i.e., \( \log(-\log(1 - \theta)) \) when `inverse = FALSE`, and if `inverse = TRUE` then `1-\exp(-\exp(\theta))`.

For `deriv = 1`, then the function returns \( \frac{d\theta}{d\eta} \) as a function of `theta` if `inverse = FALSE`, else if `inverse = TRUE` then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base `e`.

**Note**

Numerical instability may occur when `theta` is close to 1 or 0. One way of overcoming this is to use `bvalue`.

Changing 1s to 0s and 0s to 1s in the response means that effectively a loglog link is fitted. That is, `transform y` by `1 - y`. That’s why only one of `cloglog` and `loglog` is written.

With constrained ordination (e.g., `cqo` and `cao`) used with `binomialff`, a complementary log-log link function is preferred over the default `logit` link, for a good reason. See the example below.

In terms of the threshold approach with cumulative probabilities for an ordinal response this link function corresponds to the extreme value distribution.

**Author(s)**

Thomas W. Yee

**References**


**See Also**

[Links](#), `logit`, `probit`, `cauchit`.
Examples

```r
p <- seq(0.01, 0.99, by = 0.01)
cloglog(p)
max(abs(cloglog(cloglog(p), inverse = TRUE) - p))  # Should be 0

p <- c(seq(-0.02, 0.02, by = 0.01), seq(0.97, 1.02, by = 0.01))
cloglog(p)  # Has NAs
cloglog(p, bvalue = .Machine$double.eps)  # Has no NAs

## Not run:
p <- seq(0.01, 0.99, by = 0.01)
plot(p, logit(p), type = "l", col = "limegreen", lwd = 2, las = 1,
     main = "Some probability link functions", ylab = "transformation")
lines(p, probit(p), col = "purple", lwd = 2)
lines(p, cloglog(p), col = "chocolate", lwd = 2)
lines(p, cauchit(p), col = "tan", lwd = 2)
abline(v = 0.5, h = 0, lty = "dashed")
legend(0.1, 4, c("logit", "probit", "cloglog", "cauchit"),
       col = c("limegreen", "purple", "chocolate", "tan"), lwd = 2)

## End(Not run)

## Not run:
# This example shows that a cloglog link is preferred over the logit
n <- 500; p <- 5; S <- 3; Rank <- 1  # Species packing model:
mydata <- rcqo(n, p, S, eq.tol = TRUE, es.opt = TRUE, eq.max = TRUE,
               family = "binomial", hi.abundance = 5, seed = 123,
               Rank = Rank)
fitc <- cqo(attr(mydata, "formula"), I.tol = TRUE, data = mydata,
            fam = binomialff(multiple.responses = TRUE, link = "cloglog"),
            Rank = Rank)
fitl <- cqo(attr(mydata, "formula"), I.tol = TRUE, data = mydata,
            fam = binomialff(multiple.responses = TRUE, link = "logit"),
            Rank = Rank)

# Compare the fitted models (cols 1 and 3) with the truth (col 2)
cbind(concoef(fitc), attr(mydata, "concoefficients"), concoef(fitl))

## End(Not run)
```

coalminers

Description

Coalminers who are smokers without radiological pneumoconiosis, classified by age, breathlessness and wheeze.

Usage

data(coalminers)
Format

A data frame with 9 age groups with the following 5 columns.

- **BW** Counts with breathlessness and wheeze.
- **BnW** Counts with breathlessness but no wheeze.
- **nBW** Counts with no breathlessness but wheeze.
- **nBnW** Counts with neither breathlessness or wheeze.
- **age** Age of the coal miners (actually, the midpoints of the 5-year category ranges).

Details

The data were published in Ashford and Sowden (1970). A more recent analysis is McCullagh and Nelder (1989, Section 6.6).

Source


References


Examples

```r
str(coalminers)
```

---

**Description**

`Coef` is a generic function which computes model coefficients from objects returned by modelling functions. It is an auxiliary function to `coef` that enables extra capabilities for some specific models.

**Usage**

```r
Coef(object, ...)
```

**Arguments**

- `object` An object for which the computation of other types of model coefficients or quantities is meaningful.
- `...` Other arguments fed into the specific methods function of the model.
Details

This function can often be useful for \texttt{vglm} objects with just an intercept term in the RHS of the formula, e.g., \(y \sim 1\). Then often this function will apply the inverse link functions to the parameters. See the example below.

For reduced-rank VGLMs, this function can return the \(A, C\) matrices, etc.

For quadratic and additive ordination models, this function can return ecological meaningful quantities such as tolerances, optimums, maximums.

Value

The value returned depends specifically on the methods function invoked.

Warning

This function may not work for all \texttt{VGAM} family functions. You should check your results on some artificial data before applying it to models fitted to real data.

Author(s)

Thomas W. Yee

References


See Also

\texttt{coef, Coef.vlm, Coef.rrvglm, Coef.qrrvglm, depvar}.

Examples

```r
nn <- 1000
bdata <- data.frame(y = rbeta(nn, shape1 = 1, shape2 = 3)) # Original scale
fit <- vglm(y ~ 1, betaR, data = bdata, trace = TRUE) # Intercept-only model
coef(fit, matrix = TRUE) # Both on a log scale
coef(fit) # On the original scale
```

\texttt{Coef.qrrvglm} \hspace{1cm} \textit{Returns Important Matrices etc. of a QO Object}

Description

This methods function returns important matrices etc. of a QO object.

Usage

\texttt{Coef.qrrvglm(object, varI.latvar = FALSE, refResponse = NULL, ...)}
Arguments

object A CQO object. The former has class "qrrvglm".

variNlatvar Logical indicating whether to scale the site scores (latent variables) to have variance-covariance matrix equal to the rank-R identity matrix. All models have uncorrelated site scores (latent variables), and this option stretches or shrinks the ordination axes if TRUE. See below for further details.

refResponse Integer or character. Specifies the reference response or reference species. By default, the reference species is found by searching sequentially starting from the first species until a positive-definite tolerance matrix is found. Then this tolerance matrix is transformed to the identity matrix. Then the sites scores (latent variables) are made uncorrelated. See below for further details.

... Currently unused.

Details

If iNtolerances=TRUE or eqNtolerances=TRUE (and its estimated tolerance matrix is positive-definite) then all species’ tolerances are unity by transformation or by definition, and the spread of the site scores can be compared to them. Vice versa, if one wishes to compare the tolerances with the sites score variability then setting variNlatvar=TRUE is more appropriate.

For rank-2 QRR-VGLMs, one of the species can be chosen so that the angle of its major axis and minor axis is zero, i.e., parallel to the ordination axes. This means the effect on the latent vars is independent on that species, and that its tolerance matrix is diagonal. The argument refResponse allows one to choose which is the reference species, which must have a positive-definite tolerance matrix, i.e., is bell-shaped. If refResponse is not specified, then the code will try to choose some reference species starting from the first species. Although the refResponse argument could possibly be offered as an option when fitting the model, it is currently available after fitting the model, e.g., in the functions Coef.qrrvglm and lvplot.qrrvglm.

Value

The A, B1, C, T, D matrices/arrays are returned, along with other slots. The returned object has class "Coef.qrrvglm" (see Coef.qrrvglm-class).

Note

Consider an equal-tolerances Poisson/binomial CQO model with noRRR = ~ 1. For \( R = 1 \) it has about \( 2S + p_2 \) parameters. For \( R = 2 \) it has about \( 3S + 2p_2 \) parameters. Here, \( S \) is the number of species, and \( p_2 = p - 1 \) is the number of environmental variables making up the latent variable. For an unequal-tolerances Poisson/binomial CQO model with noRRR = ~ 1, it has about \( 3S - 1 + p_2 \) parameters for \( R = 1 \), and about \( 6S - 3 + 2p_2 \) parameters for \( R = 2 \). Since the total number of data points is \( nS \), where \( n \) is the number of sites, it pays to divide the number of data points by the number of parameters to get some idea about how much information the parameters contain.

Author(s)

Thomas W. Yee
References


See Also
cqo, Coef.qrrvglm-class, print.Coef.qrrvglm, lvplot.qrrvglm.

Examples

```r
set.seed(123)
x2 <- rnorm(n <- 100)
x3 <- rnorm(n)
x4 <- rnorm(n)
latvar1 <- 0 + x3 - 2*x4
lambda1 <- exp(3 - 0.5 * (latvar1-0)^2)
lambda2 <- exp(2 - 0.5 * (latvar1-1)^2)
lambda3 <- exp(2 - 0.5 * ((latvar1+4)/2)^2)  # Unequal tolerances
y1 <- rpois(n, lambda1)
y2 <- rpois(n, lambda2)
y3 <- rpois(n, lambda3)
set.seed(111)
# vvv pl <- cqo(cbind(y1, y2, y3) ~ x2 + x3 + x4, poissonff, trace = FALSE)
## Not run: lvplot(pl, y = TRUE, lcol = 1:3, pch = 1:3, pcol = 1:3)

# vvv Coef(pl)
# vvv print(Coef(pl), digits=3)
```

Coef.qrrvglm-class  Class “Coef.qrrvglm”

Description

The most pertinent matrices and other quantities pertaining to a QRR-VGLM (CQO model).

Objects from the Class

Objects can be created by calls of the form Coef(object,...) where object is an object of class "qrrvglm" (created by cqo).

In this document, $R$ is the rank, $M$ is the number of linear predictors and $n$ is the number of observations.
Slots

A: Of class "matrix", A, which are the linear ‘coefficients’ of the matrix of latent variables. It is M by R.

B1: Of class "matrix", B1. These correspond to terms of the argument noRRR.

C: Of class "matrix", C, the canonical coefficients. It has R columns.

Constrained: Logical. Whether the model is a constrained ordination model.

D: Of class "array", D[,j] is an order-Rank matrix, for j = 1,...,M. Ideally, these are negative-definite in order to make the response curves/surfaces bell-shaped.

Rank: The rank (dimension, number of latent variables) of the RR-VGLM. Called R.

latvar: n by R matrix of latent variable values.

latvar.order: Of class "matrix", the permutation returned when the function order is applied to each column of latvar. This enables each column of latvar to be easily sorted.

Maximum: Of class "numeric", the M maximum fitted values. That is, the fitted values at the optimums for noRRR = ~ 1 models. If noRRR is not ~ 1 then these will be NAs.

NOS: Number of species.

Optimum: Of class "matrix", the values of the latent variables where the optimums are. If the curves are not bell-shaped, then the value will be NA or NaN.

Optimum.order: Of class "matrix", the permutation returned when the function order is applied to each column of Optimum. This enables each row of Optimum to be easily sorted.

bellshaped: Vector of logicals: is each response curve/surface bell-shaped?

dispersion: Dispersion parameter(s).

Dzero: Vector of logicals, is each of the response curves linear in the latent variable(s)? It will be if and only if D[,j] equals O, for j = 1,...,M.

Tolerance: Object of class "array", Tolerance[,j] is an order-Rank matrix, for j = 1,...,M, being the matrix of tolerances (squared if on the diagonal). These are denoted by T in Yee (2004). Ideally, these are positive-definite in order to make the response curves/surfaces bell-shaped. The tolerance matrices satisfy $T_s = -\frac{1}{2}D_s^{-1}$.

Author(s)

Thomas W. Yee

References


See Also

Coef.qrrvglm, cqo, print.Coeff.qrrvglm.
Examples
x2 <- rnorm(n <- 100)
x3 <- rnorm(n)
x4 <- rnorm(n)
latvar1 <- 0 + x3 - 2*x4
lambda1 <- exp(3 - 0.5 * (latvar1-0)^2)
lambda2 <- exp(2 - 0.5 * (latvar1-1)^2)
lambda3 <- exp(2 - 0.5 * ((latvar1+4)/2)^2)
y1 <- rpois(n, lambda1)
y2 <- rpois(n, lambda2)
y3 <- rpois(n, lambda3)
yy <- cbind(y1, y2, y3)
# vvv p1 <- cpo(yy ~ x2 + x3 + x4, fam = poissonff, trace = FALSE)
## Not run:
lvplot(p1, y = TRUE, lcol = 1:3, pch = 1:3, pcol = 1:3)

## End(Not run)
# vvv print(Coef(p1), digits = 3)

---

Coef.rvglm

Returns Important Matrices etc. of a RR-VGLM Object

Description
This methods function returns important matrices etc. of a RR-VGLM object.

Usage
Coef.rvglm(object, ...)

Arguments

  object  An object of class "rvglm".

  ...  Currently unused.

Details
The A, B1, C matrices are returned, along with other slots. See rvglm for details about RR-VGLMs.

Value
An object of class "Coef.rvglm" (see Coef.rvglm-class).

Note
This function is an alternative to coef.rvglm.
**Author(s)**

Thomas W. Yee

**References**


**See Also**

`coef.rrvglm-class`, `print.Coef.rrvglm`, `rrvglm`.

**Examples**

```r
# Rank-1 stereotype model of Anderson (1984)
pneumo <- transform(pneumo, let = log(exposure.time), x3 = runif(nrow(pneumo)))
fit <- rrvglm(cbind(normal, mild, severe) ~ let + x3, multinomial, data = pneumo)
coef(fit, matrix = TRUE)
Coef(fit)
```

**Description**

The most pertinent matrices and other quantities pertaining to a RR-VGLM.

**Objects from the Class**

Objects can be created by calls of the form `Coef(object, ...) where object is an object of class rrvglm (see rrvglm-class).`

In this document, $M$ is the number of linear predictors and $n$ is the number of observations.

**Slots**

- **A**: Of class "matrix", $A$.
- **B1**: Of class "matrix", $B_1$.
- **C**: Of class "matrix", $C$.
- **rank**: The rank of the RR-VGLM.
- **colx1.index**: Index of the columns of the "vlm"-type model matrix corresponding to the variables in $x_1$. These correspond to $B_1$.
- **colx2.index**: Index of the columns of the "vlm"-type model matrix corresponding to the variables in $x_2$. These correspond to the reduced-rank regression.
- **atilde**: Object of class "matrix", the $A$ matrix with the corner rows removed. Thus each of the elements have been estimated. This matrix is returned only if corner constraints were used.
Author(s)
Thomas W. Yee

References

See Also
`coef.rrvglm`, `rrvglm`, `rrvglm-class`, `print.Coef.rrvglm`.

Examples
```r
# Rank-1 stereotype model of Anderson (1984)
pneumo <- transform(pneumo, let = log(exposure.time), x3 = runif(nrow(pneumo)))
fit <- rrvglm(cbind(normal, mild, severe) ~ let + x3, multinomial, data = pneumo)
coef(fit, matrix = TRUE)
Coef(fit)
# print(Coeff(fit), digits = 3)
```

---

**Coef.vlm**

*Extract Model Coefficients for VLM Objects*

**Description**
Amongst other things, this function applies inverse link functions to the parameters of intercept-only VGLMs.

**Usage**
```
Coef.vlm(object, ...)
```

**Arguments**
- `object` A fitted model.
- `...` Arguments which may be passed into `coef`.

**Details**
Most VGAM family functions apply a link function to the parameters, e.g., positive parameter are often have a log link, parameters between 0 and 1 have a logit link. This function can back-transform the parameter estimates to the original scale.

**Value**
For intercept-only models (e.g., formula is `y ~ 1`) the back-transformed parameter estimates can be returned.
**Warning**

This function may not work for all VGAM family functions. You should check your results on some artificial data before applying it to models fitted to real data.

**Author(s)**

Thomas W. Yee

**References**


**See Also**

`coef`, `coef`.

**Examples**

```r
set.seed(123); nn <- 1000
data <- data.frame(y = rbeta(nn, shape1 = 1, shape2 = 3))
fit <- vglm(y ~ 1, betaff, data = bdata, trace = TRUE) # intercept-only model
coefficients(fit, matrix = TRUE) # log scale
Coef(fit) # On the original scale
```

---

**coefvlm** *(Extract Model Coefficients)*

**Description**

Extracts the estimated coefficients from VLM objects such as VGLMs.

**Usage**

```r
coeffvlm(object, matrix.out = FALSE, label = TRUE, colon = FALSE)
```

**Arguments**

- `object`:
  - An object for which the extraction of coefficients is meaningful. This will usually be a `vglm` object.
- `matrix.out`:
  - Logical. If TRUE then a matrix is returned. The explanatory variables are the rows. The linear/additive predictors are the columns. The constraint matrices are used to compute this matrix.
- `label`:
  - Logical. If FALSE then the names of the vector of coefficients are set to NULL.
- `colon`:
  - Logical. If it only appears in one linear/additive predictor then the :1 is omitted by default. Then setting colon = TRUE will add the :1.
Details

This function works in a similar way to applying coef() to a `lm` or `glm` object. However, for VGLMs, there are more options available.

Value

A vector usually. A matrix if `matrix.out = TRUE`.

Author(s)

Thomas W. Yee

References


See Also

`vglm`, `coef`.

Examples

```r
zdata <- data.frame(x2 = runif(nn <- 200))
zdata <- transform(zdata, pstr0 = logit(-0.5 + 1*x2, inverse = TRUE),
                   lambda = loge( 0.5 + 2*x2, inverse = TRUE))
zdata <- transform(zdata, y2 = rzipois(nn, lambda, pstr0 = pstr0))

fit2 <- vglm(y2 ~ x2, zipoisson(zero = 1), data = zdata, trace = TRUE)
coef(fit2, matrix = TRUE)  # Always a good idea
coef(fit2)
coef(fit2, colon = TRUE)
```
Usage

TypicalVGAMfamilyFunction(ls = "log", 
ls = NULL, 
link.list = list("(Default)" = "identitylink", 
x2 = "log", 
x3 = "logoff", 
x4 = "multilogit", 
x5 = "multilogit"), 
earg.list = list("(Default)" = list(), 
x2 = list(), 
x3 = list(offset = -1), 
x4 = list(), 
x5 = list()), 
gs = exp(-5:5), 
par = TRUE, 
is = NULL, im = 1, 
type.fit = c("mean", "pobs", "prstr", "onempstr"), 
probs.x = c(0.15, 0.85), 
probs.y = c(0.25, 0.50, 0.75), 
multiple.responses = FALSE, earg.list = FALSE, 
whitespace = FALSE, bred = FALSE, lss = TRUE, 
oim = FALSE, nsim = 100, zero = NULL)

Arguments

ls
Character. Link function applied to a parameter and not necessarily a mean. See Links for a selection of choices. If there is only one parameter then this argument is often called link.

link.list, earg.list
Some VGAM family functions (such as normal.vcm) implement models with potentially lots of parameter link functions. These two arguments allow many such links and extra arguments to be inputted more easily. One has something like link.list = list("(Default)" = "identitylink", x2 = "log", x3 = "logoff") and earg.list = list("(Default)" = list(), x2 = list(), x3 = list(offset = -1)). Then any unnamed terms will have the default link with its corresponding extra argument. Note: the multilogit link is also possible, and if so, at least two instances of it are necessary. Then the last term is the baseline/reference group.

is
Optional initial values can often be inputted using an argument beginning with "i". For example, "is" and "ilocation", or just "init" if there is one parameter. A value of NULL means a value is computed internally, i.e., a self-starting VGAM family function. If a failure to converge occurs make use of these types of arguments.

gs
Grid-search initial values can be inputted using an argument beginning with "g", e.g., "gs", "gshape" and "gscale". If argument is is inputted then that has precedence over gs, etc.
If the grid search is 2-dimensional then it is advisable not to make the vectors too long as a nested for loop may be used. Ditto for 3-dimensions.
**parallel**

A logical, or a simple formula specifying which terms have equal/unequal coefficients. The formula must be simple, i.e., additive with simple main effects terms. Interactions and nesting etc. are not handled. To handle complex formulas use the `constraints` argument (of `vglm` etc.); however, there is a lot more setting up involved and things will not be as convenient.

Here are some examples. 1. `parallel = TRUE ~ x2 + x5` means the parallelism assumption is only applied to $X_2, X_5$, and the intercept. 2. `parallel = TRUE ~ -1` and `parallel = TRUE ~ 0` mean the parallelism assumption is applied to no variables at all. Similarly, `parallel = FALSE ~ -1` and `parallel = FALSE ~ 0` mean the parallelism assumption is applied to all the variables including the intercept. 3. `parallel = FALSE ~ x2 - 1` and `parallel = FALSE ~ x2 + 0` applies the parallelism constraint to all terms (including the intercept) except for $X_2$.

This argument is common in **VGAM** family functions for categorical responses, e.g., `cumulative`, `acat`, `cratio`, `sratio`. For the proportional odds model (`cumulative`) having parallel constraints applied to each explanatory variable (except for the intercepts) means the fitted probabilities do not become negative or greater than 1. However this parallelism or proportional-odds assumption ought to be checked.

**nsimEIM**

Some **VGAM** family functions use simulation to obtain an approximate expected information matrix (EIM). For those that do, the `nsimEIM` argument specifies the number of random variates used per observation; the mean of `nsimEIM` random variates is taken. Thus `nsimEIM` controls the accuracy and a larger value may be necessary if the EIMs are not positive-definite. For intercept-only models ($y ~ QI$) the value of `nsimEIM` can be smaller (since the common value used is also then taken as the mean over the observations), especially if the number of observations is large.

Some **VGAM** family functions provide two algorithms for estimating the EIM. If applicable, set `nsimEIM = NULL` to choose the other algorithm.

**imethod**

An integer with value 1 or 2 or 3 or ... which specifies the initialization method for some parameters or a specific parameter. If failure to converge occurs try the next higher value, and continue until success. For example, `imethod = 1` might be the method of moments, and `imethod = 2` might be another method. If no value of `imethod` works then it will be necessary to use arguments such as `isigma`. For many **VGAM** family functions it is advisable to try this argument with all possible values to safeguard against problems such as converging to a local solution. **VGAM** family functions with this argument usually correspond to a model or distribution that is relatively hard to fit successfully, therefore care is needed to ensure the global solution is obtained. So using all possible values that this argument supplies is a good idea.

**type.fitted**

Character. Type of fitted value returned by the `fitted()` methods function. The first choice is always the default. The available choices depends on what kind of family function it is. Using the first few letters of the chosen choice is okay. See `fittedvlm` for more details.

**probs.x, probs.y**

Numeric, with values in (0, 1). The probabilities that define quantiles with respect to some vector, usually an x or y of some sort. This is used to create
two subsets of data corresponding to ‘low’ and ‘high’ values of \( x \) or \( y \). Each value is separately fed into the \( \text{probs} \) argument of \texttt{quantile}. If the data set size is small then it may be necessary to increase/decrease slightly the first/second values respectively.

\textbf{\texttt{lss}}

Logical. This stands for the ordering: location, scale and shape. Should the ordering of the parameters be in this order? Almost all \texttt{VGAM} family functions have this order by default, but in order to match the arguments of existing R functions, one might need to set \texttt{lss = FALSE}. For example, the arguments of \texttt{weibullR} are scale and shape, whereas \texttt{rweibull} are shape and scale. As a temporary measure (from \texttt{VGAM} 0.9-7 onwards but prior to version 1.0-0), some family functions such as \texttt{sinmad} have an \texttt{lss} argument without a default. For these, setting \texttt{lss = FALSE} will work. Later, \texttt{lss = TRUE} will be the default. Be careful for the \texttt{dpqr}-type functions, e.g., \texttt{rsinmad}.

\textbf{\texttt{whitespace}}

Logical. Should white spaces (" ") be used in the labelling of the linear/additive predictors? Setting \texttt{TRUE} usually results in more readability but it occupies more columns of the output.

\textbf{\texttt{oim}}

Logical. Should the observed information matrices (OIMs) be used for the working weights? In general, setting \texttt{oim = TRUE} means the Newton-Raphson algorithm, and \texttt{oim = FALSE} means Fisher-scoring. The latter uses the EIM, and is usually recommended. If \texttt{oim = TRUE} then \texttt{nseim} is ignored.

\textbf{\texttt{zero}}

An integer specifying which linear/additive predictor is modelled as intercept-only. That is, the regression coefficients are set to zero for all covariates except for the intercept. If \texttt{zero} is specified then it may be a vector with values from the set \( \{1, 2, \ldots, M \} \). The value \texttt{zero = NULL} means model all linear/additive predictors as functions of the explanatory variables. Here, \( M \) is the number of linear/additive predictors. Technically, if \texttt{zero} contains the value \( j \) then the \( j \)th row of every constraint matrix (except for the intercept) consists of all 0 values. Some \texttt{VGAM} family functions allow the \texttt{zero} argument to accept negative values; if so then its absolute value is recycled over each (usual) response. For example, \texttt{zero = -2} for the two-parameter negative binomial distribution would mean, for each response, the second linear/additive predictor is modelled as intercepts-only. That is, for all the \( k \) parameters in \texttt{negbinomial} (this \texttt{VGAM} family function can handle a matrix of responses).

Suppose \texttt{zero = zerovec} where \texttt{zerovec} is a vector of negative values. If \( G \) is the usual \( M \) value for a univariate response then the actual values for argument \texttt{zero} are all values in \( c(\text{abs(zerovec)}, G + \text{abs(zerovec)}, 2*G + \text{abs(zerovec)}, \ldots) \) lying in the integer range 1 to \( M \). For example, setting \texttt{zero = -c(2, 3)} for a matrix response of 4 columns with \texttt{zinbinomial} (which usually has \( G = M = 3 \) for a univariate response) would be equivalent to \texttt{zero = c(2, 3, 5, 6, 8, 9, 11, 12)}. This example has \( M = 12 \). Note that if \texttt{zerovec} contains negative values then their absolute values should be elements from the set \( 1:G \).

Note: \texttt{zero} may have positive and negative values, for example, setting \texttt{zero = c(-2, 3)} in the above example would be equivalent to \texttt{zero = c(2, 3, 5, 8, 11)}.

\textbf{\texttt{ishrinkage}}

Shrinkage factor \( s \) used for obtaining initial values. Numeric, between 0 and 1. In general, the formula used is something like \( s\mu + (1 - s)y \) where \( \mu \) is a measure of central tendency such as a weighted mean or median, and \( y \) is the response vector. For example, the initial values are slight perturbations of the
mean towards the actual data. For many types of models this method seems to work well and is often reasonably robust to outliers in the response. Often this argument is only used if the argument \texttt{imethod} is assigned a certain value.

\textbf{nointercept} \quad An integer-valued vector specifying which linear/additive predictors have no intercepts. Any values must be from the set \{1,2,\ldots, M\}. A value of \texttt{NULL} means no such constraints.

\textbf{multiple.responses} \quad Logical. Some \texttt{VGAM} family functions allow a multivariate or vector response. If so, then usually the response is a matrix with columns corresponding to the individual response variables. They are all fitted simultaneously. Arguments such as \texttt{parallel} may then be useful to allow for relationships between the regressions of each response variable. If \texttt{multiple.responses} = \texttt{TRUE} then sometimes the response is interpreted differently, e.g., \texttt{posbinomial} chooses the first column of a matrix response as success and combines the other columns as failure, but when \texttt{multiple.responses} = \texttt{TRUE} then each column of the response matrix is the number of successes and the \texttt{weights} argument is of the same dimension as the response and contains the number of trials.

\textbf{earg.link} \quad Sometimes the link argument can receive \texttt{earg}-type input, such as \texttt{quasibinomial} calling \texttt{binomial}. This argument should be generally ignored.

\textbf{bred} \quad Logical. Some \texttt{VGAM} family functions will allow bias-reduction based on the work by Kosmidis and Firth. Currently none are working yet!

\section*{Details}

Full details will be given in documentation yet to be written, at a later date!

\section*{Value}

An object of class "\texttt{vglmff}" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm} and \texttt{vgam}.

\section*{Warning}

The \texttt{zero} argument is supplied for convenience but conflicts can arise with other arguments, e.g., the \texttt{constraints} argument of \texttt{vglm} and \texttt{vgam}. See Example 5 below for an example. If not sure, use, e.g., \texttt{constraints(fit)} and \texttt{coef(fit, matrix = TRUE)} to check the result of a fit.

The arguments \texttt{zero} and \texttt{nointercept} can be inputted with values that fail. For example, \texttt{multinomial(zero = 2, nointercept = 1)} means the second linear/additive predictor is identically zero, which will cause a failure.

Be careful about the use of other potentially contradictory constraints, e.g., \texttt{multinomial(zero = 2, parallel = TRUE ~ x)}.

If in doubt, apply \texttt{constraints()} to the fitted object to check.

\texttt{VGAM} family functions with the \texttt{nsimeim} may have inaccurate working weight matrices. If so, then the standard errors of the regression coefficients may be inaccurate. Thus output from \texttt{summary(fit), vcov(fit), etc.} may be misleading.

Changes relating to the \texttt{codelss} argument have very important consequences and users must beware. Good programming style is to rely on the argument names and not on the order.
Note

See links regarding a major change in link functions, for version 0.9-0 and higher (released during the 2nd half of 2012).

Author(s)

T. W. Yee

References


See Also

links, vglmff-class, normal.vcm, multilogit.

Examples

```r
# Example 1
cumulative()
cumulative(link = "probit", reverse = TRUE, parallel = TRUE)

# Example 2
wdata <- data.frame(x2 = runif(nn <- 1000))
wdata <- transform(wdata,
                   y = rweibull(nn, shape = 2 + exp(1 + x2), scale = exp(-0.5)))
fit <- vglm(y ~ x2, weibullR(ishape = logoff(offset = -2), zero = 2), data = wdata)
coeff(fit, mat = TRUE)

# Example 3; multivariate (multiple) response
## Not run:
data <- data.frame(x = runif(nn <- 500))
data <- transform(ndata,
                    y1 = rnbinom(nn, mu = exp(3+x), size = exp(1)), # k is size
                    y2 = rnbinom(nn, mu = exp(2-x), size = exp(0)))
fit <- vglm(cbind(y1, y2) ~ x, negbinomial(zero = -2), data = ndata)
coeff(fit, matrix = TRUE)

## End(Not run)
# Example 4
## Not run:
# fit1 and fit2 are equivalent
fit1 <- vglm(ymatrix ~ x2 + x3 + x4 + x5,
             cumulative(parallel = FALSE ~ 1 + x3 + x5), data = cdata)
fit2 <- vglm(ymatrix ~ x2 + x3 + x4 + x5,
             cumulative(parallel = TRUE ~ x2 + x4), data = cdata)

## End(Not run)

# Example 5
udata <- data.frame(x2 = rnorm(nn <- 200))
```
concoef <- transform(udata,
  y1 = rnorm(nn, mean = 1 - 3*x2, sd = exp(1 + 0.2*x2)),
  y2 = rnorm(nn, mean = 1 - 3*x2, sd = exp(1)))
args(uninormal)
fit1 <- vglm(y1 ~ x2, uninormal, data = udata) # This is okay
fit2 <- vglm(y2 ~ x2, uninormal(zero = 2), data = udata) # This is okay

# This creates potential conflict
clist <- list("(Intercept)" = diag(2), "x2" = diag(2))
fit3 <- vglm(y2 ~ x2, uninormal(zero = 2), data = udata,
  constraints = clist) # Conflict!
coef(fit3, matrix = TRUE) # Shows that clist["x2"] was overwritten,
constraints(fit3) # i.e., 'zero' seems to override the 'constraints' arg

# Example 6 ('whitespace' argument)
pneumo <- transform(pneumo, let = log(exposure.time))
fit1 <- vglm(cbind(normal, mild, severe) ~ let,
  sratio(whitespace = FALSE, parallel = TRUE), data = pneumo)
fit2 <- vglm(cbind(normal, mild, severe) ~ let,
  sratio(whitespace = TRUE, parallel = TRUE), data = pneumo)
head(predict(fit1, 2)) # No white spaces
head(predict(fit2, 2)) # Uses white spaces

---

concoef  

**Extract Model Constrained/Canonical Coefficients**

**Description**

concoef is a generic function which extracts the constrained (canonical) coefficients from objects returned by certain modelling functions.

**Usage**

concoef(object, ...)

**Arguments**

- **object**  
  An object for which the extraction of canonical coefficients is meaningful.

- **...**  
  Other arguments fed into the specific methods function of the model.

**Details**

For constrained quadratic and ordination models, *canonical coefficients* are the elements of the \( C \) matrix used to form the latent variables. They are highly interpretable in ecology, and are looked at as weights or loadings.

They are also applicable for reduced-rank VGLMs.
Value

The value returned depends specifically on the methods function invoked.

Warning

concoef replaces ccoef; the latter is deprecated.

For QO models, there is a direct inverse relationship between the scaling of the latent variables (site scores) and the tolerances. One normalization is for the latent variables to have unit variance. Another normalization is for all the species’ tolerances to be unit (provided eq.tolerances is TRUE). These two normalizations cannot simultaneously hold in general. For rank R models with R > 1 it becomes more complicated because the latent variables are also uncorrelated. An important argument when fitting quadratic ordination models is whether eq.tolerances is TRUE or FALSE. See Yee (2004) for details.

Author(s)

Thomas W. Yee

References


See Also

concoef-method, concoef.qrrvglm, concoef.cao, coef.

Examples

```r
# Not run: set.seed(111) # This leads to the global solution
hspear[,1:6] <- scale(hspar[,1:6]) # Standardized environmental vars
pl <- conq(cbind(Alopecce, Alopume, Alopabfr, Arctlute, Arctperi,
                  Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
                  Trocterr, Zoraspin) -
        WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + RefiLux,
        fam = quasipoissonff, data = hspeder, Crowlpositive = FALSE)
concoef(p1)

# End(Not run)
```
**concoef-methods**

**Constrained (Canonical) Coefficients**

**Description**

`concoef` is a generic function used to return the constrained (canonical) coefficients of a constrained ordination model. The function invokes particular methods which depend on the class of the first argument.

**Methods**

- **object** The object from which the constrained coefficients are extracted.

**constraints**

**Constraint Matrices**

**Description**

Extractor function for the *constraint matrices* of objects in the *VGAM* package.

**Usage**

```r
constraints(object, ...)  
constraints.vlm(object, type = c("lm", "term"), all = TRUE, which, matrix.out = FALSE, colnames.arg = TRUE, ...)  
```

**Arguments**

- **object** Some *VGAM* object, for example, having class *vglmff-class*.
- **type** Character. Whether LM- or term-type constraints are to be returned. The number of such matrices returned is equal to `nvar(object, type = "lm")` and the number of terms, respectively.
- **all, which** If `all = FALSE` then `which` gives the integer index or a vector of logicals specifying the selection.
- **matrix.out** Logical. If `TRUE` then the constraint matrices are `cbind()`ed together. The result is usually more compact because the default is a list of constraint matrices.
- **colnames.arg** Logical. If `TRUE` then column names are assigned corresponding to the variables.
- **...** Other possible arguments such as `type`.  

Details

Constraint matrices describe the relationship of coefficients/component functions of a particular explanatory variable between the linear/additive predictors in VGLM/VGAM models. For example, they may be all different (constraint matrix is the identity matrix) or all the same (constraint matrix has one column and has unit values).

VGLMs and VGAMs have constraint matrices which are known. The class of RR-VGLMs have constraint matrices which are unknown and are to be estimated.

Value

The extractor function `constraints()` returns a list comprising of constraint matrices—usually one for each column of the VLM model matrix, and in that order. The list is labelled with the variable names. Each constraint matrix has \( M \) rows, where \( M \) is the number of linear/additive predictors, and whose rank is equal to the number of columns. A model with no constraints at all has an order \( M \) identity matrix as each variable’s constraint matrix.

For `vglm` and `vgam` objects, feeding in `type = "term"` constraint matrices back into the same model should work and give an identical model. The default are the "lm"-type constraint matrices; this is a list with one constraint matrix per column of the LM matrix. See the `constraints` argument of `vglm`, and the example below.

Note

In all VGAM family functions `zero = NULL` means none of the linear/additive predictors are modelled as intercepts-only. Other arguments found in certain VGAM family functions which affect constraint matrices include `parallel` and `exchangeable`.

The `constraints` argument in `vglm` and `vgam` allows constraint matrices to be inputted. If so, then `constraints(fit, type = "lm")` can be fed into the `constraints` argument of the same object to get the same model.

The `xij` argument does not affect constraint matrices; rather, it allows each row of the constraint matrix to be multiplied by a specified vector.

Author(s)

T. W. Yee

References


See Also

`is.parallel`, `is.zero`. VGLMs are described in `vglm-class`; RR-VGLMs are described in `rrvglm-class`. 
Arguments such as zero and parallel found in many VGAM family functions are a way of creating/modifying constraint matrices conveniently, e.g., see zero. See CommonVGAMffArguments for more information.

Examples

# Fit the proportional odds model:
{r}
pneumo <- transform(pneumo, let = log(exposure.time))
(fit1 <- vglm(cbind(normal, mild, severe) ~ sm.bs(let, 3),
              cumulative(parallel = TRUE, reverse = TRUE), data = pneumo))
coef(fit1, matrix = TRUE)
constraints(fit1)  # Parallel assumption results in this
constraints(fit1, type = "term")  # Same as the default ("vlm"-type)
is.parallel(fit1)

# An equivalent model to fit1 (needs the type "term" constraints):
clist.term <- constraints(fit1, type = "term")  # "term"-type constraints
(fit2 <- vglm(cbind(normal, mild, severe) ~ sm.bs(let, 3), data = pneumo,
              cumulative(reverse = TRUE), constraints = clist.term))
abs(max(coef(fit1, matrix = TRUE) -
       coef(fit2, matrix = TRUE)))  # Should be zero

# Fit a rank-1 stereotype (RR-multinomial logit) model:
fit <- rrvglm(Country ~ Width + Height + HP, multinomial, data = car.all)
constraints(fit)  # All except the first are the estimated A matrix

Corbet's Butterfly Data

About 3300 individual butterflies were caught in Malaya by naturalist Corbet trapping butterflies. They were classified to about 500 species.

Usage

data(corbet)

Format

A data frame with 24 observations on the following 2 variables.

species  Number of species.
ofreq  Observed frequency of individual butterflies of that species.

Details

In the early 1940s Corbet spent two years trapping butterflies in Malaya. Of interest was the total number of species. Some species were so rare (e.g., 118 species had only one specimen) that it was thought likely that there were many unknown species.
References


Examples

summary(corbet)

Fitting Constrained Quadratic Ordination (CQO)

Description

A constrained quadratic ordination (CQO; formerly called canonical Gaussian ordination or CGO) model is fitted using the quadratic reduced-rank vector generalized linear model (QRR-VGLM) framework.

Usage

cqo(formula, family = list(), weights = NULL, subset = NULL,
na.action = na.fail, etastart = NULL, mustart = NULL,
coefstart = NULL, control = qrrvglm.control(...), offset = NULL,
method = "cqo.fit", model = FALSE, x.arg = TRUE, y.arg = TRUE,
contrasts = NULL, constraints = NULL, extra = NULL,
smart = TRUE, ...)

Arguments

formulaa symbolic description of the model to be fit. The RHS of the formula is applied to each linear predictor. Different variables in each linear predictor can be chosen by specifying constraint matrices.

familya function of class "vglmff" (see vglmff-class) describing what statistical model is to be fitted. This is called a "VGAM family function". See CommonVGAMffArguments for general information about many types of arguments found in this type of function. Currently the following families are supported: poissonff, binomialff (logit and cloglog links available), negbinomial, gammaR, gaussianff. Sometimes special arguments are required for cqo(), e.g., binomialff(multiple.responses = TRUE). Also, quasipoissonff and quasibinomialff may or may not work.

dataan optional data frame containing the variables in the model. By default the variables are taken from environment(formula), typically the environment from which cqo is called.

weightsan optional vector or matrix of (prior) weights to be used in the fitting process. Currently, this argument should not be used.

subsetan optional logical vector specifying a subset of observations to be used in the fitting process.
na.action  a function which indicates what should happen when the data contain NAs. The
     default is set by the na.action setting of options, and is na.fail if that is
     unset. The “factory-fresh” default is na.omit.

etastart  starting values for the linear predictors. It is a M-column matrix. If M = 1 then
     it may be a vector. Currently, this argument probably should not be used.

mustart  starting values for the fitted values. It can be a vector or a matrix. Some family
     functions do not make use of this argument. Currently, this argument probably
     should not be used.

coefstart  starting values for the coefficient vector. Currently, this argument probably
     should not be used.

control  a list of parameters for controlling the fitting process. See qrrvglm.control
     for details.

offset  This argument must not be used.

method  the method to be used in fitting the model. The default (and presently only)
     method qrrvglm.fit uses iteratively reweighted least squares (IRLS).

model  a logical value indicating whether the model frame should be assigned in the
     model slot.

x.arg, y.arg  logical values indicating whether the model matrix and response matrix used in
     the fitting process should be assigned in the x and y slots. Note the model matrix
     is the LM model matrix.

contrasts  an optional list. See the contrasts.arg of model.matrix.default.

constraints  an optional list of constraint matrices. The components of the list must be named
     with the term it corresponds to (and it must match in character format). Each
     constraint matrix must have M rows, and be of full-column rank. By default,
     constraint matrices are the M by M identity matrix unless arguments in the
     family function itself override these values. If constraints is used it must
     contain all the terms; an incomplete list is not accepted. Constraint matrices for
     x2 variables are taken as the identity matrix.

extra  an optional list with any extra information that might be needed by the family
     function.

smart  logical value indicating whether smart prediction (smartpred) will be used.

...  further arguments passed into qrrvglm.control.

Details

QRR-VGLMs or constrained quadratic ordination (CQO) models are estimated here by maximum
likelihood estimation. Optimal linear combinations of the environmental variables are computed,
called latent variables (these appear as latvar for R = 1 else latvar1, latvar2, etc. in the
output). Here, R is the rank or the number of ordination axes. Each species’ response is then a
regression of these latent variables using quadratic polynomials on a transformed scale (e.g., log for
Poisson counts, logit for presence/absence responses). The solution is obtained iteratively in order
to maximize the log-likelihood function, or equivalently, minimize the deviance.

The central formula (for Poisson and binomial species data) is given by
\[ \eta = B^T x_1 + A\nu + \sum_{m=1}^{M} (\nu^T D_m \nu) e_m \]
where $x_1$ is a vector (usually just a 1 for an intercept), $x_2$ is a vector of environmental variables, $\nu = C^T x_2$ is a $R$-vector of latent variables, $e_m$ is a vector of 0s but with a 1 in the $m$th position. The $\eta$ are a vector of linear/additive predictors, e.g., the $m$th element is $\eta_m = \log(E[Y_m])$ for the $m$th species. The matrices $B_1, A, C$ and $D_m$ are estimated from the data, i.e., contain the regression coefficients. The tolerance matrices satisfy $T_s = -\frac{1}{2} D_s^{-1}$. Many important CQO details are directly related to arguments in `qrrvglm.control`, e.g., the argument `norrr` specifies which variables comprise $x_1$.

Theoretically, the four most popular VGAM family functions to be used with c$qo$ correspond to the Poisson, binomial, normal, and negative binomial distributions. The latter is a 2-parameter model. All of these are implemented, as well as the 2-parameter gamma. The Poisson is or should be catered for by `quasipoissonff` and `poissonff`, and the binomial by `quasibinomialff` and `binomialff`. Those beginning with "quasi" have dispersion parameters that are estimated for each species.

For initial values, the function `.Init.Poisson.QO` should work reasonably well if the data is Poisson with species having equal tolerances. It can be quite good on binary data too. Otherwise the `Cinit` argument in `qrrvglm.control` can be used.

It is possible to relax the quadratic form to an additive model. The result is a data-driven approach rather than a model-driven approach, so that CQO is extended to constrained additive ordination (CAO) when $R = 1$. See `cao` for more details.

In this documentation, $M$ is the number of linear predictors, $S$ is the number of responses (species). Then $M = S$ for Poisson and binomial species data, and $M = 2S$ for negative binomial and gamma distributed species data.

Incidentally, Unconstrained quadratic ordination (UQO) may be performed by, e.g., fitting a Goodman’s RC association model; see `uqo` and the Yee and Hadi (2014) referenced there. For UQO, the response is the usual site-by-species matrix and there are no environmental variables; the site scores are free parameters. UQO can be performed under the assumption that all species have the same tolerance matrices.

**Value**

An object of class "`qrrvglm`".

**Warning**

Local solutions are not uncommon when fitting CQO models. To increase the chances of obtaining the global solution, increase the value of the argument `Bestof` in `qrrvglm.control`. For reproducibility of the results, it pays to set a different random number seed before calling c$qo$ (the function `set.seed` does this). The function `c$qo` chooses initial values for $C$ using `.Init.Poisson.QO()` if `Use.Init.Poisson.QO` = TRUE, else random numbers.

Unless `i.tolerances` = TRUE or `eq.tolerances` = FALSE, CQO is computationally expensive with memory and time. It pays to keep the rank down to 1 or 2. If `eq.tolerances` = TRUE and `i.tolerances` = FALSE then the cost grows quickly with the number of species and sites (in terms of memory requirements and time). The data needs to conform quite closely to the statistical model, and the environmental range of the data should be wide in order for the quadratics to fit the data well (bell-shaped response surfaces). If not, RR-VGLMs will be more appropriate because the response is linear on the transformed scale (e.g., log or logit) and the ordination is called constrained linear ordination or CLO.
Like many regression models, CQO is sensitive to outliers (in the environmental and species data), sparse data, high leverage points, multicollinearity etc. For these reasons, it is necessary to examine the data carefully for these features and take corrective action (e.g., omitting certain species, sites, environmental variables from the analysis, transforming certain environmental variables, etc.). Any optimum lying outside the convex hull of the site scores should not be trusted. Fitting a CAO is recommended first, then upon transformations etc., possibly a CQO can be fitted.

For binary data, it is necessary to have 'enough' data. In general, the number of sites $n$ ought to be much larger than the number of species $S$, e.g., at least 100 sites for two species. Compared to count (Poisson) data, numerical problems occur more frequently with presence/absence (binary) data. For example, if Rank $= 1$ and if the response data for each species is a string of all absences, then all presences, then all absences (when enumerated along the latent variable) then infinite parameter estimates will occur. In general, setting `iNtolerances = TRUE` may help.

This function was formerly called `cgo`. It has been renamed to reinforce a new nomenclature described in Yee (2006).

**Note**

The input requires care, preparation and thought—a lot more than other ordination methods. Here is a partial checklist.

1. The number of species should be kept reasonably low, e.g., 12 max. Feeding in 100+ species wholesale is a recipe for failure. Choose a few species carefully. Using 10 well-chosen species is better than 100+ species thrown in willy-nilly.

2. Each species should be screened individually first, e.g., for presence/absence is the species totally absent or totally present at all sites? For presence/absence data `sort(colMeans(data))` can help avoid such species.

3. The number of explanatory variables should be kept low, e.g., 7 max.

4. Each explanatory variable should be screened individually first, e.g., is it heavily skewed or are there outliers? They should be plotted and then transformed where needed. They should not be too highly correlated with each other.

5. Each explanatory variable should be scaled, e.g., to mean 0 and unit variance. This is especially needed for `iNtolerance = TRUE`.

6. Keep the rank low. Only if the data is very good should a rank-2 model be attempted. Usually a rank-1 model is all that is practically possible even after a lot of work. The rank-1 model should always be attempted first. Then might be clever and try use this for initial values for a rank-2 model.

7. If the number of sites is large then choose a random sample of them. For example, choose a maximum of 500 sites. This will reduce the memory and time expense of the computations.

8. Try `iNtolerance = TRUE` or `eq.tolerance = FALSE` if the inputted data set is large, so as to reduce the computational expense. That's because the default, `iNtolerance = FALSE` and `eq.tolerance = TRUE`, is very memory hungry.

By default, a rank-1 equal-tolerances QRR-VGLM model is fitted (see `qrrvglm.control` for the default control parameters). If `Rank > 1` then the latent variables are always transformed so that they are uncorrelated. By default, the argument `trace` is `TRUE` meaning a running log is printed out while the computations are taking place. This is because the algorithm is computationally expensive, therefore users might think that their computers have frozen if `trace = FALSE`!
The argument `bestof` in `qrrvglm.control` controls the number of models fitted (each uses different starting values) to the data. This argument is important because convergence may be to a local solution rather than the global solution. Using more starting values increases the chances of finding the global solution. Always plot an ordination diagram (use the generic function `lvplot`) and see if it looks sensible. Local solutions arise because the optimization problem is highly nonlinear, and this is particularly true for CAO.

Many of the arguments applicable to `cqo` are common to `vglm` and `qrrvglm.control`. The most important arguments are `Rank`, `noRRR`, `Bestof`, `I.tolerances`, `eq.tolerances`, `isd.latvar`, and `MUXfactor`. When fitting a 2-parameter model such as the negative binomial or gamma, it pays to have `eq.tolerances = TRUE` and `I.tolerances = FALSE`. This is because numerical problems can occur when fitting the model far away from the global solution when `I.tolerances = TRUE`. Setting the two arguments as described will slow down the computation considerably, however it is numerically more stable.

In Example 1 below, an unequal-tolerances rank-1 QRR-VGLM is fitted to the hunting spiders dataset, and Example 2 is the equal-tolerances version. The latter is less likely to have convergence problems compared to the unequal-tolerances model. In Example 3 below, an equal-tolerances rank-2 QRR-VGLM is fitted to the hunting spiders dataset. The numerical difficulties encountered in fitting the rank-2 model suggests a rank-1 model is probably preferable. In Example 4 below, constrained binary quadratic ordination (in old nomenclature, constrained Gaussian logit ordination) is fitted to some simulated data coming from a species packing model. With multivariate binary responses, one must use `multiple.responses = TRUE` to indicate that the response (matrix) is multivariate. Otherwise, it is interpreted as a single binary response variable. In Example 5 below, the deviance residuals are plotted for each species. This is useful as a diagnostic plot. This is done by (re)regressing each species separately against the latent variable.

Sometime in the future, this function might handle input of the form `cqo(x, y)`, where `x` and `y` are matrices containing the environmental and species data respectively.

**Author(s)**

Thomas W. Yee. Thanks to Alvin Sou for converting a lot of the original FORTRAN code into C.

**References**


**See Also**

`qrrvglm.control, Coef.qrrvglm, predictqrrvglm, rcqo, cao, uqo, rrvglm, poissonff, binomialff, negbinomial, gamma2, lvplot.qrrvglm, perspqrrvglm, trplot.qrrvglm, vglm, set.seed, hspider, trapO`. 
Examples

```r
## Not run:
## Example 1; Fit an unequal tolerances model to the hunting spiders data
hsprise[1:6] <- cbind(hspider[,1:6]) # Standardized environmental variables
set.seed(1234) # For reproducibility of the results
plut <- cpo(setseed(Alopace, Alopume, Alopabr, Arctluve, Arctperi,
Auloalbi, Pardlugru, Pardmont, Pardnigr, Pardpull,
Trocterr, Zoraspin) ~
WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + RefLux,
fm = poissonff, data = hspider, Crowapositive = FALSE,
eq.tol = FALSE)
sort(deviance(plut, history = TRUE)) # A history of all the iterations
if (deviance(plut) > 1777) warning("suboptimal fit obtained")
S <- ncol(depvar(plut)) # Number of species
col <- -(1:(S+1))[-7] # Omits yellow
lvplot(plut, y = TRUE, lcol = clor, pch = 1:S, pcol = clor,
las = 1) # Ordination diagram
legend("topright", leg = colnames(depvar(plut)), col = clor,
pch = 1:S, merge = TRUE, bty = "n", lty = 1:S, lwd = 2)
(cp <- Coef(plut))
(a <- latvar(cp)[cp@latvar.order]) # Ordered site scores along the gradient
# Names of the ordered sites along the gradient:
rownames(latvar(cp)[cp@latvar.order])
(aa <- Opt(cp)[, cp@optimum.order]) # Ordered optimums along the gradient
aa <- aa[is.na(aa)] # Delete the species that is not unimodal
names(aa) # Names of the ordered optimums along the gradient
trplot(plut, which.species = 1:3, log = "xy", type = "b", lty = 1, lwd = 2,
col = c("blue"", "red", "green"), label = TRUE) -> ii # Trajectory plot
legend(0.00005, 0.3, paste(iispecies[1], iispecies[2], sep = " and "),
lwd = 2, lty = 1, col = c("blue", "red", "green"))
abline(a = 0, b = 1, lty = "dashed")
S <- ncol(depvar(plut)) # Number of species
col <- -(1:(S+1))[-7] # Omits yellow
persp(plut, col = clor, label = TRUE, las = 1) # Perspective plot

## Example 2; Fit an equal tolerances model. Less numerically fraught.
set.seed(1234)
plet <- cpo(setseed(Alopace, Alopume, Alopabr, Arctluve, Arctperi,
Auloalbi, Pardlugru, Pardmont, Pardnigr, Pardpull,
Trocterr, Zoraspin) ~
WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + RefLux,
poissonff, data = hspider, Crowapositive = FALSE)
sort(deviance(plet, history = TRUE)) # A history of all the iterations
if (deviance(plet) > 1586) warning("suboptimal fit obtained")
S <- ncol(depvar(plet)) # Number of species
col <- -(1:(S+1))[-7] # Omits yellow
persp(plet, col = clor, label = TRUE, las = 1)
```

# Example 3: A rank-2 equal tolerances CQO model with Poisson data
# This example is numerically fraught... need I.toler = TRUE too.
set.seed(555)
p2 <- cko(cbind(Alopacce, Alocpune, Alopfabr, Arctlute, Arctperi,
               Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
               Trocterr, Zoraspin) ~
              WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + RefiLux,
              poissonff, data = hsipider, Crowlpositive = FALSE,
              I.toler = TRUE, Rank = 2, Bestof = 3, isd.latvar = c(2.1, 0.9))
sort(deviance(p2, history = TRUE))  # A history of all the iterations
if (deviance(p2) > 1127) warning("suboptimal fit obtained")
LVplot(p2, ellips = FALSE, label = TRUE, xlim = c(-3, 4),
       C = TRUE, Ccol = "brown", sites = TRUE, scol = "grey",
       pcol = "blue", pch = "+", chull = TRUE, ccol = "grey")

# Example 4: Species packing model with presence/absence data
set.seed(2345)
n <- 200; p <- 5; S <- 5
mydata <- rko(n, p, S, fam = "binomial", hi.abundance = 4,
               eq.tol = TRUE, es.opt = TRUE, eq.max = TRUE)
myform <- attr(mydata, "formula")
set.seed(1234)
bet <- cko(myform, binomialff(multiple.responses = TRUE, link = "cloglog"),
            data = mydata)
sort(deviance(bet, history = TRUE))  # A history of all the iterations
LVplot(bet, y = TRUE, lcol = 1:S, pch = 1:S, pcol = 1:S, las = 1)
Coef(bet)

# Compare the fitted model with the 'truth'
cbind(truth = attr(mydata, "concoefficient"), fitted = concoef(bet))

# Example 5: Plot the deviance residuals for diagnostic purposes
set.seed(1234)
plet <- cko(cbind(Alopacce, Alocpune, Alopfabr, Arctlute, Arctperi,
                  Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
                  Trocterr, Zoraspin) ~
                 WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + RefiLux,
                 poissonff, data = hsipider, eq.tol = TRUE, trace = FALSE)
sort(deviance(plet, history = TRUE))  # A history of all the iterations
if (deviance(plet) > 1586) warning("suboptimal fit obtained")
S <- ncol(depvar(plet))
par(mfrow = c(3, 4))
for (ii in 1:S) {
  tempdata <- data.frame(latvar1 = c(latvar(plet)),
                          sppCounts = depvar(plet)[, ii])
  tempdata <- transform(tempdata, myOffset = -0.5 * latvar1^2)
  # For species ii, refit the model to get the deviance residuals
  fiti <- vglm(sppCounts ~ offset(myOffset) + latvar1, poissonff,
data = tempdata, trace = FALSE)

# For checking: this should be 0
# print("max(abs((Coef(p1)@B1[1,ii],Coef(p1)@A[ii,1])-coef(fit1)))")
# print( max(abs((Coef(p1)@B1[1,ii],Coef(p1)@A[ii,1])-coef(fit1))) )

# Plot the deviance residuals
devresid <- resid(fit1, type = "deviance")
predvalues <- predict(fit1) + fit1@offset
ooo <- with(tempdata, order(latvar1))
plot(predvalues + devresid ~ latvar1, data = tempdata, col = "red",
xlab = "latvar1", ylab = "", main = colnames(depvar(p1))[ii])
with(tempdata, lines(latvar1[ooo], predvalues[ooo], col = "blue"))

## End(Not run)

---

### Crashes on New Zealand Roads in 2009

**Description**

A variety of reported crash data cross-classified by time (hour of the day) and day of the week, accumulated over 2009. These include fatalities and injuries (by car), trucks, motor cycles, bicycles and pedestrians. There are some alcohol-related data too.

**Usage**

data(crashi)
data(crashf)
data(crashtr)
data(crashmc)
data(crashbc)
data(crashp)
data(alcoff)
data(alclevels)

**Format**

Data frames with hourly times as rows and days of the week as columns. The alclevels dataset has hourly times and alcohol levels.

**Mon, Tue, Wed, Thu, Fri, Sat, Sun**   Day of the week.

**0-30, 31-50, 51-80, 81-100, 101-120, 121-150, 151-200, 201-250, 251-300, 301-350, 350+**   Blood alcohol level (milligrams alcohol per 100 millilitres of blood).
Details

Each cell is the aggregate number of crashes reported at each hour-day combination, over the 2009 calendar year. The rownames of each data frame is the start time (hourly from midnight onwards) on a 24 hour clock, e.g., 21 means 9.00pm to 9.59pm.

For crashes, chrashi are the number of injuries by car, crashf are the number of fatalities by car (not included in chrashi), crashtr are the number of crashes involving trucks, crashmc are the number of crashes involving motorcyclists, crashbc are the number of crashes involving bicycles, and crashp are the number of crashes involving pedestrians. For alcohol-related offences, alcoff are the number of alcohol offenders from breath screening drivers, and alclevels are the blood alcohol levels of fatally injured drivers.

Source


References


See Also

rrvglm, rcim, grc.

Examples

```r
## Not run:  plot(unlist(alcoff), type = "l", frame.plot = TRUE,
axises = FALSE, col = "blue", bty = "o",
main = "Alcoholic offenders on NZ roads, aggregated over 2009",
sub = "Vertical lines at midnight (purple) and noon (orange)",
xlab = "Day/hour", ylab = "Number of offenders")
axis(1, at = 1 + (0:6) * 24 + 12, labels = colnames(alcoff))
axis(2, las = 1)
axis(3:4, labels = FALSE, tick = FALSE)
abline(v = sort(1 + c((0:7) * 24, (0:6) * 24 + 12)), lty = "dashed",
col = c("purple", "orange"))
## End(Not run)

# Goodmans RC models
## Not run:
fitgrc1 <- grc(alcoff)  # Rank-1 model
fitgrc2 <- grc(alcoff, Rank = 2, Corner = FALSE, Uncor = TRUE)
Coef(fitgrc2)

## End(Not run)
## Not run:  biplot(fitgrc2, scaleA = 2.3, Ccol = "blue", Acol = "orange",
Clabels = as.character(1:23), xlim = c(-1.3, 2.3),
ylim = c(-1.2, 1))
## End(Not run)
```
Ordinal Regression with Continuation Ratios

Description

Fits a continuation ratio logit/probit/cloglog/cauchit/... regression model to an ordered (preferably) factor response.

Usage

```
cratio(link = "logit", parallel = FALSE, reverse = FALSE, zero = NULL, whitespace = FALSE)
```

Arguments

- **link**: Link function applied to the \(M\) continuation ratio probabilities. See Links for more choices.
- **parallel**: A logical, or formula specifying which terms have equal/unequal coefficients.
- **reverse**: Logical. By default, the continuation ratios used are \(\eta_j = \logit(P[Y > j | Y \geq j])\) for \(j = 1, \ldots, M\). If reverse is TRUE, then \(\eta_j = \logit(P[Y < j + 1 | Y \leq j + 1])\) will be used.
- **zero**: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \{1,2,\ldots,M\}. The default value means none are modelled as intercept-only terms.
- **whitespace**: See CommonVGAMffArguments for information.

Details

In this help file the response \(Y\) is assumed to be a factor with ordered values 1, 2, \ldots, \(M + 1\), so that \(M\) is the number of linear/additive predictors \(\eta_j\).

There are a number of definitions for the continuation ratio in the literature. To make life easier, in the VGAM package, we use continuation ratios and stopping ratios (see sratio). Stopping ratios deal with quantities such as \(\logit(P[Y=j|Y>\cdot])\).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

No check is made to verify that the response is ordinal if the response is a matrix; see ordered.
Note

The response should be either a matrix of counts (with row sums that are all positive), or a factor. In both cases, the y slot returned by vglm/vgam/rrvglm is the matrix of counts.

For a nominal (unordered) factor response, the multinomial logit model (multinomial) is more appropriate.

Here is an example of the usage of the parallel argument. If there are covariates x1, x2 and x3, then parallel = TRUE ~ x1 + x2 -1 and parallel = FALSE ~ x3 are equivalent. This would constrain the regression coefficients for x1 and x2 to be equal; those of the intercepts and x3 would be different.

Author(s)

Thomas W. Yee

References


See Also

sratio, acat, cumulative, multinomial, pneumo, logit, probit, cloglog, cauchit.

Examples

```r
pneumo <- transform(pneumo, let = log(exposure.time))
(fit <- vglm(cbind(normal, mild, severe) ~ let,
             cratio(parallel = TRUE), data = pneumo))
coef(fit, matrix = TRUE)
constraints(fit)
predict(fit)
predict(fit, untransform = TRUE)
```

---

**cumulative**

**Ordinal Regression with Cumulative Probabilities**

Description

Fits a cumulative link regression model to a (preferably ordered) factor response.
Usage

cumulative(link = "logit", parallel = FALSE, reverse = FALSE, multiple.responses = FALSE, whitespace = FALSE)

Arguments

link Link function applied to the \( J \) cumulative probabilities. See Links for more choices, e.g., for the cumulative probit/cloglog/cauchit/... models.

parallel A logical or formula specifying which terms have equal/unequal coefficients. See below for more information about the parallelism assumption. The default results in what some people call the generalized ordered logit model to be fitted. If parallel = TRUE then it does not apply to the intercept.

reverse Logical. By default, the cumulative probabilities used are \( P(Y \leq 1) \), \( P(Y \leq 2) \), ..., \( P(Y \leq J) \). If reverse is TRUE then \( P(Y \geq 2) \), \( P(Y \geq 3) \), ..., \( P(Y \geq J + 1) \) are used. This should be set to TRUE for link = golf, polf, nbolf. For these links the cutpoints must be an increasing sequence; if reverse = FALSE for then the cutpoints must be an decreasing sequence.

multiple.responses Logical. Multiple responses? If TRUE then the input should be a matrix with values 1, 2, ..., \( L \), where \( L = J + 1 \) is the number of levels. Each column of the matrix is a response, i.e., multivariate response. A suitable matrix can be obtained from Cut.

whitespace See CommonVGAMArguments for information.

Details

In this help file the response \( Y \) is assumed to be a factor with ordered values 1, 2, ..., \( J + 1 \). Hence \( M \) is the number of linear/additive predictors \( \eta_j \); for cumulative() one has \( M = J \).

This VGAM family function fits the class of cumulative link models to (hopefully) an ordinal response. By default, the non-parallel cumulative logit model is fitted, i.e.,

\[
\eta_j = \logit(P[Y \leq j])
\]

where \( j = 1, 2, \ldots, M \) and the \( \eta_j \) are not constrained to be parallel. This is also known as the non-proportional odds model. If the logit link is replaced by a complementary log-log link (cloglog) then this is known as the proportional-hazards model.

In almost all the literature, the constraint matrices associated with this family of models are known. For example, setting parallel = TRUE will make all constraint matrices (except for the intercept) equal to a vector of \( M \) 1’s. If the constraint matrices are equal, unknown and to be estimated, then this can be achieved by fitting the model as a reduced-rank vector generalized linear model (RR-VGLM; see rrvglm). Currently, reduced-rank vector generalized additive models (RR-VGAMs) have not been implemented here.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.
Warning

No check is made to verify that the response is ordinal if the response is a matrix; see ordered.

Note

The response should be either a matrix of counts (with row sums that are all positive), or a factor. In both cases, the y slot returned by vglm/vgam/rrvglm is the matrix of counts. The formula must contain an intercept term. Other VGAM family functions for an ordinal response include acat, cratio, sratio. For a nominal (unordered) factor response, the multinomial logit model (multinomial) is more appropriate.

With the logit link, setting parallel = TRUE will fit a proportional odds model. Note that the TRUE here does not apply to the intercept term. In practice, the validity of the proportional odds assumption needs to be checked, e.g., by a likelihood ratio test (LRT). If acceptable on the data, then numerical problems are less likely to occur during the fitting, and there are less parameters. Numerical problems occur when the linear/additive predictors cross, which results in probabilities outside of (0, 1); setting parallel = TRUE will help avoid this problem.

Here is an example of the usage of the parallel argument. If there are covariates x2, x3 and x4, then parallel = TRUE ~ x2 + x3 -1 and parallel = FALSE ~ x4 are equivalent. This would constrain the regression coefficients for x2 and x3 to be equal; those of the intercepts and x4 would be different.

If the data is inputted in long format (not wide format, as in pneumo below) and the self-starting initial values are not good enough then try using mustart, coefstart and/or etatstart. See the example below.

To fit the proportional odds model one can use the VGAM family function propodds. Note that propodds(reverse) is equivalent to cumulative(parallel = TRUE, reverse = reverse) (which is equivalent to cumulative(parallel = TRUE, reverse = reverse, link = "logit")). It is for convenience only. A call to cumulative() is preferred since it reminds the user that a parallelism assumption is made, as well as being a lot more flexible.

Author(s)

Thomas W. Yee

References

See Also

propodds, prplot, margeff, acat, cratio, sratio, multinomial, pneumo, Links, logit, probit, cloglog, cauchit, golf, polf, nbolf, logistic1.

Examples

# Fit the proportional odds model, p.179, in McCullagh and Nelder (1989)

pneumo <- transform(pneumo, let = log(exposure.time))

(fit <- vglm(cbind(normal, mild, severe) ~ let,
            cumulative(parallel = TRUE, reverse = TRUE), data = pneumo))

depvar(fit) # Sample proportions (good technique)
fit@y # Sample proportions (bad technique)
weights(fit, type = "prior") # Number of observations
coef(fit, matrix = TRUE)
constraints(fit) # Constraint matrices
apply(fitted(fit), 1, which.max) # Classification
apply(predict(fit, newdata = pneumo, type = "response"),
       1, which.max) # Classification

# Check that the model is linear in let -----------------------------

fit2 <- vglm(cbind(normal, mild, severe) ~ s(let, df = 2),
            cumulative(reverse = TRUE), data = pneumo)

# Not run: plot(fit2, se = TRUE, overlay = TRUE, lcol = 1:2, scol = 1:2)

# Check the proportional odds assumption with a LRT ----------

(fit3 <- vglm(cbind(normal, mild, severe) ~ let,
             cumulative(parallel = FALSE, reverse = TRUE), data = pneumo))

pchisq(2 * (logLik(fit3) - logLik(fit)),
       df = length(coef(fit3)) - length(coef(fit)), lower.tail = FALSE)

lrtest(fit3, fit) # More elegant

# A factor() version of fit ----------------------------------------
# This is in long format (cf. wide format above)

Nobs <- round(depvar(fit) * c(weights(fit, type = "prior")))

sumNobs <- colSums(Nobs) # apply(Nobs, 2, sum)

pneumo.long <-
  data.frame(sympoms = ordered(rep(colnames(Nobs), nrow(Nobs)),
                               times = c(t(Nobs))),
             levels = colnames(Nobs),
             let = rep(with(pneumo, let), each = ncol(Nobs)),
             times = c(t(Nobs))))

with(pneumo.long, table(let, sympoms)) # Should be same as pneumo

(fit.long1 <- vglm(sympoms ~ let, data = pneumo.long, trace = TRUE,
                   cumulative(parallel = TRUE, reverse = TRUE)))

coef(fit.long1, matrix = TRUE) # Should be as coef(fit, matrix = TRUE)

# Could try using mustart if fit.long1 failed to converge.

mymustart <- matrix(sumNobs / sum(sumNobs),
                     nrow(pneumo.long), ncol(Nobs), byrow = TRUE)

fit.long2 <- vglm(sympoms ~ let, mustart = mymustart,
Dagum

The Dagum Distribution

Description

Density, distribution function, quantile function and random generation for the Dagum distribution with shape parameters a and p, and scale parameter scale.

Usage

\[
\begin{align*}
\text{ddagum}(x, \text{scale} = 1, \text{shape1.a}, \text{shape2.p}, \text{log} = \text{FALSE}) \\
\text{pdagum}(q, \text{scale} = 1, \text{shape1.a}, \text{shape2.p}, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE}) \\
\text{qdagum}(p, \text{scale} = 1, \text{shape1.a}, \text{shape2.p}, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE}) \\
\text{rdagum}(n, \text{scale} = 1, \text{shape1.a}, \text{shape2.p})
\end{align*}
\]

Arguments

\(x, q\)
vector of quantiles.

\(p\)
vector of probabilities.

\(n\)
number of observations. If \(\text{length}(n) > 1\), the length is taken to be the number required.

\(\text{shape1.a}, \text{shape2.p}\)
shape parameters.

\(\text{scale}\)
scale parameter.

\(\text{log}\)
Logical. If \(\text{log} = \text{TRUE}\) then the logarithm of the density is returned.

\(\text{lower.tail}, \text{log.p}\)
Same meaning as in \text{pnorm} or \text{qnorm}.

Details

See \text{dagum}, which is the \text{VGAM} family function for estimating the parameters by maximum likelihood estimation.

Value

\text{ddagum} gives the density, \text{pdagum} gives the distribution function, \text{qdagum} gives the quantile function, and \text{rdagum} generates random deviates.

Note

The Dagum distribution is a special case of the 4-parameter generalized beta II distribution.
Author(s)

T. W. Yee and Kai Huang

References


See Also

dagum, genbetaII.

Examples

```r
probs <- seq(0.1, 0.9, by = 0.1)
shape1.a <- 1; shape2.p <- 2
# Should be 0:
max(abs(pdagum(qdagum(p = probs, shape1.a = shape1.a, shape2.p = shape2.p),
                shape1.a = shape1.a, shape2.p = shape2.p) - probs))

## Not run: par(mfrow = c(1, 2))
x <- seq(0.01, 5.1, len = 401)
plot(x, dexp(x), type = "l", col = "black", ylab = "", las = 1, ylim = c(0, 1),
     main = "Black is standard exponential, others are ddagum(x, ...")
lines(x, ddagum(x, shape1.a = shape1.a, shape2.p = 1), col = "orange")
lines(x, ddagum(x, shape1.a = shape1.a, shape2.p = 2), col = "blue")
lines(x, ddagum(x, shape1.a = shape1.a, shape2.p = 5), col = "green")
legend("topright", col = c("orange","blue","green"), lty = rep(1, len = 3),
       legend = paste("shape1.a =", shape1.a, ", shape2.p =", c(1, 2, 5)))

plot(x, pexp(x), type = "l", col = "black", ylab = "", las = 1,
     main = "Black is standard exponential, others are pdagum(x, ...")
lines(x, pdagum(x, shape1.a = shape1.a, shape2.p = 1), col = "orange")
lines(x, pdagum(x, shape1.a = shape1.a, shape2.p = 2), col = "blue")
lines(x, pdagum(x, shape1.a = shape1.a, shape2.p = 5), col = "green")
legend("bottomright", col = c("orange","blue","green"), lty = rep(1, len = 3),
       legend = paste("shape1.a =", shape1.a, ", shape2.p =", c(1, 2, 5)))

## End(Not run)
```

---

**Description**

Maximum likelihood estimation of the 3-parameter Dagum distribution.
Usage

dagum(lscale = "loge", lshape1.a = "loge", lshape2.p = "loge",
    iscale = NULL, ishape1.a = NULL, ishape2.p = NULL, imethod = 1,
    lss = TRUE, gscale = exp(-5:5), gshape1.a = exp(-5:5), gshape2.p = exp(-5:5),
    probs.y = c(0.25, 0.5, 0.75), zero = ifelse(lss, -(2:3), -c(1, 3)))

Arguments

lss See CommonVGAMffArguments for important information.
lshape1.a, lscale, lshape2.p
    Parameter link functions applied to the (positive) parameters a, scale, and p.
    See Links for more choices.
iscale, ishape1.a, ishape2.p, imethod, zero
    See CommonVGAMffArguments for information. For imethod = 2 a good initial
    value for ishape2.p is needed to obtain a good estimate for the other parameter.
gscale, gshape1.a, gshape2.p
    See CommonVGAMffArguments for information.
probs.y See CommonVGAMffArguments for information.

Details

The 3-parameter Dagum distribution is the 4-parameter generalized beta II distribution with shape parameter \( q = 1 \). It is known under various other names, such as the Burr III, inverse Burr, beta-K, and 3-parameter kappa distribution. It can be considered a generalized log-logistic distribution. Some distributions which are special cases of the 3-parameter Dagum are the inverse Lomax (\( a = 1 \)), Fisk (\( p = 1 \)), and the inverse paralogistic (\( a = p \)). More details can be found in Kleiber and Kotz (2003).

The Dagum distribution has a cumulative distribution function

\[
F(y) = [1 + (y/b)^{-a}]^{-p}
\]

which leads to a probability density function

\[
f(y) = apy^{ap-1}/[b^{ap}[1 + (y/b)^a]^{p+1}]
\]

for \( a > 0, b > 0, p > 0, y \geq 0 \). Here, \( b \) is the scale parameter scale, and the others are shape parameters. The mean is

\[
E(Y) = b \Gamma(p + 1/a) \Gamma(1 - 1/a)/\Gamma(p)
\]

provided \(-ap < 1 < a\); these are returned as the fitted values. This family function handles multiple responses.

Value

An object of class "vglmff" (see vglm-class). The object is used by modelling functions such as vglm, and vgam.
Note

See the notes in `genbetaII`.

From Kleiber and Kotz (2003), the MLE is rather sensitive to isolated observations located sufficiently far from the majority of the data. Reliable estimation of the scale parameter requires \( n > 7000 \), while estimates for \( a \) and \( p \) can be considered unbiased for \( n > 2000 \) or 3000.

Author(s)

T. W. Yee

References


See Also

`dagum`, `genbetaII`, `sinmad`, `fisk`, `inv.lomax`, `lomax`, `paralogistic`, `inv.paralogistic`, `simulate.vlm`.

Examples

```r
ddata <- data.frame(y = rdagum(n = 3000, scale = exp(2),
shape1 = exp(1), shape2 = exp(1)))
fit <- vglm(y ~ 1, dagum(lss = FALSE), data = ddata, trace = TRUE)
fit <- vglm(y ~ 1, dagum(lss = FALSE, ishape1.a = exp(1)),
data = ddata, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)
```

---

**dAR1**

*The AR-1 Autoregressive Process*

Description

Density for the AR-1 model.

Usage

```r
dAR1(x, drift = 0, var.error = 1, ARcoef1 = 0.0,
type.likelihood = c("exact", "conditional"), log = FALSE)
```
Arguments

- **x**, vector of quantiles.
- **drift**, the scaled mean (also known as the drift parameter), $\mu^*$. Note that the mean is $\mu^*/(1 - \rho)$. The default corresponds to observations that have mean 0.
- **log**, Logical. If TRUE then the logarithm of the density is returned.
- **type.likelihood**, var.error, ARcoeff

See AR1. The argument ARcoeff is $\rho$. The argument var.error is the variance of the i.i.d. random noise, i.e., $\sigma^2$. If type.likelihood = "conditional" then the first element or row of the result is currently assigned NA—this is because the density of the first observation is effectively ignored.

Details

Most of the background to this function is given in AR1. All the arguments are converted into matrices, and then all their dimensions are obtained. They are then coerced into the same size: the number of rows is the maximum of all the single rows, and ditto for the number of columns.

Value
dAR1 gives the density.

Author(s)

T. W. Yee and Victor Miranda

See Also

AR1.

Examples

```r
nn <- 100; set.seed(1)
tdata <- data.frame(index = 1:nn,
                    TS1 = arima.sim(nn, model = list(ar = -0.50),
                                   sd = exp(1)))
fit1 <- vglm(TS1 ~ 1, AR1, data = tdata, trace = TRUE)
                      rhobit(-0.5)
coef(fit1, matrix = TRUE)
(Cfit1 <- Coef(fit1))
summary(fit1)  # SEs are useful to know
logLik(fit1)
sum(dAR1(depvar(fit1), drift = Cfit1[1], var.error = (Cfit1[2])^2,
         ARcoeff1 = Cfit1[3], log = TRUE))

fit2 <- vglm(TS1 ~ 1, AR1(type.likelihood = "cond"), data = tdata, trace = TRUE)
(Cfit2 <- Coef(fit2))  # Okay for intercept-only models
logLik(fit2)
head(keep <- dAR1(depvar(fit2), drift = Cfit2[1], var.error = (Cfit2[2])^2,
                  ARcoeff1 = Cfit2[3], type.likelihood = "cond", log = TRUE))
sum(keep[-1])
```
Captures of *Peromyscus maniculatus*, also known as deer mice.

**Description**

Captures of *Peromyscus maniculatus* collected at East Stuart Gulch, Colorado, USA.

**Usage**

```r
data(deermice)
```

**Format**

The format is a data frame.

**Details**

*Peromyscus maniculatus* is a rodent native to North America. The deer mouse is small in size, only about 8 to 10 cm long, not counting the length of the tail.

Originally, the columns of this data frame represent the sex (m or f), the ages (y: young, sa: semi-adult, a: adult), the weights in grams, and the capture histories of 38 individuals over 6 trapping occasions (1: captured, 0: not captured).

The data set was collected by V. Reid and distributed with the **CAPTURE** program of Otis et al. (1978).

*deermice* has 38 deermice whereas *Perom* had 36 deermice (*Perom* has been withdrawn.) In *deermice* the two semi-adults have been classified as adults. The `sex` variable has 1 for female, and 0 for male.

**References**


**See Also**

`posbernoulli.b`, `posbernoulli.t`, `fill1`.

**Examples**

```r
head(deermice)
## Not run:
fit1 <- vglm(cbind(y1, y2, y3, y4, y5, y6) ~ sex + age,
             posbernoulli.t(parallel.t = TRUE), data = deermice, trace = TRUE)
coef(fit1)
coef(fit1, matrix = TRUE)
```
density plot for LMS quantile regression

Description

Plots a probability density function associated with a LMS quantile regression.

Usage

deplot.lmscreg(object, newdata = NULL, x0, y.arg, show.plot = TRUE, ...)

Arguments

object A VGAM quantile regression model, i.e., an object produced by modelling functions such as vglm and vgam with a family function beginning with "lms.", e.g., lms.yjn.
newdata Optional data frame containing secondary variables such as sex. It should have a maximum of one row. The default is to use the original data.
x0 Numeric. The value of the primary variable at which to make the 'slice'.
y.arg Numerical vector. The values of the response variable at which to evaluate the density. This should be a grid that is fine enough to ensure the plotted curves are smooth.
show.plot Logical. Plot it? If FALSE no plot will be done.
... Graphical parameter that are passed into plotdeplot.lmscreg.

Details

This function calls, e.g., deplot.lms.yjn in order to compute the density function.

Value

The original object but with a list placed in the slot post, called @post$deplot. The list has components
newdata The argument newdata above, or a one-row data frame constructed out of the x0 argument.
y The argument y.arg above.
density Vector of the density function values evaluated at y.arg.

Note

plotdeplot.lmscreg actually does the plotting.
Author(s)

Thomas W. Yee

References


See Also

plotdeplot.lmscreg, qtplot.lmscreg, lms.bcn, lms.bcg, lms.yjn.

Examples

```r
## Not run:
fit <- vgam(BMI ~ s(age, df = c(4, 2)), fam = lms.bcn(zero = 1), data = bmi.nz)
ygrid <- seq(15, 43, by = 0.25)
deplot(fit, x0 = 20, y = ygrid, xlab = "BMI", col = "green", llwd = 2,
       main = "BMI distribution at ages 20 (green), 40 (blue), 60 (red")
) deplot(fit, x0 = 40, y = ygrid, add = TRUE, col = "blue", llwd = 2)
deplot(fit, x0 = 60, y = ygrid, add = TRUE, col = "red", llwd = 2) -> a

names(a@post$deplot)
a@post$deplot$newdata
head(a@post$deplot$y)
head(a@post$deplot$density)
```

## End(Not run)

---

**depvar**

**Response variable extracted**

Description

A generic function that extracts the response/dependent variable from objects.

Usage

depvar(object, ...)

Arguments

- **object**
  - An object that has some response/dependent variable.

- **...**
  - Other arguments fed into the specific methods function of the model. In particular, sometimes type = c("lm", "lm2") is available, in which case the first one is chosen if the user does not input a value. The latter value corresponds to argument form2, and sometimes a response for that is optional.
Details

By default this function is preferred to calling `fit@y`, say.

Value

The response/dependent variable, usually as a matrix or vector.

Author(s)

Thomas W. Yee

See Also

`model.matrix`, `vglm`.

Examples

```r
pneumo <- transform(pneumo, let = log(exposure.time))
(fit <- vglm(cbind(normal, mild, severe) ~ let, propodds, data = pneumo))
fit@y    # Sample proportions (not recommended)
depvar(fit) # Better than using fit@y; dependent variable (response)
weights(fit, type = "prior") # Number of observations
```

---

**df.residual**

<table>
<thead>
<tr>
<th>Residual Degrees-of-Freedom</th>
</tr>
</thead>
</table>

Description

Returns the residual degrees-of-freedom extracted from a fitted VGLM object.

Usage

```r
df.residual_vlm(object, type = c("vlm", "lm"), ...)
```

Arguments

- `object` an object for which the degrees-of-freedom are desired, e.g., a `vglm` object.
- `type` the type of residual degrees-of-freedom wanted. In some applications the 'usual' LM-type value may be more appropriate. The default is the first choice.
- `...` additional optional arguments.
Details

When a VGLM is fitted, a *large* (VLM) generalized least squares (GLS) fit is done at each IRLS iteration. To do this, an ordinary least squares (OLS) fit is performed by transforming the GLS using Cholesky factors. The number of rows is $M$ times the ‘ordinary’ number of rows of the LM-type model: $nM$. Here, $M$ is the number of linear/additive predictors. So the formula for the VLM-type residual degrees-of-freedom is $nM - p^*$ where $p^*$ is the number of columns of the ‘big’ VLM matrix. The formula for the LM-type residual degrees-of-freedom is $n - p_j$ where $p_j$ is the number of columns of the ‘ordinary’ LM matrix corresponding to the $j$th linear/additive predictor.

Value

The value of the residual degrees-of-freedom extracted from the object. When `type = "vlm"` this is a single integer, and when `type = "lm"` this is a $M$-vector of integers.

See Also

`vglm`, `deviance`, `lm`.

Examples

```r
pneumo <- transform(pneumo, let = log(exposure.time))
(fit <- vglm(cbind(normal, mild, severe) ~ let, propodds, data = pneumo))
head(model.matrix(fit, type = "vlm"))
head(model.matrix(fit, type = "lm"))

df.residual(fit, type = "vlm") # n * M - p_vlm
nobs(fit, type = "vlm") # n * M
nvar(fit, type = "vlm") # p_vlm

df.residual(fit, type = "lm") # n - p_LM(j); Useful in some situations
nobs(fit, type = "lm") # n
nvar(fit, type = "lm") # p_LM
nvar_vlm(fit, type = "lm") # p_LM(j) (<= p_LM elementwise)
```

dgenpois

*The Generalized Poisson Distribution*

Description

Density for the Generalized Poisson Distribution.

Usage

dgenpois(x, lambda = 0, theta, log = FALSE)
Arguments

- `x`: vector of quantiles.
- `lambda`, `theta`: See `genpoisson`. The default value of `lambda` corresponds to an ordinary Poisson distribution.
- `log`: Logical. If TRUE then the logarithm of the density is returned.

Details

Most of the background to this function is given in `genpoisson`. Some warnings relevant to this distribution are given there, especially relating to the complicated range of the parameter `lambda` about or near $-1$.

Note that numerical round off errors etc. can occur; see below for an example.

Value

dgenpois gives the density. The value NaN is returned for elements not satisfying the parameter restrictions, e.g., if $\lambda > 1$.

Author(s)

T. W. Yee

See Also

`genpoisson, dpois`.

Examples

```R
sum(dgenpois(0:1000, lambda = -0.5, theta = 2))  # Not perfect...
## Not run:
lambda <- -0.2; theta <- 2; y <- 0:10
proby <- dgenpois(y, lambda = lambda, theta = theta, log = FALSE)
plot(y, proby, type = "h", col = "blue", lwd = 2, ylab = "P[Y=y]",
     main = paste("Y ~ Generalized Poisson(lambda=", lambda,
             ", theta=", theta, ")", sep = ""), las = 1,
     sub = "Orange is the Poisson probability function")
sum(proby)
lines(y + 0.1, dpois(y, theta), type = "h", lwd = 2, col = "orange")

## End(Not run)
```
**dhuber**

*Huber’s least favourable distribution*

**Description**
Density, distribution function, quantile function and random generation for Huber’s least favourable distribution, see Huber and Ronchetti (2009).

**Usage**
- `dhuber(x, k = 0.862, mu = 0, sigma = 1, log = FALSE)`
- `edhuber(x, k = 0.862, mu = 0, sigma = 1, log = FALSE)`
- `rhuber(n, k = 0.862, mu = 0, sigma = 1)`
- `qhuber(p, k = 0.862, mu = 0, sigma = 1, lower.tail = TRUE, log.p = FALSE)`
- `phuber(q, k = 0.862, mu = 0, sigma = 1, lower.tail = TRUE, log.p = FALSE)`

**Arguments**
- `x, q` numeric vector, vector of quantiles.
- `p` vector of probabilities.
- `n` number of random values to be generated. If `length(n) > 1` then the length is taken to be the number required.
- `k` numeric. Borderline value of central Gaussian part of the distribution. This is known as the tuning constant, and should be positive. For example, `k = 0.862` refers to a 20% contamination neighborhood of the Gaussian distribution. If `k = 1.40` then this is 5% contamination.
- `mu` numeric. Distribution mean.
- `sigma` numeric. Distribution scale (`sigma = 1` defines the distribution in standard form, with standard Gaussian centre).
- `log` Logical. If `log = TRUE` then the logarithm of the result is returned.
- `lower.tail, log.p` Same meaning as in `pnorm` or `qnorm`.

**Details**
Details are given in `huber2`, the VGAM family function for estimating the parameters `mu` and `sigma`.

**Value**
- `dhuber` gives out a vector of density values.
- `edhuber` gives out a list with components `val` (density values) and `eps` (contamination proportion).
- `rhuber` gives out a vector of random numbers generated by Huber’s least favourable distribution.
- `phuber` gives the distribution function, `qhuber` gives the quantile function.
Author(s)

Christian Hennig wrote \( [d, \text{ed}, r]\)huber() (from smoothmest) and slight modifications were made by T. W. Yee to replace looping by vectorization and addition of the log argument. Arash Ardalan wrote \([pq]\)huber(), and two arguments for these were implemented by Kai Huang. This helpfile was adapted from smoothmest.

See Also

huber2.

Examples

```r
data <- read.matrix()
set.seed(123456)
edhuber(1:5, k = 1.5)
rhuber5()

# Not run: mu <- 3; xx <- seq(-2, 7, len = 100) # Plot CDF and PDF
plot(xx, dhuber(xx, mu = mu), type = "l", col = "blue", las = 1, ylab = "",
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Purple lines are the 10, 20, ..., 90 percentiles",
     ylim = 0:1)
abline(h = 0, col = "blue", lty = 2)
lines(xx, phuber(xx, mu = mu), type = "l", col = "orange")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qhuber(probs, mu = mu)
lines(Q, dhuber(Q, mu = mu), col = "purple", lty = 3, type = "h")
lines(Q, phuber(Q, mu = mu), col = "purple", lty = 3, type = "h")
abline(h = probs, col = "purple", lty = 3)
phuber(Q, mu = mu) - probs # Should be all 0s

# End(Not run)
```

dirichlet

Fitting a Dirichlet Distribution

Description

Fits a Dirichlet distribution to a matrix of compositions.

Usage

```r
dirichlet(link = "loge", parallel = FALSE, zero = NULL,
          imethod = 1)
```

Arguments

- **link**
  - Link function applied to each of the \( M \) (positive) shape parameters \( \alpha_j \). See Links for more choices. The default gives \( \eta_j = \log(\alpha_j) \).
- **parallel**
  - zero
  - **imethod**
    - See CommonVGAMffArguments for more information.
Details

In this help file the response is assumed to be a \( M \)-column matrix with positive values and whose rows each sum to unity. Such data can be thought of as compositional data. There are \( M \) linear/additive predictors \( \eta_j \).

The Dirichlet distribution is commonly used to model compositional data, including applications in genetics. Suppose \((Y_1, \ldots, Y_M)^T\) is the response. Then it has a Dirichlet distribution if \((Y_1, \ldots, Y_{M-1})^T\) has density

\[
\frac{\Gamma(\alpha_+) \prod_{j=1}^{M} y_j^{\alpha_j - 1}}{\prod_{j=1}^{M} \Gamma(\alpha_j)}
\]

where \( \alpha_+ = \alpha_1 + \cdots + \alpha_M, \alpha_j > 0 \), and the density is defined on the unit simplex

\[
\Delta_M = \left\{ (y_1, \ldots, y_M)^T : y_1 > 0, \ldots, y_M > 0, \sum_{j=1}^{M} y_j = 1 \right\}.
\]

One has \( E(Y_j) = \frac{\alpha_j}{\alpha_+} \), which are returned as the fitted values. For this distribution Fisher scoring corresponds to Newton-Raphson.

The Dirichlet distribution can be motivated by considering the random variables \((G_1, \ldots, G_M)^T\) which are each independent and identically distributed as a gamma distribution with density \( f(g_j) = g_j^{\alpha_j - 1}e^{-g_j}/\Gamma(\alpha_j) \). Then the Dirichlet distribution arises when \( Y_j = G_j/(G_1 + \cdots + G_M) \).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

When fitted, the fitted.values slot of the object contains the \( M \)-column matrix of means.

Note

The response should be a matrix of positive values whose rows each sum to unity. Similar to this is count data, where probably a multinomial logit model (multinomial) may be appropriate. Another similar distribution to the Dirichlet is the Dirichlet-multinomial (see dirmultinomial).

Author(s)

Thomas W. Yee

References


See Also

rdiric, dirmultinomial, multinomial, simplex.
Examples

```r
dirmul.old

Examples
ydata <- data.frame(rdirc(n = 1000,
shape = exp(c(y1 = -1, y2 = 1, y3 = 0))))
fit <- vglm(cbind(y1, y2, y3) ~ 1, dirichlet,
data = ydata, trace = TRUE, crit = "coef")
Coef(fit)
coeff(fit, matrix = TRUE)
head(fitted(fit))
```

Fitting a Dirichlet-Multinomial Distribution

dirmul.old

Description

Fits a Dirichlet-multinomial distribution to a matrix of non-negative integers.

Usage

```r
dirmul.old(link = "logCl", ialpha = 0.01, parallel = FALSE, zero = NULL)
```

Arguments

- `link`: Link function applied to each of the \( M \) (positive) shape parameters \( \alpha_j \) for \( j = 1, \ldots, M \). See `Links` for more choices. Here, \( M \) is the number of columns of the response matrix.
- `ialpha`: Numeric vector. Initial values for the alpha vector. Must be positive. Recycled to length \( M \).
- `parallel`: A logical, or formula specifying which terms have equal/unequal coefficients.
- `zero`: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \{1,2,\ldots,M\}.

Details

The Dirichlet-multinomial distribution, which is somewhat similar to a Dirichlet distribution, has probability function

\[
P(Y_1 = y_1, \ldots, Y_M = y_M) = \binom{2y_*}{y_1, \ldots, y_M} \frac{\Gamma(\alpha_+)}{\Gamma(2y_* + \alpha_+)} \prod_{j=1}^{M} \frac{\Gamma(y_j + \alpha_j)}{\Gamma(\alpha_j)}
\]

for \( \alpha_j > 0, \alpha_+ = \alpha_1 + \cdots + \alpha_M \), and \( 2y_* = y_1 + \cdots + y_M \). Here, \( \binom{n}{k} \) means “\( n \) choose \( k \)” and refers to combinations (see `choose`). The (posterior) mean is

\[
E(Y_j) = (y_j + \alpha_j) / (2y_* + \alpha_+)
\]

for \( j = 1, \ldots, M \), and these are returned as the fitted values as a \( M \)-column matrix.
Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Note

The response should be a matrix of non-negative values. Convergence seems to slow down if there are zero values. Currently, initial values can be improved upon.

This function is almost defunct and may be withdrawn soon. Use dirmultinomial instead.

Author(s)

Thomas W. Yee

References


See Also
dirmultinomial, dirichlet, betabinomialff, multinomial.

Examples

```r
# Data from p.50 of Lange (2002)
alleleCounts <- c(2, 84, 59, 41, 53, 131, 2, 0,
    0, 50, 137, 78, 54, 51, 0, 0,
    0, 88, 128, 26, 55, 95, 0, 0,
    0, 16, 40, 8, 68, 14, 7, 1)
dim(alleleCounts) <- c(8, 4)
alleleCounts <- data.frame(t(alleleCounts))
dimnames(alleleCounts) <- list(c("White","Black","Chicano","Asian"),
    paste("Allele", 5:12, sep = "\""))

set.seed(123)  # @initialize uses random numbers
fit <- vglm(cbind(Allele5,Allele6,Allele7,Allele8,Allele9,
    Allele10,Allele11,Allele12) ~ 1, dirmul.old,
    trace = TRUE, crit = "c", data = alleleCounts)

(sfit <- summary(fit))
vcov(sfit)
round(eta2theta(coef(fit), fit@misc$link, fit@misc$earg), digits = 2)  # not preferred
```
dirmultinomial

Fitting a Dirichlet-Multinomial Distribution

Description

Fits a Dirichlet-multinomial distribution to a matrix response.

Usage

dirmultinominal(lphi = "logit", iphi = 0.10, parallel = FALSE, zero = "M")

Arguments

lphi Link function applied to the \( \phi \) parameter, which lies in the open unit interval \((0, 1)\). See Links for more choices.
iphi Numeric. Initial value for \( \phi \). Must be in the open unit interval \((0, 1)\). If a failure to converge occurs then try assigning this argument a different value.
parallel A logical (formula not allowed here) indicating whether the probabilities \( \pi_1, \ldots, \pi_{M-1} \) are to be equal via equal coefficients. Note \( \pi_M \) will generally be different from the other probabilities. Setting parallel = TRUE will only work if you also set zero = NULL because of interference between these arguments (with respect to the intercept term).
zero An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \{1, 2, \ldots, M\}. If the character "M" then this means the numerical value \( M \), which corresponds to linear/additive predictor associated with \( \phi \). Setting zero = NULL means none of the values from the set \{1, 2, \ldots, M\}.

Details

The Dirichlet-multinomial distribution arises from a multinomial distribution where the probability parameters are not constant but are generated from a multivariate distribution called the Dirichlet distribution. The Dirichlet-multinomial distribution has probability function

\[
P(Y_1 = y_1, \ldots, Y_M = y_M) = \binom{N_*}{y_1, \ldots, y_M} \prod_{j=1}^{M} \prod_{r=1}^{y_j} (\pi_j(1 - \phi) + (r - 1)\phi) \prod_{r=1}^{N_*} (1 - \phi + (r - 1)\phi)
\]
where $\phi$ is the over-dispersion parameter and $N_\ast = y_1 + \cdots + y_M$. Here, $\binom{n}{k}$ means “$a$ choose $b$” and refers to combinations (see choose). The above formula applies to each row of the matrix response. In this VGAM family function the first $M - 1$ linear/additive predictors correspond to the first $M - 1$ probabilities via

$$\eta_j = \log(P[Y = j]/P[Y = M]) = \log(\pi_j/\pi_M)$$

where $\eta_j$ is the $j$th linear/additive predictor ($\eta_M = 0$ by definition for $P[Y = M]$ but not for $\phi$) and $j = 1, \ldots, M - 1$. The $M$th linear/additive predictor corresponds to $\text{lphi}$ applied to $\phi$.

Note that $E(Y_j) = N_\ast \pi_j$ but the probabilities (returned as the fitted values) $\pi_j$ are bundled together as a $M$-column matrix. The quantities $N_\ast$ are returned as the prior weights.

The beta-binomial distribution is a special case of the Dirichlet-multinomial distribution when $M = 2$; see betabinomial. It is easy to show that the first shape parameter of the beta distribution is $\text{shape1} = \pi(1/\phi - 1)$ and the second shape parameter is $\text{shape2} = (1 - \pi)(1/\phi - 1)$. Also, $\phi = 1/(1 + \text{shape1} + \text{shape2})$, which is known as the intra-cluster correlation coefficient.

Value

An object of class ”vglmff” (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

If the model is an intercept-only model then @misc (which is a list) has a component called shape which is a vector with the $M$ values $\pi_j(1/\phi - 1)$.

Warning

This VGAM family function is prone to numerical problems, especially when there are covariates.

Note

The response can be a matrix of non-negative integers, or else a matrix of sample proportions and the total number of counts in each row specified using the weights argument. This dual input option is similar to multinomial.

To fit a ‘parallel’ model with the $\phi$ parameter being an intercept-only you will need to use the constraints argument.

Currently, Fisher scoring is implemented. To compute the expected information matrix a for loop is used; this may be very slow when the counts are large. Additionally, convergence may be slower than usual due to round-off error when computing the expected information matrices.

Author(s)

Thomas W. Yee

References


**See Also**

`dirmul.old`, `betabinomial`, `betabinomialff`, `dirichlet`, `multinomial`.

**Examples**

```r
nn <- 10; M <- 5
ydata <- data.frame(round(matrix(runif(nn * M, max = 10), nn, M))) # Integer counts
colnames(ydata) <- paste("y", 1:M, sep = "")

fit <- vglm(cbind(y1, y2, y3, y4, y5) ~ 1, dirmultinomial, data = ydata, trace = TRUE)
head(fitted(fit))
depvar(fit) # Sample proportions
weights(fit, type = "prior", matrix = FALSE) # Total counts per row

ydata <- transform(ydata, x2 = runif(nn))
fit <- vglm(cbind(y1, y2, y3, y4, y5) ~ x2, dirmultinomial, data = ydata, trace = TRUE)
## Not run: # This does not work:
Coef(fit)
## End(Not run)
coef(fit, matrix = TRUE)
(sfit <- summary(fit))
vcov(sfit)
```

---

**dlogF**  

*log F Distribution*

**Description**

Density for the log F distribution.

**Usage**

```r
dlogF(x, shape1, shape2, log = FALSE)
```

**Arguments**

- `x`  
  Vector of quantiles.
- `shape1`, `shape2`  
  Positive shape parameters.
- `log`  
  if TRUE then the log density is returned, else the density.

**Details**

The details are given in `logF`.
Value
dlogf gives the density.

Author(s)
T. W. Yee

See Also
hypersecant.

Examples
```r
## Not run: shape1 <- 1.5; shape2 <- 0.5; x <- seq(-5, 8, length = 1001)
plot(x, dlogf(x, shape1, shape2), type = "l",
     las = 1, col = "blue", ylab = "pdf",
     main = "log f density function")

## End(Not run)
```

---

double.cens.normal Univariate Normal Distribution with Double Censoring

Description
Maximum likelihood estimation of the two parameters of a univariate normal distribution when there is double censoring.

Usage
double.cens.normal(r1 = 0, r2 = 0, lmu = "identitylink", lsd = "loge",
                   imu = NULL, isd = NULL, zero = 2)

Arguments
- `r1, r2` Integers. Number of smallest and largest values censored, respectively.
- `lmu, lsd` Parameter link functions applied to the mean and standard deviation. See Links for more choices.
- `imu, isd, zero` See CommonVGAMffArguments for more information.

Details
This family function uses the Fisher information matrix given in Harter and Moore (1966). The matrix is not diagonal if either `r1` or `r2` are positive.

By default, the mean is the first linear/additive predictor and the log of the standard deviation is the second linear/additive predictor.
Value

An object of class "vglmff" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, and \texttt{vgam}.

Note

This family function only handles a vector or one-column matrix response. The \texttt{weights} argument, if used, are interpreted as frequencies, therefore it must be a vector with positive integer values.

With no censoring at all (the default), it is better (and equivalent) to use \texttt{uninormal}.

Author(s)

T. W. Yee

References


See Also

\texttt{uninormal, cens.normal, tobit}.

Examples

```r
## Not run:  # Repeat the simulations described in Harter and Moore (1966)
SIMS <- 100  # Number of simulations (change this to 1000)
mu.save <- sd.save <- rep(NA, len = SIMS)
r1 <- 0; r2 <- 4; nn <- 20
for (sim in 1:SIMS) {
  y <- sort(rnorm(nn))
  y <- y[(1+r1):(nn-r2)]  # Delete r1 smallest and r2 largest
  fit <- vglm(y ~ 1, double.cens.normal(r1 = r1, r2 = r2))
  mu.save[sim] <- predict(fit)[1, 1]
  sd.save[sim] <- exp(predict(fit)[1, 2])  # Assumes a log link and −1
}
c(mean(mu.save), mean(sd.save))  # Should be c(0,1)
c(sd(mu.save), sd(sd.save))

## End(Not run)

# Data from Sarhan and Greenberg (1962); MLEs are mu = 9.2606, sd = 1.3754
strontium90 <- data.frame(y = c(8.2, 8.4, 9.1, 9.8, 9.9))
fit <- vglm(y ~ 1, double.cens.normal(r1 = 2, r2 = 3, isd = 6), strontium90, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
```
double.expbinomial

Description

Fits a double exponential binomial distribution by maximum likelihood estimation. The two parameters here are the mean and dispersion parameter.

Usage

double.expbinomial(lmean = "logit", ldispersion = "logit",
                   idispersion = 0.25, zero = 2)

Arguments

lmean, ldispersion
Link functions applied to the two parameters, called \( \mu \) and \( \theta \) respectively below. See Links for more choices. The defaults cause the parameters to be restricted to \((0, 1)\).

idispersion
Initial value for the dispersion parameter. If given, it must be in range, and is recycled to the necessary length. Use this argument if convergence failure occurs.

zero
An integer specifying which linear/additive predictor is to be modelled as an intercept only. If assigned, the single value should be either \( Q \) or \( R \). The default is to have a single dispersion parameter value. To model both parameters as functions of the covariates assign zero = NULL.

Details

This distribution provides a way for handling overdispersion in a binary response. The double exponential binomial distribution belongs the family of double exponential distributions proposed by Efron (1986). Below, equation numbers refer to that original article. Briefly, the idea is that an ordinary one-parameter exponential family allows the addition of a second parameter \( \theta \) which varies the dispersion of the family without changing the mean. The extended family behaves like the original family with sample size changed from \( n \) to \( n\theta \). The extended family is an exponential family in \( \mu \) when \( n \) and \( \theta \) are fixed, and an exponential family in \( \theta \) when \( n \) and \( \mu \) are fixed. Having \( 0 < \theta < 1 \) corresponds to overdispersion with respect to the binomial distribution. See Efron (1986) for full details.

This VGAM family function implements an approximation (2.10) to the exact density (2.4). It replaces the normalizing constant by unity since the true value nearly equals 1. The default model fitted is \( \eta_1 = \logit(\mu) \) and \( \eta_2 = \logit(\theta) \). This restricts both parameters to lie between 0 and 1, although the dispersion parameter can be modelled over a larger parameter space by assigning the arguments ldispersion and edispersion.

Approximately, the mean (of \( Y \)) is \( \mu \). The effective sample size is the dispersion parameter multiplied by the original sample size, i.e., \( n\theta \). This family function uses Fisher scoring, and the two estimates are asymptotically independent because the expected information matrix is diagonal.
**Value**

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm.

**Warning**

Numerical difficulties can occur; if so, try using idispersion.

**Note**

This function processes the input in the same way as binomialff, however multivariate responses are not allowed (binomialff(multiple.responses = FALSE)).

**Author(s)**

T. W. Yee

**References**


**See Also**

binomialff, toxop, CommonVGAMffArguments.

**Examples**

```r
# This example mimics the example in Efron (1986).
# The results here differ slightly.

# Scale the variables
toxop <- transform(toxop,
  phat = positive / ssize,
  srainfall = scale(rainfall),  # (6.1)
  sN = scale(ssize))          # (6.2)

# A fit similar (should be identical) to Section 6 of Efron (1986).
# But does not use poly(), and M = 1.25 here, as in (5.3)
cmlist <- list("(Intercept)" = diag(2),
  "I(srainfall)" = rbind(1, 0),
  "I(srainfall^2)" = rbind(1, 0),
  "I(srainfall^3)" = rbind(1, 0),
  "I(sN)" = rbind(0, 1),
  "I(sN^2)" = rbind(0, 1))
fit <- vglm(cbind(phat, 1 - phat) * ssize ~
  1(srainfall) + I(srainfall^2) + I(srainfall^3) +
  1(sN) + I(sN^2),
  double.expbinomial(ldisp = extlogit(min = 0, max = 1.25),
  idisp = 0.2, zero = NULL),
  toxop, trace = TRUE, constraints = cmlist)
```
# Now look at the results
coef(fit, matrix = TRUE)
head(fitted(fit))
summary(fit)
vcov(fit)
sqrt(diag(vcov(fit))) # Standard errors

# Effective sample size (not quite the last column of Table 1)
head(predict(fit))
Dispersion <- extlogit(predict(fit)[,2], min = 0, max = 1.25, inverse = TRUE)
c(round(weights(fit, type = "prior") * Dispersion, digits = 1))

# Ordinary logistic regression (gives same results as (6.5))
ofit <- vglm(cbind(phat, 1 - phat) * ssize ~
  I(srainfall) + I(srainfall^2) + I(srainfall^3),
  binomialff, toxop, trace = TRUE)

# Same as fit but it uses poly(), and can be plotted (cf. Figure 1)
cmlist2 <- list("(Intercept)" = diag(2),
  "poly(srainfall, degree = 3)" = rbind(1, 0),
  "poly(sN, degree = 2)" = rbind(0, 1))
fit2 <- vglm(cbind(phat, 1 - phat) * ssize ~
  poly(srainfall, degree = 3) + poly(sN, degree = 2),
  double.expbinomial(idisp = extlogit(min = 0, max = 1.25),
  idisp = 0.2, zero = NULL),
  toxop, trace = TRUE, constraints = cmlist2)

## Not run: par(mfrow = c(1, 2))
plot(as(fit2, "vgam"), se = TRUE, lcol = "blue", scol = "orange") # Cf. Figure 1

## Cf. Figure 1(a)
par(mfrow = c(1, 2))
ooo <- with(toxop, sort.list(rainfall))
with(toxop, plot(rainfall[ooo], fitted(fit2)[ooo], type = "l",
  col = "blue", las = 1, ylim = c(0.3, 0.65)))
with(toxop, points(rainfall[ooo], fitted(ofit)[ooo], col = "orange",
  type = "b", pch = 19))

## Cf. Figure 1(b)
ooo <- with(toxop, sort.list(ssize))
with(toxop, plot(ssize[ooo], Dispersion[ooo], type = "l", col = "blue",
  las = 1, xlim = c(0, 100)))

## End(Not run)
**Description**
Relative frequencies of serum proteins in white Pekin ducklings as determined by electrophoresis.

**Usage**
data(ducklings)

**Format**
The format is: chr "ducklings"

**Details**
Columns p1, p2, p3 stand for pre-albumin, albumin, globulins respectively. These were collected from 3-week old white Pekin ducklings. Let $Y_1$ be proportional to the total milligrams of pre-albumin in the blood serum of a duckling. Similarly, let $Y_2$ and $Y_3$ be directly proportional to the same factor as $Y_1$ to the total milligrams respectively of albumin and globulins in its blood serum. The proportion of pre-albumin is given by $Y_1/(Y_1 + Y_2 + Y_3)$, and similarly for the others.

**Source**
Mosimann, J. E. (1962) On the compound multinomial distribution, the multivariate $\beta$-distribution, and correlations among proportions, Biometrika, 49, 65–82.

**See Also**
dirichlet.

**Examples**
print(ducklings)

---

**enzyme**

**Enzyme Data**

**Description**
Enzyme velocity and substrate concentration.

**Usage**
data(enzyme)

**Format**
A data frame with 12 observations on the following 2 variables.

concentration a numeric explanatory vector; substrate concentration
velocity a numeric response vector; enzyme velocity
Details
Sorry, more details need to be included later.

Source
Sorry, more details need to be included later.

References

See Also
micmen.

Examples
```r
## Not run:
fit <- vglm(velocity ~ 1, micmen, data = enzyme, trace = TRUE,
            form2 = ~ conc - 1, crit = "crit")
summary(fit)
## End(Not run)
```

---

**erf**  
*Error Function, and variants*

**Description**
Computes the error function, or its inverse, based on the normal distribution. Also computes the complement of the error function, or its inverse.

**Usage**
erf(x, inverse = FALSE)  
erfc(x, inverse = FALSE)

**Arguments**
x Numeric.
inverse Logical. Of length 1.
Details

\( \text{Erf}(x) \) is defined as

\[
\text{Erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2) \, dt
\]

so that it is closely related to \( \text{pnorm} \). The inverse function is defined for \( x \) in \((-1, 1)\).

Value

Returns the value of the function evaluated at \( x \).

Note

Some authors omit the term \( 2/\sqrt{\pi} \) from the definition of \( \text{Erf}(x) \). Although defined for complex arguments, this function only works for real arguments.

The complementary error function \( \text{erfc}(x) \) is defined as \( 1 - \text{erf}(x) \), and is implemented by \( \text{erfc} \). Its inverse function is defined for \( x \) in \((0, 2)\).

Author(s)

T. W. Yee

References


See Also

\text{pnorm}.

Examples

```r
## Not run:
curve(\text{erf}, -3, 3, \text{col} = \text{"orange"}, \text{ylab} = \text{""}, \text{las} = 1)
curve(\text{pnorm}, -3, 3, \text{add} = \text{TRUE}, \text{col} = \text{"blue"}, \text{lty} = \text{"dotted"}, \text{lwd} = 2)
\text{abline}(v = 0, h = 0, \text{lty} = \text{"dashed")}
\text{legend}(\text{"topleft"}, \text{c(\text{"erf(x)"}, \text{"pnorm(x)"})}, \text{col} = \text{c("orange", \"blue")},
\text{lty} = \text{c("solid", \"dotted")}, \text{lwd} = 1:2)
## End(Not run)
```
Erlang Distribution

Description
Estimates the scale parameter of the Erlang distribution by maximum likelihood estimation.

Usage
erlang(shape.arg, link = "loge", imethod = 1, zero = NULL)

Arguments
- shape.arg: The shape parameters. The user must specify a positive integer, or integers for multiple responses. They are recycled by .row = TRUE according to matrix.
- link: Link function applied to the (positive) scale parameter. See Links for more choices.
- imethod, zero: See CommonVGAMffArguments for more details.

Details
The Erlang distribution is a special case of the gamma distribution with shape that is a positive integer. If shape.arg = 1 then it simplifies to the exponential distribution. As illustrated in the example below, the Erlang distribution is the distribution of the sum of shape.arg independent and identically distributed exponential random variates.

The probability density function of the Erlang distribution is given by

\[ f(y) = \exp(-y/scale) y^{shape-1} scale^{-shape} / \Gamma(shape) \]

for known positive integer shape, unknown scale > 0 and y > 0. Here, \( \Gamma(\text{shape}) \) is the gamma function, as in gamma. The mean of \( Y \) is \( \mu = \text{shape} \times \text{scale} \) and its variance is \( \text{shape} \times \text{scale}^2 \).

The linear/additive predictor, by default, is \( \eta = \log(\text{scale}) \).

Value
An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note
Multiple responses are permitted. The rate parameter found in gammaR is 1/scale here—see also rgamma.

Author(s)
T. W. Yee
References

Most standard texts on statistical distributions describe this distribution, e.g.,


See Also

gammaR, exponential, simulate.vlm.

Examples

```r
rate <- exp(2); myshape <- 3
edata <- data.frame(y = rep(0, nn <- 1000))
for (ii in 1:myshape)
edata <- transform(edata, y = y + rexp(nn, rate = rate))
fit <- vglm(y ~ 1, erlang(shape = myshape), data = edata, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit) # Answer = 1/rate
1/rate
summary(fit)
```

Expectiles-Exponential

Expectiles of the Exponential Distribution

Description

Density function, distribution function, and expectile function and random generation for the distribution associated with the expectiles of an exponential distribution.

Usage

```r
deexp(x, rate = 1, log = FALSE)
peexp(q, rate = 1, lower.tail = TRUE, log.p = FALSE)
qeexp(p, rate = 1, Maxit.nr = 10, Tol.nr = 1.0e-6,
        lower.tail = TRUE, log.p = FALSE)
reexp(n, rate = 1)
```

Arguments

- `x, p, q` See deunif.
- `n, rate, log` See rexp.
- `lower.tail, log.p` Same meaning as in pexp or qexp.
- `Maxit.nr, Tol.nr` See deunif.
Details

General details are given in `deunif` including a note regarding the terminology used. Here, \( \text{exp} \) corresponds to the distribution of interest, \( F \), and \( \text{eexp} \) corresponds to \( G \). The addition of “e” is for the ‘other’ distribution associated with the parent distribution. Thus \( \text{deexp} \) is for \( g \), \( \text{peexp} \) is for \( G \), \( \text{qeexp} \) is for the inverse of \( G \), \( \text{reexp} \) generates random variates from \( g \).

For \( \text{qeexp} \) the Newton-Raphson algorithm is used to solve for \( y \) satisfying \( p = G(y) \). Numerical problems may occur when values of \( p \) are very close to 0 or 1.

Value

\( \text{deexp}(x) \) gives the density function \( g(x) \). \( \text{peexp}(q) \) gives the distribution function \( G(q) \). \( \text{qeexp}(p) \) gives the expectile function: the value \( y \) such that \( G(y) = p \). \( \text{reexp}(n) \) gives \( n \) random variates from \( G \).

Author(s)

T. W. Yee and Kai Huang

See Also

denorm, dexp.

Examples

```r
my.p <- 0.25; y <- rexp(nn <- 1000)
(myexp <- qeexp(my.p))
sum(myexp - y[y <= myexp]) / sum(abs(myexp - y))  # Should be my.p

## Not run: par(mfrow = c(2,1))
y <- seq(-0.5, 4, len = nn)
plot(y, deexp(yy), col = "blue", ylim = 0:1, xlab = "y", ylab = "g(y)",
     type = "l", main = "g(y) for Exp(1); dotted green is f(y) = dexp(y)")
lines(y, dexp(yy), col = "darkgreen", lty = "dotted", lwd = 2)  # 'original'

plot(y, peexp(yy), type = "l", col = "blue", ylim = 0:1,
     xlab = "y", ylab = "G(y)", main = "G(y) for Exp(1)")
abline(v = 1, h = 0.5, col = "red", lty = "dashed")
lines(y, pexp(yy), col = "darkgreen", lty = "dotted", lwd = 2)
## End(Not run)
```

Description

Density function, distribution function, and expectile function and random generation for the distribution associated with the expectiles of a normal distribution.
Usage

denorm(x, mean = 0, sd = 1, log = FALSE)
penorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qenorm(p, mean = 0, sd = 1, Maxit.nr = 10, Tol.nr = 1.0e-6,
     lower.tail = TRUE, log.p = FALSE)
renorm(n, mean = 0, sd = 1)

Arguments

x, p, q See deunif.
n, mean, sd, log See rnorm.
lower.tail, log.p
     Same meaning as in pnorm or qnorm.
Maxit.nr, Tol.nr See deunif.

Details

General details are given in deunif including a note regarding the terminology used. Here, norm corresponds to the distribution of interest, \( F \), and enorm corresponds to \( G \). The addition of “e” is for the ‘other’ distribution associated with the parent distribution. Thus denorm is for \( g \), penorm is for \( G \), qenorm is for the inverse of \( G \), renorm generates random variates from \( g \).

For qenorm the Newton-Raphson algorithm is used to solve for \( y \) satisfying \( p = G(y) \). Numerical problems may occur when values of \( p \) are very close to 0 or 1.

Value

denorm(x) gives the density function \( g(x) \). penorm(q) gives the distribution function \( G(q) \). qenorm(p) gives the expectile function: the value \( y \) such that \( G(y) = p \). renorm(n) gives \( n \) random variates from \( G \).

Author(s)

T. W. Yee and Kai Huang

See Also

deunif, deexp, dnorm, amlnormal, lms.bcn.

Examples

my.p <- 0.25; y <- rnorm(nn <- 1000)
(myexp <- qenorm(my.p))
sum(myexp - y[y <= myexp]) / sum(abs(myexp - y)) # Should be my.p

# Non-standard normal
mymean <- 1; mysd <- 2
yy <- rnorm(nn, mymean, mysd)
Expectiles/Quantiles of the Scaled Student t Distribution with 2 Df

Description

Density function, distribution function, and quantile/expectile function and random generation for the scaled Student t distribution with 2 degrees of freedom.

Usage

```r
dsc.t2(x, location = 0, scale = 1, log = FALSE)
psc.t2(q, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
qsc.t2(p, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
rsc.t2(n, location = 0, scale = 1)
```

Arguments

- `x, q` Vector of expectiles/quantiles. See the terminology note below.
- `p` Vector of probabilities. These should lie in (0, 1).
- `n, log` See `runif`.
- `location, scale` Location and scale parameters. The latter should have positive values. Values of these vectors are recycled.
- `lower.tail, log.p` Same meaning as in `pt` or `qt`.

```r
(myexp <- qenorm(my.p, mymean, mysd))
sum(myexp - yy[yy <= myexp]) / sum(abs(myexp - yy))  # Should be my.p
penorm(-Inf, mymean, mysd)  # Should be 0
penorm(Inf, mymean, mysd)  # Should be 1
penorm(mean(yy), mymean, mysd)  # Should be 0.5
abs(qenorm(0.5, mymean, mysd) - mean(yy))  # Should be 0
abs(penorm(myexp, mymean, mysd) - my.p)  # Should be 0
integrate(f = denorm, lower = -Inf, upper = Inf,
        mymean, mysd)  # Should be 1
```

```r
## Not run:
par(mfrow = c(2, 1))
yy <- seq(-3, 3, len = nn)
plot(yy, denorm(yy), type = "l", col="blue", xlab = "y", ylab = "g(y)",
     main = "g(y) for N(0,1); dotted green is f(y) = dnorm(y)")
lines(yy, dnorm(yy), col = "darkgreen", lty = "dotted", lwd = 2)  # 'original'

plot(yy, penorm(yy), type = "l", col = "blue", ylim = c(0:1,
           xlab = "y", ylab = "G(y)", main = "G(y) for N(0,1)")
     abline(v = 0, h = 0.5, col = "red", lty = "dashed")
lines(yy, pnorm(yy), col = "darkgreen", lty = "dotted", lwd = 2)
## End(Not run)
```
Details

A Student-t distribution with 2 degrees of freedom and a scale parameter of sqrt(2) is equivalent to the standard form of this distribution (called Koenker’s distribution below). Further details about this distribution are given in sc.studentt.

Value

dsc.t2(x) gives the density function. psc.t2(q) gives the distribution function. qsc.t2(p) gives the expectile and quantile function. rsc.t2(n) gives n random variates.

Author(s)

T. W. Yee and Kai Huang

See Also

dt, sc.studentt.

Examples

my.p <- 0.25; y <- rsc.t2(nn <- 5000)
(myexp <- qsc.t2(my.p))
sum(myexp - y[y <= myexp]) / sum(abs(myexp - y))  # Should be my.p
# Equivalently:
I1 <- mean(y <= myexp) * mean( myexp - y[y <= myexp])
I2 <= mean(y >  myexp) * mean(-myexp + y[y >  myexp])
I1 / (I1 + I2)  # Should be my.p
# Or:
I1 <- sum( myexp - y[y <= myexp])
I2 <- sum(-myexp + y[y >  myexp])

# Non-standard Koenker distribution
myloc <- 1; myscale <- 2
yy <- rsc.t2(nn, myloc, myscale)
(myexp <- qsc.t2(my.p, myloc, myscale))
sum(myexp - yy[yy <= myexp]) / sum(abs(myexp - yy))  # Should be my.p
psc.t2(mean(yy), myloc, myscale)  # Should be 0.5
abs(qsc.t2(0.5, myloc, myscale) - mean(yy))  # Should be 0
abs(psc.t2(myexp, myloc, myscale) - my.p)  # Should be 0

integrate(f = dsc.t2, lower = -Inf, upper = Inf,
locat = myloc, scale = myscale)  # Should be 1

y <- seq(-7, 7, len = 201)
max(abs(dsc.t2(y) - dt(y / sqrt(2), df = 2) / sqrt(2)))  # Should be 0
## Not run: plot(y, dsc.t2(y), type = "l", col = "blue", las = 1,
       ylim = c(0, 0.4), main = "Blue = Koenker; orange = N(0, 1)"
lines(y, dnorm(y), type = "l", col = "orange")
abline(h = 0, v = 0, lty = 2)
## End(Not run)
Description

Density function, distribution function, and expectile function and random generation for the distribution associated with the expectiles of a uniform distribution.

Usage

deunif(x, min = 0, max = 1, log = FALSE)
peunif(q, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
qeunif(p, min = 0, max = 1, Maxit.nr = 10, Tol.nr = 1.0e-6,
       lower.tail = TRUE, log.p = FALSE)
reunif(n, min = 0, max = 1)

Arguments

x, q  Vector of expectiles. See the terminology note below.
p  Vector of probabilities. These should lie in (0, 1).
n, min, max, log
lower.tail, log.p  
See runif.
Maxit.nr  Numeric. Maximum number of Newton-Raphson iterations allowed. A warning is issued if convergence is not obtained for all p values.
Tol.nr  Numeric. Small positive value specifying the tolerance or precision to which the expectiles are computed.

Details

Jones (1994) elucidated on the property that the expectiles of a random variable \(X\) with distribution function \(F(x)\) correspond to the quantiles of a distribution \(G(x)\) where \(G\) is related by an explicit formula to \(F\). In particular, let \(y\) be the \(p\)-expectile of \(F\). Then \(y\) is the \(p\)-quantile of \(G\) where

\[
p = G(y) = (P(y) - yF(y))/(2[P(y) - yF(y)] + y - \mu),
\]

and \(\mu\) is the mean of \(X\). The derivative of \(G\) is

\[
g(y) = (\mu F(y) - P(y))/(2[P(y) - yF(y)] + y - \mu)^2.
\]

Here, \(P(y)\) is the partial moment \(\int_{-\infty}^{y} xf(x) \, dx\) and \(0 < p < 1\). The 0.5-expectile is the mean \(\mu\) and the 0.5-quantile is the median.

A note about the terminology used here. Recall in the \(S\) language there are the dpqr-type functions associated with a distribution, e.g., dunif, punif, qunif, runif, for the uniform distribution. Here, unif corresponds to \(F\) and eunif corresponds to \(G\). The addition of “e” (for expectile) is for the
'other' distribution associated with the parent distribution. Thus deunif is for g, peunif is for G, qeunif is for the inverse of G, reunif generates random variates from g.

For qeunif the Newton-Raphson algorithm is used to solve for y satisfying \( p = G(y) \). Numerical problems may occur when values of \( p \) are very close to 0 or 1.

**Value**

deunif(x) gives the density function \( g(x) \). peunif(q) gives the distribution function \( G(q) \). qeunif(p) gives the expectile function: the expectile \( y \) such that \( G(y) = p \). reunif(n) gives \( n \) random variates from \( G \).

**Author(s)**

T. W. Yee and Kai Huang

**References**


**See Also**

dexp, denorm, dunif, dsc.t2.

**Examples**

```r
my.p <- 0.25; y <- runif(nn <- 1000)
(myexp <- qeunif(my.p))
sum(myexp - y[y <= myexp]) / sum(abs(myexp - y))  # Should be my.p
# Equivalently:
I1 <- mean(y <= myexp) * mean(myexp - y[y <= myexp])
I2 <- mean(y > myexp) * mean(-myexp + y[y > myexp])
I1 / (I1 + I2)  # Should be my.p
# Or:
I1 <- sum(myexp - y[y <= myexp])
I2 <- sum(-myexp + y[y > myexp])

# Non-standard uniform
mymin <- 1; mymax <- 8
yy <- runif(nn, mymin, mymax)
(myexp <- qeunif(yy, mymin, mymax))
sum(myexp - yy[yy <= myexp]) / sum(abs(myexp - yy))  # Should be my.p
peunif(mymin, mymin, mymax)  # Should be 0
peunif(mymax, mymin, mymax)  # Should be 1
peunif(mean(yy), mymin, mymax)  # Should be 0.5
abs(qeunif(0.5, mymin, mymax) - mean(yy))  # Should be 0
abs(qeunif(0.5, mymin, mymax) - (mymin+mymax)/2)  # Should be 0
abs(peunif(myexp, mymin, mymax) - my.p)  # Should be 0
integrate(f = deunif, lower = mymin - 3, upper = mymax + 3,
```
expexpff

Description

Estimates the two parameters of the exponentiated exponential distribution by maximum likelihood estimation.

Usage

expexpff(lrate = "loge", lshape = "loge",
         lrate = NULL, ishape = 1.1, tolerance = 1.0e-6, zero = NULL)

Arguments

lshape, lrate  Parameter link functions for the $\alpha$ and $\lambda$ parameters. See Links for more choices. The defaults ensure both parameters are positive.

ishape  Initial value for the $\alpha$ parameter. If convergence fails try setting a different value for this argument.

irate  Initial value for the $\lambda$ parameter. By default, an initial value is chosen internally using ishape.

tolerance  Numeric. Small positive value for testing whether values are close enough to 1 and 2.

zero  An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The default is none of them. If used, choose one value from the set {1,2}. 

---

230 expexpff

min = mymin, max = mymax)  # Should be 1

## Not run:
par(mfrow = c(2,1))
yy <- seq(0.0, 1.0, len = nn)
plot(yy, deunif(yy), type = "l", col = "blue", ylim = c(0, 2),
xlab = "y", ylab = "g(y)", main = "g(y) for Uniform(0,1)"
lines(yy, dunif(yy), col = "darkgreen", lty = "dotted", lwd = 2)  # 'original'

plot(yy, peunif(yy), type = "l", col = "blue", ylim = 0:1,
xlab = "y", ylab = "G(y)", main = "G(y) for Uniform(0,1)"
abline(a = 0.0, b = 1.0, col = "darkgreen", lty = "dotted", lwd = 2)
abline(v = 0.5, h = 0.5, col = "red", lty = "dashed")
## End(Not run)
The exponentiated exponential distribution is an alternative to the Weibull and the gamma distributions. The formula for the density is

\[ f(y; \lambda, \alpha) = \alpha \lambda (1 - \exp(-\lambda y))^{\alpha - 1} \exp(-\lambda y) \]

where \( y > 0, \lambda > 0 \) and \( \alpha > 0 \). The mean of \( Y \) is \( (\psi'(\alpha + 1) - \psi'(1))/\lambda \) (returned as the fitted values) where \( \psi \) is the digamma function. The variance of \( Y \) is \( (\psi''(1) - \psi''(\alpha + 1))/\lambda^2 \) where \( \psi' \) is the trigamma function.

This distribution has been called the two-parameter generalized exponential distribution by Gupta and Kundu (2006). A special case of the exponentiated exponential distribution: \( \alpha = 1 \) is the exponential distribution.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

Practical experience shows that reasonably good initial values really helps. In particular, try setting different values for the \( \text{ishape} \) argument if numerical problems are encountered or failure to convergence occurs. Even if convergence occurs try perturbing the initial value to make sure the global solution is obtained and not a local solution. The algorithm may fail if the estimate of the shape parameter is too close to unity.

Note

Fisher scoring is used, however, convergence is usually very slow. This is a good sign that there is a bug, but I have yet to check that the expected information is correct. Also, I have yet to implement Type-I right censored data using the results of Gupta and Kundu (2006).

Another algorithm for fitting this model is implemented in expexpff1.

Author(s)

T. W. Yee

References


See Also

expexpff1, gammaR, weibullR, CommonVGAMffArguments.
Examples

# A special case: exponential data
edata <- data.frame(y = rexp(n <- 1000))
fit <- vglm(y ~ 1, fam = expexpfl, data = edata, trace = TRUE, maxit = 99)
coef(fit, matrix = TRUE)
Coef(fit)

# Ball bearings data (number of million revolutions before failure)
edata <- data.frame(bbearings = c(17.88, 28.92, 33.00, 41.52, 42.12, 45.60, 48.80, 51.84, 51.96, 54.12, 55.56, 67.80, 68.64, 68.64, 68.88, 84.12, 93.12, 98.64, 105.12, 105.84, 127.92, 128.04, 173.40))
fit <- vglm(bbearings ~ 1, fam = expexpfl(irate = 0.05, ish = 5),
           trace = TRUE, maxit = 300, data = edata)
coef(fit, matrix = TRUE)
Coef(fit)  # Authors get c(rate=0.0314, shape=5.2589)
logLik(fit)  # Authors get -112.9763

# Failure times of the airconditioning system of an airplane
edata <- data.frame(acplane = c(23, 261, 87, 7, 120, 14, 62, 47, 225, 71, 246, 21, 42, 20, 5, 12, 120, 11, 3, 14, 71, 11, 14, 11, 16, 90, 1, 16, 52, 95))
fit <- vglm(acplane ~ 1, fam = expexpfl(ishape = 0.8, irate = 0.15),
            trace = TRUE, maxit = 99, data = edata)
coef(fit, matrix = TRUE)
Coef(fit)  # Authors get c(rate=0.0145, shape=0.8130)
logLik(fit)  # Authors get log-lik -152.264

expexpfl  

Exponentiated Exponential Distribution

Description

Estimates the two parameters of the exponentiated exponential distribution by maximizing a profile (concentrated) likelihood.

Usage

expexpfl(lrate = "loge", irate = NULL, ishape = 1)

Arguments

lrate  Parameter link function for the (positive) \( \lambda \) parameter. See Links for more choices.
irate  Initial value for the \( \lambda \) parameter. By default, an initial value is chosen internally using ishape.
ishape  Initial value for the \( \alpha \) parameter. If convergence fails try setting a different value for this argument.
Details

See `expexpff` for details about the exponentiated exponential distribution. This family function uses a different algorithm for fitting the model. Given \( \lambda \), the MLE of \( \alpha \) can easily be solved in terms of \( \lambda \). This family function maximizes a profile (concentrated) likelihood with respect to \( \lambda \). Newton-Raphson is used, which compares with Fisher scoring with `expexpff`.

Value

An object of class "`vglmff`" (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Warning

The standard errors produced by a summary of the model may be wrong.

Note

This family function works only for intercept-only models, i.e., \( y \sim 1 \) where \( y \) is the response.

The estimate of \( \alpha \) is attached to the `misc` slot of the object, which is a list and contains the component shape.

As Newton-Raphson is used, the working weights are sometimes negative, and some adjustment is made to these to make them positive.

Like `expexpff`, good initial values are needed. Convergence may be slow.

Author(s)

T. W. Yee

References


See Also

`expexpff`, `CommonVGAMffArguments`.

Examples

```r
# Ball bearings data (number of million revolutions before failure)
edata <- data.frame(bbearings = c(17.88, 28.92, 33.00, 41.52, 42.12, 45.60, 48.80, 51.84, 51.96, 54.12, 55.56, 67.80, 68.64, 68.64, 68.88, 84.12, 93.12, 98.64, 105.12, 105.84, 127.92, 128.04, 173.40))
fit <- vglm(bbearings ~ 1, expexpff(ishape = 4), trace = TRUE, 
            maxit = 250, checkwz = FALSE, data = edata)
coef(fit, matrix = TRUE)
Coef(fit)  # Authors get c(0.0314, 5.2589) with log-lik -112.9763
logLik(fit)
```
The Exponential Geometric Distribution

Description
Density, distribution function, quantile function and random generation for the exponential geometric distribution.

Usage

dexpgeom(x, scale = 1, shape, log = FALSE)
pexpgeom(q, scale = 1, shape)
qexpgeom(p, scale = 1, shape)
rexpgeom(n, scale = 1, shape)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. If length(n) > 1 then the length is taken to be the number required.
scale, shape positive scale and shape parameters.
log Logical. If log = TRUE then the logarithm of the density is returned.

Details
See expgeometric, the VGAM family function for estimating the parameters, for the formula of the probability density function and other details.

Value
dexpgeom gives the density, pexpgeom gives the distribution function, qexpgeom gives the quantile function, and rexpgeom generates random deviates.
expgeometric

### Note

We define scale as the reciprocal of the scale parameter used by Adamidis and Loukas (1998).

### Author(s)

J. G. Lauder and T. W. Yee

### See Also

`expgeometric`, `exponential`, `geometric`.

### Examples

```r
## Not run:
shape <- 0.5; scale <- 1; nn <- 50
x <- seq(-0.10, 3.0, len = nn)
plot(x, dexpgeom(x, scale, shape), type = "l", las = 1, ylim = c(0, 2),
     ylab = paste("[d]expgeom(shape = ", shape, ", scale = "", scale, ")"),
     col = "blue", cex.main = 0.8,
     main = "Blue is density, red is cumulative distribution function",
     sub = "Purple lines are the 10,20,...,90 percentiles")
lines(x, pexpgeom(x, shape, scale), col = "red")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qexpgeom(probs, scale, shape)
lines(Q, dexpgeom(Q, scale, shape), col = "purple", lty = 3, type = "h")
lines(Q, pexpgeom(Q, scale, shape), col = "purple", lty = 3, type = "h")
abline(h = probs, col = "purple", lty = 3)
max(abs(pexpgeom(Q, scale, shape) - probs)) # Should be 0

## End(Not run)
```

---

# expgeometric

## Exponential Geometric Distribution Family Function

### Description

Estimates the two parameters of the exponential geometric distribution by maximum likelihood estimation.

### Usage

```r
expgeometric(lscale = "loge", lshape = "logit",
              iscale = NULL, ishape = NULL,
              tol12 = 1e-05, zero = 1, nsimEIM = 400)
```
expgeometric

Arguments

lscale, lshape  Link function for the two parameters. See Links for more choices.
iscale, ishape  Numeric. Optional initial values for the scale and shape parameters.
tol12  Numeric. Tolerance for testing whether a parameter has value 1 or 2.
zero, nsimEIM  See CommonVGAMffArguments.

Details

The exponential geometric distribution has density function

\[ f(y; c = \text{scale}, s = \text{shape}) = (1/c)(1 - s)e^{-y/c}(1 - se^{-y/c})^{-2} \]

where \( y > 0, c > 0 \) and \( s \in (0, 1) \). The mean, \((c(s - 1)/s) \log(1 - s)\) is returned as the fitted values. Note the median is \( c \log(2 - s) \). Simulated Fisher scoring is implemented.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

We define scale as the reciprocal of the scale parameter used by Adamidis and Loukas (1998).

Author(s)

J. G. Lauder and T. W. Yee

References


See Also

dexpgeom, exponential, geometric.

Examples

```r
## Not run:
Scale <- exp(2); shape = logit(-1, inverse = TRUE);
edata <- data.frame(y = rexpgeom(n = 2000, scale = Scale, shape = shape))
fit <- vglm(y ~ 1, expgeometric, edata, trace = TRUE)
c(with(edata, mean(y)), head(fitted(fit), 1))
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)
## End(Not run)
```
expint  The Exponential Integral and Variants

Description

Computes the exponential integral $E_i(x)$ for real values, as well as $\exp(-x) \times E_i(x)$ and $E_1(x)$.

Usage

expint(x)
expexpint(x)
expint.E1(x)

Arguments

x Numeric. Ideally a vector of positive reals.

Details

The exponential integral $E_i(x)$ function is the integral of $\exp(t)/t$ from 0 to $x$, for positive real $x$. The function $E_1(x)$ is the integral of $\exp(-t)/t$ from $x$ to infinity, for positive real $x$.

Value

Function expint(x) returns $E_i(x)$, function expexpint(x) returns $\exp(-x) \times E_i(x)$, function expint.E1(x) returns $E_1(x)$.

Note

This function has not been tested thoroughly.

Author(s)

T. W. Yee has simply written a small wrapper function to call the above FORTRAN code.

References

http://www.netlib.org/specfun/ei.

See Also

log, exp.
Examples

```r
## Not run:
par(mfrow = c(2, 2))
curve(expint, 0.01, 2, xlim = c(0, 2), ylim = c(-3, 5),
     las = 1, col = "orange")
able(v = (-3):5, h = (-4):5, lwd = 2, lty = "dotted", col = "gray")
able(h = 0, v = 0, lty = "dashed", col = "blue")
curve(exppexpint, 0.01, 2, xlim = c(0, 2), ylim = c(-3, 2),
     las = 1, col = "orange")
able(v = (-3):2, h = (-4):5, lwd = 2, lty = "dotted", col = "gray")
able(h = 0, v = 0, lty = "dashed", col = "blue")
curve(expint.E1, 0.01, 2, xlim = c(0, 2), ylim = c(0, 5),
     las = 1, col = "orange")
able(v = (-3):2, h = (-4):5, lwd = 2, lty = "dotted", col = "gray")
able(h = 0, v = 0, lty = "dashed", col = "blue")

## End(Not run)
```

### explink

**Exponential Link Function**

**Description**
Computes the exponential transformation, including its inverse and the first two derivatives.

**Usage**

```r
explink(theta, bvalue = NULL, inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)
```

**Arguments**

- `theta` Numeric or character. See below for further details.
- `bvalue` See `cloglog`.
- `inverse`, `deriv`, `short`, `tag`
  Details at [Links](#).

**Details**

The exponential link function is potentially suitable for parameters that are positive. Numerical values of `theta` close to negative or positive infinity may result in `0`, `Inf`, `-Inf`, `NA` or `NaN`.
explog

Value

For explink with deriv = 0, the exponential of theta, i.e., $\exp(\theta)$ when inverse = FALSE. And if inverse = TRUE then $\log(\theta)$; if theta is not positive then it will return NaN.

For deriv = 1, then the function returns $d \theta/d \eta$ as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base $e$.

Note

This function has particular use for computing quasi-variances when used with rcim and uninormal.

Numerical instability may occur when theta is close to negative or positive infinity. One way of overcoming this (one day) is to use bvalue.

Author(s)

Thomas W. Yee

See Also

Links, loge, rcim, Qvar, uninormal.

Examples

theta <- rnorm(30)
explink(theta)
max(abs(explink(explink(theta), inverse = TRUE) - theta)) # Should be 0

explog

The Exponential Logarithmic Distribution

Description

Density, distribution function, quantile function and random generation for the exponential logarithmic distribution.

Usage

dexplog(x, scale = 1, shape, log = FALSE)
pexplog(q, scale = 1, shape)
qexplog(p, scale = 1, shape)
rexplog(n, scale = 1, shape)
Arguments

- `x`, `q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If `length(n) > 1` then the length is taken to be the number required.
- `scale`, `shape` positive scale and shape parameters.
- `log` Logical. If `log = TRUE` then the logarithm of the density is returned.

Details

See `explogf`, the VGAM family function for estimating the parameters, for the formula of the probability density function and other details.

Value

dexplog gives the density, pexplog gives the distribution function, qexplog gives the quantile function, and rexplog generates random deviates.

Note

We define `scale` as the reciprocal of the scale parameter used by Tahmasabi and Rezaei (2008).

Author(s)

J. G. Lauder and T. W. Yee

See Also

`explogf`, `exponential`.

Examples

```r
## Not run:
shape <- 0.5; scale <- 2; nn <- 501
x <- seq(-0.5, 6.0, len = nn)
plot(x, dexplog(x, scale, shape), type = "l", las = 1, ylim = c(0, 1.1),
     ylab = paste("[dp]explog(shape = ", shape, ", scale = ", scale, ")"),
     col = "blue", cex.main = 0.8,
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Purple lines are the 10,20,...,90 percentiles")
lines(x, pexplog(x, scale, shape), col = "orange")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qexplog(probs, scale, shape = shape)
lines(Q, dexplog(Q, scale, shape = shape), col = "purple", lty = 3, type = "h")
lines(Q, pexplog(Q, scale, shape = shape), col = "purple", lty = 3, type = "h")
abline(h = probs, col = "purple", lty = 3)
max(abs(pexplog(Q, scale, shape = shape) - probs)) # Should be 0

## End(Not run)
```
explogff

Exponential Logarithmic Distribution Family Function

Description

Estimates the two parameters of the exponential logarithmic distribution by maximum likelihood estimation.

Usage

explogff(lscale = "loge", lshape = "logit",
          iscale = NULL, ishape = NULL,
          tol12 = 1e-05, zero = 1, nsimEIM = 400)

Arguments

lscale, lshape  See CommonVGAMffArguments for information.
tol12  Numeric. Tolerance for testing whether a parameter has value 1 or 2.

Details

The exponential logarithmic distribution has density function

\[ f(y; c, s) = \frac{1}{(\log p)}(\frac{(1/c)(1-s)e^{-y/c}}{(1 - (1-s)e^{-y/c})}) \]

where \( y > 0 \), scale parameter \( c > 0 \), and shape parameter \( s \in (0, 1) \). The mean, \((-\text{polylog}(2, 1-p)c)/\log(s)\) is not returned as the fitted values. Note the median is \( c \log(1 + \sqrt{s}) \) and it is currently returned as the fitted values. Simulated Fisher scoring is implemented.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

We define scale as the reciprocal of the rate parameter used by Tahmasabi and Sadegh (2008). Yet to do: find a polylog() function.

Author(s)

J. G. Lauder and T. W. Yee

References

See Also
dexplog, exponential,

Examples

## Not run
Scale <- exp(2); shape <- logit(-1, inverse = TRUE)
edata <- data.frame(y = rexplog(n = 2000, scale = Scale, shape = shape))
fit <- vglm(y ~ 1, explogff, data = edata, trace = TRUE)
c(with(edata, median(y)), head(fitted(fit), 1))
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)

## End(Not run)

---

exponential  

**Exponential Distribution**

Description

Maximum likelihood estimation for the exponential distribution.

Usage

exponential(link = "loge", location = 0, expected = TRUE,
            ishrinkage = 0.95, parallel = FALSE, zero = NULL)

Arguments

- **link**: Parameter link function applied to the positive parameter \( \lambda \). See Links for more choices.
- **location**: Numeric of length 1, the known location parameter, \( A \), say.
- **expected**: Logical. If TRUE Fisher scoring is used, otherwise Newton-Raphson. The latter is usually faster.
- **ishrinkage, parallel, zero**: See CommonVGAMffArguments for information.

Details

The family function assumes the response \( Y \) has density

\[
f(y) = \lambda \exp(-\lambda(y - A))
\]

for \( y > A \), where \( A \) is the known location parameter. By default, \( A = 0 \). Then \( E(Y) = A + 1/\lambda \) and \( Var(Y) = 1/\lambda^2 \).
Value

An object of class "vglmff" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, and \texttt{vgam}.

Note

Suppose $A = 0$. For a fixed time interval, the number of events is Poisson with mean $\lambda$ if the time between events has a geometric distribution with mean $\lambda^{-1}$. The argument \textit{rate} in \texttt{exponential} is the same as \texttt{rexp} etc. The argument \textit{lambda} in \texttt{rpois} is somewhat the same as \textit{rate} here.

Author(s)

T. W. Yee

References


See Also

\texttt{amlexponential}, \texttt{gpd}, \texttt{laplace}, \texttt{expgeometric}, \texttt{explogff}, \texttt{poissonff}, \texttt{mix2exp}, \texttt{freund61}, \texttt{simulate.vlm}, \texttt{Exponential}.

Examples

```r
edata <- data.frame(x2 = runif(nn <- 100) - 0.5)
edata <- transform(edata, x3 = runif(nn) - 0.5)
edata <- transform(edata, eta = 0.2 - 0.7 * x2 + 1.9 * x3)
edata <- transform(edata, rate = exp(eta))
edata <- transform(edata, y = rexp(nn, rate = rate))
with(edata, stem(y))

fit.slow <- vglm(y ~ x2 + x3, exponential, data = edata, trace = TRUE)
fit.fast <- vglm(y ~ x2 + x3, exponential(exp = FALSE), data = edata,
               trace = TRUE, crit = "coef")
coef(fit.slow, mat = TRUE)
summary(fit.slow)

# Compare results with a GPD. Has a threshold.
threshold <- 0.5
gdata <- data.frame(y1 = threshold + rexp(n = 3000, rate = exp(1.5)))

fit.exp <- vglm(y1 ~ 1, exponential(location = threshold), data = gdata)
coef(fit.exp, matrix = TRUE)
Coef(fit.exp)
logLik(fit.exp)

fit.gpd <- vglm(y1 ~ 1, gpd(threshold = threshold), data = gdata)
coef(fit.gpd, matrix = TRUE)
```
The Exponential Poisson Distribution

Description
Density, distribution function, quantile function and random generation for the exponential poisson distribution.

Usage
dexppois(x, rate = 1, shape, log = FALSE)
pexppois(q, rate = 1, shape, lower.tail = TRUE, log.p = FALSE)
qexppois(p, rate = 1, shape, lower.tail = TRUE, log.p = FALSE)
rexppois(n, rate = 1, shape)

Arguments
x, q vector of quantiles.
p vector of probabilities.
n number of observations. If length(n) > 1 then the length is taken to be the number required.
shape, rate positive parameters.
log Logical. If log = TRUE then the logarithm of the density is returned.
lower.tail, log.p
Same meaning as in pnorm or qnorm.

Details
See expoisson, the VGAM family function for estimating the parameters, for the formula of the probability density function and other details.

Value
dexppois gives the density, pexppois gives the distribution function, qexppois gives the quantile function, and rexppois generates random deviates.

Author(s)
Kai Huang and J. G. Lauder

See Also
expoisson.
Examples

```r
## Not run: rate <- 2; shape <- 0.5; nn <- 201
x <- seq(-0.05, 1.05, len = nn)
plot(x, dexpois(x, rate = rate, shape), type = "l", las = 1, ylim = c(0, 3),
ylab = paste("dexpois(rate = ", rate, ", shape = ", shape, ",)",
col = "blue", cex.main = 0.8,
main = "Blue is the density, orange the cumulative distribution function",
sub = "Purple lines are the 10,20,...,90 percentiles")
lines(x, pexpois(x, rate = rate, shape), col = "orange")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qexpois(probs, rate = rate, shape)
lines(Q, dexpois(Q, rate = rate, shape), col = "purple", lty = 3, type = "h")
lines(Q, pexpois(Q, rate = rate, shape), col = "purple", lty = 3, type = "h")
abline(h = probs, col = "purple", lty = 3); abline(h = 0, col = "gray50")
max(abs(pexpois(Q, rate = rate, shape) - probs)) # Should be 0
## End(Not run)
```

exppoisson

Exponential Poisson Distribution Family Function

Description

Estimates the two parameters of the exponential Poisson distribution by maximum likelihood estimation.

Usage

```r
exppoisson(lrate = "loge", lshape = "loge",
           irate = 2, ishape = 1.1, zero = NULL)
```

Arguments

- **lshape, lrate** Link function for the two positive parameters. See `Links` for more choices.
- **ishape, irate** Numeric. Initial values for the shape and rate parameters. Currently this function is not intelligent enough to obtain better initial values.
- **zero** See `CommonVGAMffArguments`.

Details

The exponential Poisson distribution has density function

\[
f(y; \beta = \text{rate}, \lambda = \text{shape}) = \frac{\lambda \beta}{1 - e^{-\lambda}} e^{-\lambda - \beta y + \lambda \exp(-\beta y)}
\]

where \(y > 0\), and the parameters shape, \(\lambda\), and rate, \(\beta\), are positive. The distribution implies a population facing discrete hazard rates which are multiples of a base hazard. This `VGAM` family function requires the `hypergeo` package (to use their `genhypergeo` function). The median is returned as the fitted value.
Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

This VGAM family function does not work properly!

Author(s)

J. G. Lauder, jamesglauder@gmail.com

References


See Also
dexp, exponential, poisson.

Examples

```r
## Not run:
shape <- exp(1); rate <- exp(2)
rdata <- data.frame(y = rexp(1000, rate = rate, shape = shape))
library("hypergeo")  # Required!
fit <- vglm(y ~ 1, expoisson, data = rdata, trace = FALSE, maxit = 1200)
c(with(rdata, median(y)), head(fitted(fit), 1))
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)
## End(Not run)
```

Felix

*The Felix Distribution*

Description

Density for the Felix distribution.

Usage

dfelix(x, a = 0.25, log = FALSE)
Arguments

- x: vector of quantiles.
- a: See felix.
- log: Logical. If log = TRUE then the logarithm of the density is returned.

Details

See felix, the VGAM family function for estimating the parameter, for the formula of the probability density function and other details.

Value

dfelix gives the density.

Warning

The default value of a is subjective.

Author(s)

T. W. Yee

See Also

felix.

Examples

```r
## Not run:
a <- 0.25; x <- 1:15
plot(x, dfelix(x, a), type = "h", las = 1, col = "blue",
     ylab = paste("dfelix(a="a", ")",
     main = "Felix density function")

## End(Not run)
```
Arguments

link Link function for the parameter; see Links for more choices and for general information.
imethod See CommonVGAMffArguments. Valid values are 1, 2, 3 or 4.

Details

The Felix distribution is an important basic Lagrangian distribution. The density function is

\[ f(y; a) = \frac{1}{((y - 1)/2)!} y^{(y-3)/2} a^{(y-1)/2} \exp(-ay) \]

where \( y = 1, 3, 5, \ldots \) and \( 0 < a < 0.5 \). The mean is \( 1/(1 - 2a) \) (returned as the fitted values). Fisher scoring is implemented.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Author(s)

T. W. Yee

References


See Also
dfelix, borel.tanner.

Examples

```r
fdata <- data.frame(y = 2 * rpois(n = 200, 1) + 1)  # Not real data!
fit <- vglm(y~1, felix, data = fdata, trace = TRUE, crit = "coef")
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)
```
**Description**

Maximum likelihood estimation of the (2-parameter) F distribution.

**Usage**

```r
fff(link = "loge", idf1 = NULL, idf2 = NULL, nsimEIM = 100,
imethod = 1, zero = NULL)
```

**Arguments**

- `link` Parameter link function for both parameters. See [Links](#) for more choices. The default keeps the parameters positive.
- `idf1`, `idf2` Numeric and positive. Initial value for the parameters. The default is to choose each value internally.
- `nsimEIM`, `zero` See [CommonVGAMffArguments](#) for more information.
- `imethod` Initialization method. Either the value 1 or 2. If both fail try setting values for `idf1` and `idf2`.

**Details**

The F distribution is named after Fisher and has a density function that has two parameters, called `df1` and `df2` here. This function treats these degrees of freedom as *positive reals* rather than integers. The mean of the distribution is $\frac{df2}{(df2 - 2)}$ provided $df2 > 2$, and its variance is $\frac{2df2^2(df1 + df2 - 2)}{(df1(df2 - 2)^2(df2 - 4))}$ provided $df2 > 4$. The estimated mean is returned as the fitted values. Although the F distribution can be defined to accommodate a non-centrality parameter `ncp`, it is assumed zero here. Actually it shouldn’t be too difficult to handle any known `ncp`; something to do in the short future.

**Value**

An object of class "vglmff" (see [vglmff-class](#)). The object is used by modelling functions such as `vglm` and `vgam`.

**Warning**

Numerical problems will occur when the estimates of the parameters are too low or too high.

**Author(s)**

T. W. Yee
References


See Also

FDist.

Examples

```r
## Not run:
fdata <- data.frame(x2 = runif(nn <- 2000))
fdata <- transform(fdata, df1 = exp(2+0.5*x2),
                   df2 = exp(2-0.5*x2))
fdata <- transform(fdata, y = rf(nn, df1, df2))
fit <- vglm(y ~ x2, fff, data = fdata, trace = TRUE)
coef(fit, matrix = TRUE)
```

## End(Not run)

fill

*Creates a Matrix of Appropriate Dimension*

Description

A support function for the argument \( x_{ij} \), it generates a matrix of an appropriate dimension.

Usage

```r
fill(x, values = 0, ncolx = ncol(x))
```

Arguments

- **x**: A vector or matrix which is used to determine the dimension of the answer, in particular, the number of rows. After converting \( x \) to a matrix if necessary, the answer is a matrix of values values, of dimension \( \text{nrow}(x) \) by \( \text{ncolx} \).
- **values**: Numeric. The answer contains these values, which are recycled *columnwise* if necessary, i.e., as \( \text{matrix}(\text{values}, \ldots, \text{byrow}=\text{TRUE}) \).
- **ncolx**: The number of columns of the returned matrix. The default is the number of columns of \( x \).
Details

The \texttt{xij} argument for \texttt{vglm} allows the user to input variables specific to each linear/additive predictor. For example, consider the bivariate logit model where the first/second linear/additive predictor is the logistic regression of the first/second binary response respectively. The third linear/additive predictor is \( \log(OR) = \eta_3 \), where OR is the odds ratio. If one has ocular pressure as a covariate in this model then \texttt{xij} is required to handle the ocular pressure for each eye, since these will be different in general. [This contrasts with a variable such as age, the age of the person, which has a common value for both eyes.] In order to input these data into \texttt{vglm} one often finds that functions \texttt{fill, fill1, etc.} are useful.

All terms in the \texttt{xij} and \texttt{formula} arguments in \texttt{vglm} must appear in the \texttt{form2} argument too.

Value

\texttt{matrix(values, nrow=nrow(x), ncol=ncol(x)), i.e., a matrix consisting of values values, with the number of rows matching x, and the default number of columns is the number of columns of x.}

Note

The effect of the \texttt{xij} argument is after other arguments such as \texttt{exchangeable} and \texttt{zero}. Hence \texttt{xij} does not affect constraint matrices.

Additionally, there are currently 3 other identical \texttt{fill} functions, called \texttt{fill1, fill2} and \texttt{fill3}; if you need more then assign \texttt{fill4 = fill5 = fill1} etc. The reason for this is that if more than one \texttt{fill} function is needed then they must be unique. For example, if \( M = 4 \) then \texttt{xij = op ~ lop + rop + fill(mop) + fill(mop)} would reduce to \texttt{xij = op ~ lop + rop + fill(mop)}, whereas \texttt{xij = op ~ lop + rop + fill1(mop) + fill2(mop)} would retain all \( M \) terms, which is needed.

In Examples 1 to 3 below, the \texttt{xij} argument illustrates covariates that are specific to a linear predictor. Here, \texttt{lop/rop} are the ocular pressures of the left/right eye in an artificial dataset, and \texttt{mop} is their mean. Variables \texttt{leye} and \texttt{reye} might be the presence/absence of a particular disease on the LHS/RHS eye respectively.

In Example 3, the \texttt{xij} argument illustrates fitting the (exchangeable) model where there is a common smooth function of the ocular pressure. One should use regression splines since \texttt{s} in \texttt{vgam} does not handle the \texttt{xij} argument. However, regression splines such as \texttt{bs} and \texttt{ns} need to have the same basis functions here for both functions, and Example 3 illustrates a trick involving a function \texttt{BS} to obtain this, e.g., same knots. Although regression splines create more than a single column per term in the model matrix, \texttt{fill(BS(lop, rop))} creates the required (same) number of columns.

Author(s)

T. W. Yee

See Also

\texttt{vglm.control, vglm, multinomial, Select.}
Examples

```r
fill(runif(5))
fill(runif(5, ncol = 3))
fill(runif(5), val = 1, ncol = 3)
```

# Generate eyes data for the examples below. Eyes are independent (OR=1).

```r
nn <- 1000  # Number of people
eyesdata <- data.frame(lop = round(runif(nn), 2),
                      rop = round(runif(nn), 2),
                      age = round(rnorm(nn, 40, 10)))
eyesdata <- transform(eyesdata,  
                      mop = (lop + rop) / 2,  # Mean ocular pressure  
                      op = (lop + rop) / 2,  # Value unimportant unless plotting  
                      # op = lop,
                      # Choose this if plotting  
                      etal = 0 - 2*lop + 0.04*age,  # Linear predictor for left eye  
                      etar = 0 - 2*rop + 0.04*age)  # Linear predictor for right eye  
eyesdata <- transform(eyesdata,  
                      leye = rbinom(nn, size = 1, prob = logit(eta1, inverse = TRUE)),
                      reye = rbinom(nn, size = 1, prob = logit(eta2, inverse = TRUE)))
```

# Example 1

# All effects are linear

```r
fit1 <- vglm(cbind(leye, reye) ~ op + age,
              family = binom2.or(exchangeable = TRUE, zero = 3),
              data = eyesdata, trace = TRUE,
              xij = list(op ~ lop + rop + fill(lop)),
              form2 = ~ op + lop + rop + fill(lop) + age)
head(model.matrix(fit1, type = "lm"))  # LM model matrix
head(model.matrix(fit1, type = "vlm"))  # Big VLM model matrix
coef(fit1)
coef(fit1, matrix = TRUE)  # Unchanged with 'xij'
constraints(fit1)
max(abs(predict(fit1) - predict(fit1, new = eyesdata)))  # Predicts correctly
summary(fit1)
## Not run:
plotvgam(fit1, se = TRUE)  # Wrong, e.g., because it plots against op, not lop.
# So set op = lop in the above for a correct plot.
## End(Not run)
```

# Example 2

# Model OR as a linear function of mop

```r
fit2 <- vglm(cbind(leye, reye) ~ op + age, data = eyesdata, trace = TRUE,  
              binom2.or(exchangeable = TRUE, zero = NULL),
              xij = list(op ~ lop + rop + mop),
              form2 = ~ op + lop + rop + mop + age)
head(model.matrix(fit2, type = "lm"))  # LM model matrix
head(model.matrix(fit2, type = "vlm"))  # Big VLM model matrix
coef(fit2)
coef(fit2, matrix = TRUE)  # Unchanged with 'xij'
```
max(abs(predict(fit2) - predict(fit2, new = eyesdata))) # Predicts correctly
summary(fit2)
## Not run:
plotvgam(fit2, se = TRUE) # Wrong because it plots against op, not lop.
## End(Not run)

# Example 3. This model uses regression splines on ocular pressure.
# It uses a trick to ensure common basis functions.
BS <- function(x, ...)
sm.bs(c(x,...), df = 3)[1:length(x), , drop = FALSE] # trick

fit3 <- vglm(cbind(leye, reye) ~ BS(lop, rop) + age,
              family = binom2.or(exchangeable = TRUE, zero = 3),
              data = eyesdata, trace = TRUE,
              xij = list(BS(lop,rop) - BS(lop,rop) +
                         BS(rop,lop) +
                         fill(BS(lop,rop))),
              form2 = BS(lop,rop) + BS(rop,lop) + fill(BS(lop,rop)) +
                      lop + rop + age)
head(model.matrix(fit3, type = "lm")) # LM model matrix
head(model.matrix(fit3, type = "vlm")) # Big VLM model matrix
coef(fit3)
coef(fit3, matrix = TRUE)
summary(fit3)
fit3@smart.prediction
max(abs(predict(fit3) - predict(fit3, new = eyesdata))) # Predicts correctly
predict(fit3, new = head(eyesdata)) # Note the 'scalar' OR, i.e., zero=3
max(abs(head(predict(fit3)) -
       predict(fit3, new = head(eyesdata)))) # Should be 0
## Not run:
plotvgam(fit3, se = TRUE, xlab = "lop") # Correct
## End(Not run)

# Example 4. Capture-recapture model with ephemeral and enduring
# memory effects. Similar to Yang and Chao (2005), Biometrics.
deermice <- transform(deermice, Lag1 = y1)
M.tbh.lag1 <-
vglm(cbind(y1, y2, y3, y4, y5, y6) ~ sex + weight + Lag1,
     posbernoulli.tb(parallel.t = FALSE ~ 0,
                    parallel.b = FALSE ~ 0,
                    drop.b = FALSE ~ 1),
     xij = list(Lag1 ~ fill(y1) + fill(y2) + fill(y3) + fill(y4) +
               fill(y5) + fill(y6) +
               y1 + y2 + y3 + y4 + y5),
     form2 = ~ sex + weight + Lag1 +
               fill(y1) + fill(y2) + fill(y3) + fill(y4) +
               fill(y5) + fill(y6) +
               y1 + y2 + y3 + y4 + y5 + y6,
     data = deermice, trace = TRUE)
Description

A data frame of a toxicity trial.

Usage

data(finney44)

Format

A data frame with 6 observations on the following 3 variables.

- pconc: a numeric vector, percent concentration of pyrethrins.
- hatched: number of eggs that hatched.
- unhatched: number of eggs that did not hatch.

Details

Finney (1944) describes a toxicity trial of five different concentrations of pyrethrins (percent) plus a control that were administered to eggs of *Ephestia kuhniella*. The natural mortality rate is large, and a common adjustment is to use Abbott’s formula.

References


Examples

data(finney44)
transform(finney44, mortality = unhatched / (hatched + unhatched))
Description

Computes the Fisher Z transformation, including its inverse and the first two derivatives.

Usage

```r
fisherz(theta, bminvalue = NULL, bmaxvalue = NULL,
        inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)
```

Arguments

- `theta`: Numeric or character. See below for further details.
- `bminvalue`, `bmaxvalue`: Optional boundary values. Values of `theta` which are less than or equal to \(-1\) can be replaced by `bminvalue` before computing the link function value. Values of `theta` which are greater than or equal to 1 can be replaced by `bmaxvalue` before computing the link function value. See `Links`.
- `inverse`, `deriv`, `short`, `tag`: Details at `Links`.

Details

The `fisherz` link function is commonly used for parameters that lie between \(-1\) and 1. Numerical values of `theta` close to \(-1\) or 1 or out of range result in `Inf`, `-Inf`, `NA` or `NaN`.

Value

For `deriv = 0`, \(0.5 \times \log((1+\theta)/(1-\theta))\) (same as `atanh(theta)`) when `inverse = FALSE`, and if `inverse = TRUE` then \((\exp(2*\theta)-1)/(\exp(2*\theta)+1)\) (same as `tanh(theta)`).

For `deriv = 1`, then the function returns \(d \theta / d \eta\) as a function of \(\theta\) if `inverse = FALSE`, else if `inverse = TRUE` then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base \(e\).

Note

Numerical instability may occur when `theta` is close to \(-1\) or 1. One way of overcoming this is to use, e.g., `bminvalue`.

The link function `rhobit` is very similar to `fisherz`, e.g., just twice the value of `fisherz`. This link function may be renamed to `atanhlink` in the near future.

Author(s)

Thomas W. Yee
S. Fisk

References

See Also
Links, rhobit, atanh, logit.

Examples

```r
theta <- seq(-0.99, 0.99, by = 0.01)
y <- fisherz(theta)
## Not run: plot(theta, y, type = "l", las = 1, ylab = "",
## main = "fisherz(theta)”, col = “blue")
abline(v = (-1):1, h = 0, lty = 2, col = "gray")
## End(Not run)

x <- c(seq(-1.02, -0.98, by = 0.01), seq(0.97, 1.02, by = 0.01))
fisherz(x) # Has NAs
fisherz(x, bminvalue = -1 + .Machine$double.eps,
       bmaxvalue = 1 - .Machine$double.eps) # Has no NAs
```

Fisk

The Fisk Distribution

Description
Density, distribution function, quantile function and random generation for the Fisk distribution with shape parameter `shape` and scale parameter `scale`.

Usage
```
dfisk(x, scale = 1, shape1.a, log = FALSE)
pfisk(q, scale = 1, shape1.a, lower.tail = TRUE, log.p = FALSE)
qfisk(p, scale = 1, shape1.a, lower.tail = TRUE, log.p = FALSE)
rfisk(n, scale = 1, shape1.a)
```

Arguments
- `x, q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If `length(n) > 1` then the length is taken to be the number required.
- `shape1.a` shape parameter.
- `scale` scale parameter.
- `log` Logical. If `log = TRUE` then the logarithm of the density is returned.
- `lower.tail, log.p` Same meaning as in `pnorm` or `qnorm`.

The Fisk Distribution

```
theta <- seq(-0.99, 0.99, by = 0.01)
y <- fisherz(theta)
## Not run: plot(theta, y, type = "l", las = 1, ylab = "",
## main = "fisherz(theta)”, col = “blue")
abline(v = (-1):1, h = 0, lty = 2, col = "gray")
## End(Not run)

x <- c(seq(-1.02, -0.98, by = 0.01), seq(0.97, 1.02, by = 0.01))
fisherz(x) # Has NAs
fisherz(x, bminvalue = -1 + .Machine$double.eps,
       bmaxvalue = 1 - .Machine$double.eps) # Has no NAs
```
Details
See \texttt{fisk}, which is the \texttt{VGAM} family function for estimating the parameters by maximum likelihood estimation.

Value
\texttt{dfisk} gives the density, \texttt{pfisk} gives the distribution function, \texttt{qfisk} gives the quantile function, and \texttt{rfisk} generates random deviates.

Note
The Fisk distribution is a special case of the 4-parameter generalized beta II distribution.

Author(s)
T. W. Yee and Kai Huang

References

See Also
\texttt{fisk, genbetaII}.

Examples
\begin{verbatim}
  fdata <- data.frame(y = rfisk(n = 1000, shape = exp(1), scale = exp(2)))
  fit <- vglm(y ~ 1, fisk(lss = FALSE), data = fdata, trace = TRUE)
  coef(fit, matrix = TRUE)
  Coef(fit)
\end{verbatim}
Arguments

lss
See CommonVGAMffArguments for important information.

lshape1.a, lscale
Parameter link functions applied to the (positive) parameters \(a\) and scale. See Links for more choices.

iscale, ishape1.a, imethod, zero
See CommonVGAMffArguments for information. For imethod = 2 a good initial value for iscale is needed to obtain a good estimate for the other parameter.

gscale, gshape1.a
See CommonVGAMffArguments for information.

probs.y
See CommonVGAMffArguments for information.

Details

The 2-parameter Fisk (aka log-logistic) distribution is the 4-parameter generalized beta II distribution with shape parameter \(q = p = 1\). It is also the 3-parameter Singh-Maddala distribution with shape parameter \(q = 1\), as well as the Dagum distribution with \(p = 1\). More details can be found in Kleiber and Kotz (2003).

The Fisk distribution has density
\[
f(y) = ay^{a-1}/[b^a\{1 + (y/b)^a\}^2]
\]
for \(a > 0\), \(b > 0\), \(y \geq 0\). Here, \(b\) is the scale parameter scale, and \(a\) is a shape parameter. The cumulative distribution function is
\[
F(y) = 1 - [1 + (y/b)^a]^{-1} = [1 + (y/b)^{-a}]^{-1}.
\]

The mean is
\[
E(Y) = b \Gamma(1 + 1/a) \Gamma(1 - 1/a)
\]
provided \(a > 1\); these are returned as the fitted values. This family function handles multiple responses.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

See the notes in genbetaII.

Author(s)

T. W. Yee

References

See Also

fisk, genbetaII, betaII, dagum, sinmad, inv.lomax, lomax, paralogistic, inv.paralogistic, simulate.vlm.

Examples

fdata <- data.frame(y = rfisk(n = 200, shape = exp(1), scale = exp(2)))
fit <- vglm(y ~ 1, fisk(lss = FALSE), data = fdata, trace = TRUE)
fit <- vglm(y ~ 1, fisk(ishape1.a = exp(2)), data = fdata, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)

fittedvlm

Fitted Values of a VLM object

Description

Extractor function for the fitted values of a model object that inherits from a vector linear model (VLM), e.g., a model of class "vglm".

Usage

fittedvlm(object, drop = FALSE, type.fitted = NULL, ...)

Arguments

object a model object that inherits from a VLM.
drop Logical. If FALSE then the answer is a matrix. If TRUE then the answer is a vector.
type.fitted Character. Some VGAM family functions have a type.fitted argument. If so then a different type of fitted value can be returned. It is recomputed from the model after convergence. Note: this is an experimental feature and not all VGAM family functions have this implemented yet.

Details

The “fitted values” usually corresponds to the mean response, however, because the VGAM package fits so many models, this sometimes refers to quantities such as quantiles. The mean may even not exist, e.g., for a Cauchy distribution.

Note that the fitted value is output from the @linkinv slot of the VGAM family function, where the eta argument is the $n \times M$ matrix of linear predictors.

Value

The fitted values evaluated at the final IRLS iteration.
Note

This function is one of several extractor functions for the VGAM package. Others include coef, deviance, weights and constraints etc. This function is equivalent to the methods function for the generic function fitted.values.

If fit is a VLM or VGLM then fitted(fit) and predict(fit, type = "response") should be equivalent (see predictvglm). The latter has the advantage in that it handles a newdata argument so that the fitted values can be computed for a different data set.

Author(s)

Thomas W. Yee

References


See Also

fitted, predictvglm, vglmff-class.

Examples

# Categorical regression example 1
pneumo <- transform(pneumo, let = log(exposure.time))
(fit1 <- vglm(cbind(normal, mild, severe) ~ let, propodds, data = pneumo))
fitted(fit1)

# LMS quantile regression example 2
fit2 <- vgam(BMI ~ s(age, df = c(4, 2)),
              lms.bcn(zero = 1), data = bmi.nz, trace = TRUE)
head(predict(fit2, type = "response")) # Equal to the the following two:
head(fitted(fit2))
predict(fit2, type = "response", newdata = head(bmi.nz))

# Zero-inflated example 3
zdata <- data.frame(x2 = runif(nn <- 1000))
zdata <- transform(zdata, pstr0.3 = logit(-0.5),
                    lambda.3 = loge(-0.5 + 2*x2, inverse = TRUE))
zdata <- transform(zdata, y1 = rzipois(nn, lambda = lambda.3, pstr0 = pstr0.3))
fit3 <- vglm(y1 ~ x2, zipoisson (zero = NULL), data = zdata, crit = "coeff")
head(fitted(fit3, type.fitted = "mean") ) # E(Y), which is the default
head(fitted(fit3, type.fitted = "pobs0") ) # P(Y = 0)
head(fitted(fit3, type.fitted = "pstr0") ) # Prob of a structural 0
head(fitted(fit3, type.fitted = "onempstr0") ) # 1 - prob of a structural 0
flourbeetle  

**Mortality of Flour Beetles from Carbon Disulphide**

**Description**

The `flourbeetle` data frame has 8 rows and 4 columns. Two columns are explanatory, the other two are responses.

**Usage**

```r
data(flourbeetle)
```

**Format**

This data frame contains the following columns:

- `logdose` - `log10` applied to CS2mgL.
- `CS2mgL` - a numeric vector, the concentration of gaseous carbon disulphide in mg per litre.
- `exposed` - a numeric vector, counts; the number of beetles exposed to the poison.
- `killed` - a numeric vector, counts; the numbers killed.

**Details**

These data were originally given in Table IV of Bliss (1935) and are the combination of two series of toxicological experiments involving *Tribolium confusum*, also known as the flour beetle. Groups of such adult beetles were exposed for 5 hours of gaseous carbon disulphide at different concentrations, and their mortality measured.

**Source**


**See Also**

- `binomialff`
- `probit`

**Examples**

```r
fit1 <- vglm(cbind(killed, exposed - killed) ~ logdose, binomialff(link = probit),
            data = flourbeetle, trace = TRUE)
summary(fit1)
```
The Folded-Normal Distribution

Description
Density, distribution function, quantile function and random generation for the (generalized) folded-normal distribution.

Usage

dfoldnorm(x, mean = 0, sd = 1, a1 = 1, a2 = 1, log = FALSE)
pfoldnorm(q, mean = 0, sd = 1, a1 = 1, a2 = 1,
lower.tail = TRUE, log.p = FALSE)
qfoldnorm(p, mean = 0, sd = 1, a1 = 1, a2 = 1,
lower.tail = TRUE, log.p = FALSE, ...)
rfoldnorm(n, mean = 0, sd = 1, a1 = 1, a2 = 1)

Arguments

x, q      vector of quantiles.
p        vector of probabilities.
n      number of observations. Same as rnorm.
mean, sd     see rnorm.
a1, a2     see foldnormal.
log       Logical. If TRUE then the log density is returned.
lower.tail, log.p
        Same meaning as in pnorm or qnorm.
...      Arguments that can be passed into uniroot.

Details
See foldnormal, the VGAM family function for estimating the parameters, for the formula of the probability density function and other details.

Value
dfoldnorm gives the density, pfoldnorm gives the distribution function, qfoldnorm gives the quantile function, and rfoldnorm generates random deviates.

Note
qfoldnorm() runs very slowly because it calls uniroot for each value of the argument p. The solution is consequently not exact; the ... can be used to obtain a more accurate solution if necessary.
foldnormal

Folded Normal Distribution Family Function

Description

Fits a (generalized) folded (univariate) normal distribution.

Usage

foldnormal(lmean = "identitylink", lsd = "loge", imean = NULL, isd = NULL, 
a1 = 1, a2 = 1, nsimEIM = 500, imethod = 1, zero = NULL)

Arguments

lmean, lsd Link functions for the mean and standard deviation parameters of the usual univariate normal distribution. They are \( \mu \) and \( \sigma \) respectively. See Links for more choices.

imean, isd Optional initial values for \( \mu \) and \( \sigma \). A NULL means a value is computed internally. See CommonVGAMffArguments.

a1, a2 Positive weights, called \( a_1 \) and \( a_2 \) below. Each must be of length 1.

nsimEIM, imethod, zero See CommonVGAMffArguments.
Details

If a random variable has an ordinary univariate normal distribution then the absolute value of that random variable has an ordinary **folded normal distribution**. That is, the sign has not been recorded; only the magnitude has been measured.

More generally, suppose $X$ is normal with mean $\mu$ and standard deviation $\sigma$. Let $Y = \max(a_1 X, -a_2 X)$ where $a_1$ and $a_2$ are positive weights. This means that $Y = a_1 X$ for $X > 0$, and $Y = a_2 X$ for $X < 0$. Then $Y$ is said to have a **generalized folded normal distribution**. The ordinary folded normal distribution corresponds to the special case $a_1 = a_2 = 1$.

The probability density function of the ordinary folded normal distribution can be written $dnorm(y, \text{mean}, \text{sd}) + dnorm(y, \text{mean}, \log(\text{sd}))$ for $y \geq 0$. By default, mean and log(sd) are the linear/additive predictors. Having mean=0 and sd=1 results in the half-normal distribution. The mean of an ordinary folded normal distribution is

$$E(Y) = \sigma \sqrt{2/\pi} \exp(-\mu^2/(2\sigma^2)) + \mu [1 - 2\Phi(-\mu/\sigma)]$$

and these are returned as the fitted values. Here, $\Phi()$ is the cumulative distribution function of a standard normal ($\text{pnorm}$).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

Under- or over-flow may occur if the data is ill-conditioned. It is recommended that several different initial values be used to help avoid local solutions.

Note

The response variable for this family function is the same as uninormal except positive values are required. Reasonably good initial values are needed. Fisher scoring using simulation is implemented.

See CommonVGAMffArguments for general information about many of these arguments.

Yet to do: implement the results of Johnson (1962) which gives expressions for the EIM, albeit, under a different parameterization. Also, one element of the EIM appears to require numerical integration.

Author(s)

Thomas W. Yee

References


See Also
rfoldnorm, uninormal, dnorm, skewnormal.

Examples
```r
## Not run
m <- 2; SD <- exp(1)
ddata <- data.frame(y = rfoldnorm(n <- 1000, m = m, sd = SD))
hist(with(ddata, y), prob = TRUE, main = paste("foldnormal(m = ", m,
" , sd = ", round(SD, 2), ")"))
fit <- vglm(y ~ 1, foldnormal, data = ddata, trace = TRUE)
coef(fit, matrix = TRUE)
(Cfit <- Coef(fit))
## Add the fit to the histogram:
mygrid <- with(ddata, seq(min(y), max(y), len = 200))
lines(mygrid, dfoldnorm(mygrid, Cfit[1], Cfit[2]), col = "orange")

## End(Not run)
```

Description
Computes the folded square root transformation, including its inverse and the first two derivatives.

Usage
```r
foldsqrt(theta, min = 0, max = 1, mux = sqrt(2),
inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)
```

Arguments
- **theta**: Numeric or character. See below for further details.
- **min**, **max**, **mux**: These are called \( L, U \) and \( K \) below.
- **inverse**, **deriv**, **short**, **tag**
  - Details at Links.

Details
The folded square root link function can be applied to parameters that lie between \( L \) and \( U \) inclusive. Numerical values of \( \theta \) out of range result in NA or NaN.

Value
For \( \text{foldsqrt with deriv = 0: } K(\sqrt{\theta} - L - \sqrt{U - \theta}) \) or \( \text{mux} \times (\text{sqrt(theta-min)} - \text{sqrt(max-theta)}) \) when \( \text{inverse = FALSE} \), and if \( \text{inverse = TRUE} \) then some more complicated function that returns a NA unless \( \text{theta is between -mux*sqrt(max-min) and mux*sqrt(max-min)} \).

For \( \text{deriv = 1, then the function returns } d \theta/deta \) as a function of \( \theta \) if \( \text{inverse = FALSE} \), else if \( \text{inverse = TRUE} \) then it returns the reciprocal.
Note

The default has, if theta is 0 or 1, the link function value is -sqrt(2) and +sqrt(2) respectively. These are finite values, therefore one cannot use this link function for general modelling of probabilities because of numerical problem, e.g., with binomialff, cumulative. See the example below.

Author(s)

Thomas W. Yee

See Also

Links.

Examples

```r
p <- seq(0.01, 0.99, by = 0.01)
foldsqrt(p)
max(abs(foldsqrt(foldsqrt(p), inverse = TRUE) - p)) # Should be 0

p <- c(seq(-0.02, 0.02, by = 0.01), seq(0.97, 1.02, by = 0.01))
foldsqrt(p) # Has NAs

## Not run:
p <- seq(0.01, 0.99, by = 0.01)
par(mfrow = c(2, 2), lwd = (mylwd <- 2))
y <- seq(-4, 4, length = 100)
for (d in 0:1) {
  matplot(p, cbind(logit(p, deriv = d), foldsqrt(p, deriv = d)),
         type = "n", col = "purple", ylab = "transformation", las = 1,
         main = if (d == 0) "Some probability link functions"
                      else "First derivative")
  lines(p, logit(p, deriv = d), col = "limegreen")
  lines(p, probit(p, deriv = d), col = "purple")
  lines(p, cloglog(p, deriv = d), col = "chocolate")
  lines(p, foldsqrt(p, deriv = d), col = "tan")
  if (d == 0) {
    abline(v = 0.5, h = 0, lty = "dashed")
    legend(0, 4.5, c("logit", "probit", "cloglog", "foldsqrt"), lwd = 2,
           col = c("limegreen","purple","chocolate", "tan"))
  } else
    abline(v = 0.5, lty = "dashed")
}

for (d in 0) {
  matplot(y, cbind(logit(y, deriv = d, inverse = TRUE),
                 foldsqrt(y, deriv = d, inverse = TRUE)),
          type = "n", col = "purple", xlab = "transformation", ylab = "p",
          lwd = 2, las = 1,
          main = if (d == 0) "Some inverse probability link functions"
                      else "First derivative")
  lines(y, logit(y, deriv = d, inverse = TRUE), col = "limegreen")

```
Frank

Frank's Bivariate Distribution

Description
Density, distribution function, and random generation for the (one parameter) bivariate Frank distribution.

Usage

d bifrankcop(x1, x2, apar, log = FALSE)
p bifrankcop(q1, q2, apar)
r bifrankcop(n, apar)

Arguments

x1, x2, q1, q2 vector of quantiles.
n number of observations. Same as in runif.
The Frechet Distribution

Description

Density, distribution function, quantile function and random generation for the three parameter Frechet distribution.
Usage

dfrechet(x, location = 0, scale = 1, shape, log = FALSE)
pfrechet(q, location = 0, scale = 1, shape, 
  lower.tail = TRUE, log.p = FALSE)
qfrechet(p, location = 0, scale = 1, shape, 
  lower.tail = TRUE, log.p = FALSE)
rfrechet(n, location = 0, scale = 1, shape)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. Passed into runif.
location, scale, shape the location parameter \(a\), scale parameter \(b\), and shape parameter \(s\).
log Logical. If \(log = \text{TRUE}\) then the logarithm of the density is returned.
lower.tail, log.p Same meaning as in punif or qunif.

Details

See frechet, the VGAM family function for estimating the 2 parameters (without location parameter) by maximum likelihood estimation, for the formula of the probability density function and range restrictions on the parameters.

Value

dfrechet gives the density, pfrechet gives the distribution function, qfrechet gives the quantile function, and rfrechet generates random deviates.

Author(s)

T. W. Yee and Kai Huang

References


See Also

frechet.
Examples

```r
# Not run: shape <- 5
x <- seq(-0.1, 3.5, length = 401)
plot(x, dfrechet(x, shape = shape), type = "l", ylab = "", las = 1,
     main = "Frechet density divided into 10 equal areas; orange = cdf")
abline(h = 0, col = "blue", lty = 2)
qq <- qfrechet(seq(0.1, 0.9, by = 0.1), shape = shape)
lines(qq, dfrechet(qq, shape = shape), col = "purple", lty = 3, type = "h")
lines(x, pfrechet(q = x, shape = shape), col = "orange")
```

## frechet

Frechet Distribution Family Function

Description

Maximum likelihood estimation of the 2-parameter Frechet distribution.

Usage

```r
frechet(location = 0, lscale = "loge", lshape = logoff(offset = -2),
         iscale = NULL, ishape = NULL, nsimEIM = 250, zero = NULL)
```

Arguments

- `location`: Numeric. Location parameter. It is called \( a \) below.
- `lscale`, `lshape`: Link functions for the parameters; see `Links` for more choices.
- `iscale`, `ishape`, `zero`, `nsimEIM`: See `CommonVGAMffArguments` for information.

Details

The (3-parameter) Frechet distribution has a density function that can be written

\[
f(y) = \frac{sb}{(y-a)^2}[b/(y-a)]^{s-1} \exp[-(b/(y-a))^s]
\]

for \( y > a \) and scale parameter \( b > 0 \). The positive shape parameter is \( s \). The cumulative distribution function is

\[
F(y) = \exp[-(b/(y-a))^s].
\]

The mean of \( Y \) is \( a + b\Gamma(1-1/s) \) for \( s > 1 \) (these are returned as the fitted values). The variance of \( Y \) is \( b^2[\Gamma(2-1/s) - \Gamma^2(1-1/s)] \) for \( s > 2 \).

Family `frechet` has a known, and \( \log(b) \) and \( \log(s-2) \) are the default linear/additive predictors. The working weights are estimated by simulated Fisher scoring.
Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

Family function frechet may fail for low values of the shape parameter, e.g., near 2 or lower.

Author(s)

T. W. Yee

References


See Also

rfrechet, gev.

Examples

```r
## Not run:
set.seed(123)
ndata <- data.frame(y1 = rfrechet(nn = 1000, shape = 2 + exp(1)))
with(ndata, hist(y1))
fit2 <- vglm(y1 ~ 1, frechet, data = ndata, trace = TRUE)
coef(fit2, matrix = TRUE)
Coef(fit2)
head(fitted(fit2))
with(ndata, mean(y1))
head(weights(fit2, type = "working"))
vcov(fit2)

## End(Not run)
```

freund61  

Freund’s (1961) Bivariate Extension of the Exponential Distribution

Description

Estimate the four parameters of the Freund (1961) bivariate extension of the exponential distribution by maximum likelihood estimation.

Usage

```r
freund61( la = "loge", lap = "loge", lb = "loge", lbp = "loge", 
    ia = NULL, iap = NULL, ib = NULL, ibp = NULL, 
    independent = FALSE, zero = NULL)
```
Arguments

la, lap, lb, lbp

Link functions applied to the (positive) parameters $\alpha$, $\alpha'$, $\beta$ and $\beta'$, respectively (the “p” stands for “prime”). See links for more choices.

ia, iap, ib, ibp

Initial value for the four parameters respectively. The default is to estimate them all internally.

independent

Logical. If TRUE then the parameters are constrained to satisfy $\alpha = \alpha'$ and $\beta = \beta'$, which implies that $y_1$ and $y_2$ are independent and each have an ordinary exponential distribution.

zero

An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,3,4}. The default is none of them.

Details

This model represents one type of bivariate extension of the exponential distribution that is applicable to certain problems, in particular, to two-component systems which can function if one of the components has failed. For example, engine failures in two-engine planes, paired organs such as peoples’ eyes, ears and kidneys. Suppose $y_1$ and $y_2$ are random variables representing the lifetimes of two components $A$ and $B$ in a two component system. The dependence between $y_1$ and $y_2$ is essentially such that the failure of the $B$ component changes the parameter of the exponential life distribution of the $A$ component from $\alpha$ to $\alpha'$, while the failure of the $A$ component changes the parameter of the exponential life distribution of the $B$ component from $\beta$ to $\beta'$.

The joint probability density function is given by

$$f(y_1, y_2) = \alpha\beta' \exp(-\beta'y_2 - (\alpha + \beta - \beta')y_1)$$

for $0 < y_1 < y_2$, and

$$f(y_1, y_2) = \beta\alpha' \exp(-\alpha'y_1 - (\alpha + \beta - \alpha')y_2)$$

for $0 < y_2 < y_1$. Here, all four parameters are positive, as well as the responses $y_1$ and $y_2$. Under this model, the probability that component $A$ is the first to fail is $\alpha/(\alpha + \beta)$. The time to the first failure is distributed as an exponential distribution with rate $\alpha + \beta$. Furthermore, the distribution of the time from first failure to failure of the other component is a mixture of $\text{Exponential}(\alpha')$ and $\text{Exponential}(\beta')$ with proportions $\beta/(\alpha + \beta)$ and $\alpha/(\alpha + \beta)$ respectively.

The marginal distributions are, in general, not exponential. By default, the linear/additive predictors are $\eta_1 = \log(\alpha)$, $\eta_2 = \log(\alpha')$, $\eta_3 = \log(\beta)$, $\eta_4 = \log(\beta')$.

A special case is when $\alpha = \alpha'$ and $\beta = \beta'$, which means that $y_1$ and $y_2$ are independent, and both have an ordinary exponential distribution with means $1/\alpha$ and $1/\beta$ respectively.

Fisher scoring is used, and the initial values correspond to the MLEs of an intercept model. Consequently, convergence may take only one iteration.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.
Note

To estimate all four parameters, it is necessary to have some data where $y_1 < y_2$ and $y_2 < y_1$. The response must be a two-column matrix, with columns $y_1$ and $y_2$. Currently, the fitted value is a matrix with two columns; the first column has values $(\alpha' + \beta)/(\alpha'(\alpha + \beta))$ for the mean of $y_1$, while the second column has values $(\beta' + \alpha)/(\beta'(\alpha + \beta))$ for the mean of $y_2$. The variance of $y_1$ is

$$\frac{\alpha^2}{\alpha' + \beta^2},$$

the variance of $y_2$ is

$$\frac{\beta^2}{\beta' + \alpha^2},$$

the covariance of $y_1$ and $y_2$ is

$$\frac{\alpha' \beta' - \alpha \beta}{\alpha' \beta' (\alpha + \beta)^2}.$$

Author(s)

T. W. Yee

References


See Also

exponential.

Examples

```r
fdata <- data.frame(y1 = rexp(nn <- 1000, rate = exp(1)))
fdata <- transform(fdata, y2 = rexp(nn, rate = exp(2)))
fit1 <- vglm(cbind(y1, y2) ~ 1, fam = freund61, data = fdata, trace = TRUE)
coef(fit1, matrix = TRUE)
Coef(fit1)
vcov(fit1)
head(fitted(fit1))
summary(fit1)

# y1 and y2 are independent, so fit an independence model
fit2 <- vglm(cbind(y1, y2) ~ 1, freund61(indep = TRUE),
             data = fdata, trace = TRUE)
coef(fit2, matrix = TRUE)
constraints(fit2)
pchisq(2 * (logLik(fit1) - logLik(fit2)), # p-value
df = df.residual(fit2) - df.residual(fit1), lower.tail = FALSE)
lrtest(fit1, fit2) # Better alternative
```
**gamma1**

1-parameter Gamma Distribution

**Description**

Estimates the 1-parameter gamma distribution by maximum likelihood estimation.

**Usage**

```r
gamma1(link = "log", zero = NULL)
```

**Arguments**

- `link` Link function applied to the (positive) `shape` parameter. See `Links` for more choices and general information.
- `zero` Details at `CommonVGAMffArguments`.

**Details**

The density function is given by

\[ f(y) = \exp(-y) \times y^{\text{shape}-1} / \Gamma(\text{shape}) \]

for \( \text{shape} > 0 \) and \( y > 0 \). Here, \( \Gamma(\text{shape}) \) is the gamma function, as in `gamma`. The mean of \( Y \) (returned as the fitted values) is \( \mu = \text{shape} \), and the variance is \( \sigma^2 = \text{shape} \).

**Value**

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

**Note**

This VGAM family function can handle a multiple responses, which is inputted as a matrix.

The parameter `shape` matches with shape in `rgamma`. The argument `rate` in `rgamma` is assumed 1 for this family function.

If `rate` is unknown use the family function `gammaR` to estimate it too.

**Author(s)**

T. W. Yee

**References**

gamma2

See Also

gammaR for the 2-parameter gamma distribution, lgamma1, lindley, simulate.vlm.

Examples

gdata <- data.frame(y = rgamma(n = 100, shape = exp(3)))
fit <- vglm(y ~ 1, gamma1, data = gdata, trace = TRUE, crit = "coef")
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)

2-parameter Gamma Distribution

Description

Estimates the 2-parameter gamma distribution by maximum likelihood estimation.

Usage

gamma2(lmu = "loge", lshape = "loge",
      imethod = 1, ishape = NULL,
      parallel = FALSE, deviance.arg = FALSE, zero = -2)

Arguments

  lmu, lshape Link functions applied to the (positive) mu and shape parameters (called \( \mu \) and \( a \) respectively). See Links for more choices.

  ishape Optional initial value for shape. A NULL means a value is computed internally. If a failure to converge occurs, try using this argument. This argument is ignored if used within cqo; see the iShape argument of qrrvgam.control instead.

  imethod An integer with value 1 or 2 which specifies the initialization method for the \( \mu \) parameter. If failure to converge occurs try another value (and/or specify a value for ishape).

  deviance.arg Logical. If TRUE, the deviance function is attached to the object. Under ordinary circumstances, it should be left alone because it really assumes the shape parameter is at the maximum likelihood estimate. Consequently, one cannot use that criterion to minimize within the IRLS algorithm. It should be set TRUE only when used with cqo under the fast algorithm.

  zero Integer valued vector, usually assigned \(-2\) or \(2\) if used at all. Specifies which of the two linear/additive predictors are modelled as an intercept only. By default, the shape parameter (after lshape is applied) is modelled as a single unknown number that is estimated. It can be modelled as a function of the explanatory variables by setting zero = NULL. A negative value means that the value is recycled, so setting \(-2\) means all shape parameters are intercept only. See CommonVGAMffArguments for more information.

  parallel Details at CommonVGAMffArguments. If parallel = TRUE then the constraint is not applied to the intercept.
Details

This distribution can model continuous skewed responses. The density function is given by

\[ f(y; \mu, a) = \exp(-ay/\mu) \times \frac{(ay/\mu)^{a-1} \times a}{\mu \times \Gamma(a)} \]

for \( \mu > 0 \), \( a > 0 \) and \( y > 0 \). Here, \( \Gamma(\cdot) \) is the gamma function, as in \texttt{gamma}. The mean of \( Y \) is \( \mu = \mu \) (returned as the fitted values) with variance \( \sigma^2 = \mu^2/a \). If \( 0 < a < 1 \) then the density has a pole at the origin and decreases monotonically as \( y \) increases. If \( a = 1 \) then this corresponds to the exponential distribution. If \( a > 1 \) then the density is zero at the origin and is unimodal with mode at \( y = \mu - \mu/a \); this can be achieved with \texttt{1shape="loglog"}.

By default, the two linear/additive predictors are \( \eta_1 = \log(\mu) \) and \( \eta_2 = \log(a) \). This family function implements Fisher scoring and the working weight matrices are diagonal.

This \texttt{VGAM} family function handles \texttt{multivariate} responses, so that a matrix can be used as the response. The number of columns is the number of species, say, and \texttt{zero=-2} means that all species have a shape parameter equalling a (different) intercept only.

Value

An object of class "\texttt{vglmff}" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm} and \texttt{vgam}.

Note

The response must be strictly positive. A moment estimator for the shape parameter may be implemented in the future.

If \( \mu \) and shape are vectors, then \texttt{rgamma(n = n, shape = shape, scale = mu/shape)} will generate random gamma variates of this parameterization, etc.; see \texttt{GammaDist}.

For \texttt{cqo} and \texttt{cao}, taking the logarithm of the response means (approximately) a \texttt{gaussian} family may be used on the transformed data.

Author(s)

T. W. Yee

References

The parameterization of this \texttt{VGAM} family function is the 2-parameter gamma distribution described in the monograph


See Also

\texttt{gamma1} for the 1-parameter gamma distribution, \texttt{gammaR} for another parameterization of the 2-parameter gamma distribution that is directly matched with \texttt{rgamma, bigamma_nckay} for a bivariate gamma distribution, \texttt{expexpf, GammaDist, golf, CommonVGAMffArguments, simulate.vlm, negloge}.  

**Examples**

```r
# Essentially a 1-parameter gamma
ndata <- data.frame(y = rgamma(n = 100, shape = exp(1)))
fit1 <- glm(y ~ 1, gamma1, data =ndata)
fit2 <- glm(y ~ 1, gamma2, data =ndata, trace = TRUE, crit = "coef")
coef(fit2, matrix = TRUE)
c(Coef(fit2), colMeans(ndata))

# Essentially a 2-parameter gamma
ndata <- data.frame(y = rgamma(n = 500, rate = exp(-1), shape = exp(2)))
fit2 <- glm(y ~ 1, gamma2, data =ndata, trace = TRUE, crit = "coef")
coef(fit2, matrix = TRUE)
c(Coef(fit2), colMeans(ndata))
summary(fit2)
```

---

### Description

Estimate the parameter of a gamma hyperbola bivariate distribution by maximum likelihood estimation.

### Usage

```r
gammahyperbola(ltheta = "loge", itheta = NULL, expected = FALSE)
```

### Arguments

- `ltheta` Link function applied to the (positive) parameter \( \theta \). See [Links](#) for more choices.
- `itheta` Initial value for the parameter. The default is to estimate it internally.
- `expected` Logical. `FALSE` means the Newton-Raphson (using the observed information matrix) algorithm, otherwise the expected information matrix is used (Fisher scoring algorithm).

### Details

The joint probability density function is given by

\[
f(y_1, y_2) = \exp(-e^{-\theta}y_1/\theta - \theta y_2)
\]

for \( \theta > 0, y_1 > 0, y_2 > 1 \). The random variables \( Y_1 \) and \( Y_2 \) are independent. The marginal distribution of \( Y_1 \) is an exponential distribution with rate parameter \( \exp(-\theta)/\theta \). The marginal distribution of \( Y_2 \) is an exponential distribution that has been shifted to the right by 1 and with rate parameter \( \theta \). The fitted values are stored in a two-column matrix with the marginal means, which are \( \theta \exp(\theta) \) and \( 1 + 1/\theta \).

The default algorithm is Newton-Raphson because Fisher scoring tends to be much slower for this distribution.
Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

Note

The response must be a two-column matrix.

Author(s)

T. W. Yee

References


See Also

`exponential`.

Examples

```r
#Generate gammar data
x2 <- runif(nn <- 1000)
gdata <- data.frame(x2 = runif(nn <- 1000))
gdata <- transform(gdata, theta = exp(-2 + x2))
gdata <- transform(gdata, y1 = rexp(nn, rate = exp(-theta)/theta), y2 = rexp(nn, rate = theta + 1))

#Estimate gammar
fit <- vglm(cbind(y1, y2) ~ x2, gammahyperbola(expected = TRUE), data = gdata)
coef(fit, matrix = TRUE)
Coef(fit)
head(fitted(fit))
summary(fit)
```

---

gammaR

2-parameter Gamma Distribution

Description

Estimates the 2-parameter gamma distribution by maximum likelihood estimation.

Usage

```r
gammaR(lrate = "loge", lshape = "loge", irate = NULL, ishape = NULL, lss = TRUE, zero = ifelse(lss, -2, -1))
```
Arguments

lrate, lshape  Link functions applied to the (positive) rate and shape parameters. See Links for more choices.

irate, ishape  Optional initial values for rate and shape. A NULL means a value is computed internally. If a failure to converge occurs, try using these arguments.

zero, lss     Details at CommonVGAMffArguments.

Details

The density function is given by

\[ f(y) = \exp(-\text{rate} \times y) \times y^{\text{shape} - 1} \times \text{rate}^{\text{shape}} / \Gamma(\text{shape}) \]

for shape > 0, rate > 0 and y > 0. Here, \( \Gamma(\text{shape}) \) is the gamma function, as in gamma. The mean of Y is \( \mu = \text{shape}/\text{rate} \) (returned as the fitted values) with variance \( \sigma^2 = \mu^2/\text{shape} = \text{shape}/\text{rate}^2 \). By default, the two linear/additive predictors are \( \eta_1 = \log(\text{shape}) \) and \( \eta_2 = \log(\text{rate}) \).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

The parameters rate and shape match with the arguments rate and shape of rgamma. The order of the arguments agree too. Here, scale = 1/rate is used, so one can use negloge. Multiple responses are handled.

If rate = 1 use the family function gamma1 to estimate shape.

Author(s)

T. W. Yee

References

Most standard texts on statistical distributions describe the 2-parameter gamma distribution, e.g.,

See Also
gamma1 for the 1-parameter gamma distribution, gamma2 for another parameterization of the 2-parameter gamma distribution, bigamma.mckay for a bivariate gamma distribution, expexpff, simulate.vlm, rgamma, negloge.
Examples

# Essentially a 1-parameter gamma
gdata <- data.frame(y1 = rgamma(n <- 100, shape = exp(1)))
fit1 <- vglm(y1 ~ 1, gamma1, data = gdata, trace = TRUE)
fit2 <- vglm(y1 ~ 1, gammaR, data = gdata, trace = TRUE, crit = "coef")
coef(fit2, matrix = TRUE)
Coef(fit2)

# Essentially a 2-parameter gamma
gdata <- data.frame(y2 = rgamma(n = 500, rate = exp(1), shape = exp(2)))
fit2 <- vglm(y2 ~ 1, gammaR, data = gdata, trace = TRUE, crit = "coef")
coef(fit2, matrix = TRUE)
Coef(fit2)
summary(fit2)

garma

GARMA (Generalized Autoregressive Moving-Average) Models

Description

Fits GARMA models to time series data.

Usage

garma(link = "identitylink", p.ar.lag = 1, q.ma.lag = 0,
      coefstart = NULL, step = 1)

Arguments

  link        Link function applied to the mean response. The default is suitable for continuous responses. The link loge should be chosen if the data are counts. The link reciprocal can be chosen if the data are counts and the variance assumed for this is \( \mu^2 \). The links logit, probit, cloglog, and cauchit are supported and suitable for binary responses.
  Note that when the log or logit link is chosen: for log and logit, zero values can be replaced by bvalue. See loge and logit etc. for specific information about each link function.
  p.ar.lag    A positive integer, the lag for the autoregressive component. Called \( p \) below.
  q.ma.lag    A non-negative integer, the lag for the moving-average component. Called \( q \) below.
  coefstart   Starting values for the coefficients. Assigning this argument is highly recommended. For technical reasons, the argument coefstart in vglm cannot be used.
  step        Numeric. Step length, e.g., 0.5 means half-stepsizing.
Details

This function draws heavily on Benjamin et al. (1998). See also Benjamin et al. (2003). GARMA models extend the ARMA time series model to generalized responses in the exponential family, e.g., Poisson counts, binary responses. Currently, this function is rudimentary and can handle only certain continuous, count and binary responses only. The user must choose an appropriate link for the link argument.

The GARMA($p$, $q$) model is defined by firstly having a response belonging to the exponential family

\[ f(y_t|D_t) = \exp \left\{ \frac{y_t \theta_t - b(\theta_t)}{\phi} + c(y_t, \phi/A_t) \right\} \]

where $\theta_t$ and $\phi$ are the canonical and scale parameters respectively, and $A_t$ are known prior weights. The mean $\mu_t = E(Y_t|D_t) = b'(\theta_t)$ is related to the linear predictor $\eta_t$ by the link function $g$. Here, $D_t = \{x_t, \ldots, x_1, y_{t-1}, \ldots, y_1, \mu_{t-1}, \ldots, \mu_1\}$ is the previous information set. Secondly, the GARMA($p$, $q$) model is defined by

\[ g(\mu_t) = \eta_t = x_t^T \beta + \sum_{k=1}^{p} \phi_k (g(y_{t-k}) - x_{t-k}^T \beta) + \sum_{k=1}^{q} \theta_k (g(y_{t-k}) - \eta_{t-k}) \]

Parameter vectors $\beta$, $\phi$ and $\theta$ are estimated by maximum likelihood.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm.

Warning

This VGAM family function is 'non-standard' in that the model does need some coercing to get it into the VGLM framework. Special code is required to get it running. A consequence is that some methods functions may give wrong results when applied to the fitted object.

Note

This function is unpolished and is requires lots of improvements. In particular, initialization is very poor. Results appear very sensitive to quality of initial values. A limited amount of experience has shown that half-stepsizing is often needed for convergence, therefore choosing crit = "coef" is not recommended.

Overdispersion is not handled. For binomial responses it is currently best to input a vector of 1s and 0s rather than the cbind(successes, failures) because the initialize slot is rudimentary.

Author(s)

T. W. Yee
References


Examples

gdata <- data.frame(interspike = c(68, 41, 82, 66, 101, 66, 57, 41, 27, 78, 59, 73, 6, 44, 72, 66, 59, 60, 39, 52, 50, 29, 30, 56, 76, 55, 73, 104, 104, 52, 25, 33, 20, 60, 47, 6, 47, 22, 35, 30, 29, 58, 24, 34, 36, 34, 6, 19, 28, 16, 36, 33, 12, 26, 36, 39, 24, 14, 28, 13, 2, 30, 18, 17, 28, 9, 28, 20, 17, 12, 19, 18, 14, 23, 18, 22, 18, 19, 26, 27, 23, 24, 35, 22, 29, 28, 17, 30, 34, 17, 20, 49, 29, 35, 49, 25, 55, 42, 29, 16)) # See Zeger and Qaqish (1988)
gdata <- transform(gdata, spikenum = seq(interspike))
bvalue <- 0.1 # .Machine$double.xmin # Boundary value
fit <- vglm(interspike ~ 1, trace = TRUE, data = gdata,
            garma(loge(bvalue = bvalue),
                p = 2, coefstart = c(4, 0.3, 0.4)))
summary(fit)
coef(fit, matrix = TRUE)
Coef(fit) # A bug here
## Not run: with(gdata, plot(interspike, ylim = c(0, 120), las = 1,
     xlab = "Spike Number", ylab = "Inter-Spike Time (ms)", col = "blue")
with(gdata, lines(spikenum[1:fit$misc$plag], fitted(fit), col = "orange")
abline(h = mean(with(gdata, interspike)), lty = "dashed", col = "gray")
## End(Not run)

---

gaussianff

Gaussian (normal) Family Function

Description

Fits a generalized linear model to a response with Gaussian (normal) errors. The dispersion parameter may be known or unknown.

Usage

gaussianff(dispersion = 0, parallel = FALSE, zero = NULL)
Arguments

parallel A logical or formula. If a formula, the response of the formula should be a logical and the terms of the formula indicates whether or not those terms are parallel.

dispersion Dispersion parameter. If 0 then it is estimated and the moment estimate is put in \texttt{object@misc$dispersion}; it is assigned the value

\[
\sum_{i=1}^{n} (y_i - \eta_i)^T W_i (y_i - \eta_i) / (nM - p)
\]

where \( p \) is the total number of parameters estimated (for RR-VGLMs the value used is the number of columns in the large \( X \) model matrix; this may not be correct). If the argument is assigned a positive quantity then it is assumed to be known with that value.

zero An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \( \{1, 2, \ldots, M\} \) where \( M \) is the number of columns of the matrix response.

Details

This function is usually used in conjunction with \texttt{vglm}, else \texttt{vlm} is recommended instead. The notation \( M \) is used to denote the number of linear/additive predictors. This function can handle any finite \( M \), and the default is to use ordinary least squares. A vector linear/additive model can be fitted by minimizing

\[
\sum_{i=1}^{n} (y_i - \eta_i)^T W_i (y_i - \eta_i)
\]

where \( y_i \) is a \( M \)-vector, \( \eta_i \) is the vector of linear/additive predictors. The \( W_i \) is any positive-definite matrix, and the default is the order-\( M \) identity matrix. The \( W_i \) can be inputted using the \texttt{weights} argument of \texttt{vlm/vglm/vgam etc.}, and the format is the \texttt{matrix-band} format whereby it is a \( n \times A \) matrix with the diagonals are passed first, followed by next the upper band, all the way to the \( (1, M) \) element. Here, \( A \) has maximum value of \( M(M+1)/2 \) and a minimum value of \( M \). Usually the \texttt{weights} argument of \texttt{vlm/vglm/vgam/rrvgam} is just a vector, in which case each element is multiplied by a order-\( M \) identity matrix. If in doubt, type something like \texttt{weights(object, type="working")} after the model has been fitted.

Value

An object of class "\texttt{vglmff}" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, \texttt{rrvgam} and \texttt{vgam}.

Note

This \texttt{VGAM} family function is supposed to be similar to \texttt{gaussian} but is not compatible with \texttt{glm}. The "\texttt{ff}" in the name is added to avoid any masking problems.

Author(s)

Thomas W. Yee
GenbetaII

References


See Also

uninormal, huber2, lqnorm, binormal, SURff, vlm, vglm, vgam, rrvglm.

Examples

gdata <- data.frame(x2 = sort(runif(n <- 40)))
gdata <- transform(gdata, y1 = 1 + 2*x2 + rnorm(n, sd = 0.1),
                 y2 = 3 + 4*x2 + rnorm(n, sd = 0.1),
                 y3 = 7 + 4*x2 + rnorm(n, sd = 0.1))
fit <- vglm(cbind(y1,y2) ~ x2, gaussianff, data = gdata)
coef(fit, matrix = TRUE)

# For comparison:
coef( lmfit <- lm(y1 ~ x2, data = gdata))
coef(glmfit <- glm(y2 ~ x2, data = gdata, gaussian))
vcov(fit)
vcov(lmfit)

(t(weights(fit, type = "prior")) # Unweighted observations
head(weights(fit, type = "working")) # Identity matrices

# Reduced-rank VLM (rank-1)
fit2 <- rrvglm(cbind(y1, y2, y3) ~ x2, gaussianff, data = gdata)
Coef(fit2)

GenbetaII

The Generalized Beta II Distribution

Description

Density for the generalized beta II distribution with shape parameters a and p and q, and scale parameter scale.

Usage

dgenbetaII(x, scale = 1, shape1.a, shape2.p, shape3.q, log = FALSE)
Arguments

x: vector of quantiles.
shape1.a, shape2.p, shape3.q: positive shape parameters.
scale: positive scale parameter.
log: Logical. If log = TRUE then the logarithm of the density is returned.

Details

See genbetaII, which is the VGAM family function for estimating the parameters by maximum likelihood estimation. Several distributions, such as the Singh-Maddala, are special case of this flexible 4-parameter distribution. The product of shape1.a and shape2.p determines the behaviour of the density at the origin.

Value
dgenbetaII gives the density.

Author(s)
T. W. Yee

References


See Also
genbetaII.

Examples

dgenbetaII(0, shape1.a = 1/4, shape2.p = 4, shape3.q = 3)
dgenbetaII(0, shape1.a = 1/4, shape2.p = 2, shape3.q = 3)
dgenbetaII(0, shape1.a = 1/4, shape2.p = 8, shape3.q = 3)
Usage

```r
genbetaII(lscale = "loge", lshape1.a = "loge", lshape2.p = "loge", lshape3.q = "loge", iscale = NULL, ishape1.a = NULL, ishape2.p = NULL, ishape3.q = NULL, lss = TRUE, gscale = exp(-5:5), gshape1.a = exp(-5:5), gshape2.p = exp(-5:5), gshape3.q = exp(-5:5), zero = ifelse(lss, -(2:4), -c(1, 3:4)))
```

Arguments

- **lss**
  See `CommonVGAMffArguments` for important information.
- **lshape1.a, lscale, lshape2.p, lshape3.q**
  Parameter link functions applied to the shape parameter a, scale parameter scale, shape parameter p, and shape parameter q. All four parameters are positive. See `Links` for more choices.
- **iscale, ishape1.a, ishape2.p, ishape3.q**
  Optional initial values for the parameters. A NULL means a value is computed internally using the arguments gscale, gshape1.a, etc.
- **gscale, gshape1.a, gshape2.p, gshape3.q**
  See `CommonVGAMffArguments` for information. Replaced by iscale, ishape1.a etc. if given.
- **zero**
  An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The default is to set all the shape parameters to be intercept-only.

Details

This distribution is most useful for unifying a substantial number of size distributions. For example, the Singh-Maddala, Dagum, Fisk (log-logistic), Lomax (Pareto type II), inverse Lomax, beta distribution of the second kind distributions are all special cases. Full details can be found in Kleiber and Kotz (2003), and Brazauskas (2002). The argument names given here are used by other families that are special cases of this family. Fisher scoring is used here and for the special cases too.

The 4-parameter generalized beta II distribution has density

$$f(y) = ay^{ap-1}/[b^{ap} B(p, q)(1 + (y/b)^a)]^{p+q}$$

for $a > 0$, $b > 0$, $p > 0$, $q > 0$, $y \geq 0$. Here $B$ is the beta function, and $b$ is the scale parameter scale, while the others are shape parameters. The mean is

$$E(Y) = b \Gamma(p + 1/a) \Gamma(q - 1/a)/\Gamma(p) \Gamma(q)$$

provided $-ap < 1 < aq$; these are returned as the fitted values.

This family function handles multiple responses.

Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`. 
**Warning**

This distribution is very flexible and it is not generally recommended to use this family function when the sample size is small—numerical problems easily occur with small samples. Probably several hundred observations at least are needed in order to estimate the parameters with any level of confidence. Neither is the inclusion of covariates recommended at all—not unless there are several thousand observations. The mean is finite only when \(-ap < 1 < aq\), and this can be easily violated by the parameter estimates for small sample sizes. Try fitting some of the special cases of this distribution (e.g., sinmad, fisk, etc.) first, and then possibly use those models for initial values for this distribution.

**Note**

The default is to use a grid search with respect to all four parameters; this is quite costly and is time consuming. If the self-starting initial values fail, try experimenting with the initial value arguments. Also, the constraint \(-ap < 1 < aq\) may be violated as the iterations progress so it pays to monitor convergence, e.g., set `trace = TRUE`. Successful convergence depends on having very good initial values. This is rather difficult for this distribution so that a grid search is conducted by default. One suggestion for increasing the estimation reliability is to set `stepsize = 0.5` and `maxit = 100`; see `vglmNcontrol`.

**Author(s)**

T. W. Yee, with help from Victor Miranda.

**References**


**See Also**

dgenbetaII, betaII, dagum, sinmad, fisk, lomax, inv.lomax, paralogistic, inv.paralogistic, lino, CommonVGAMffArguments, vglm.control.

**Examples**

```r
## Not run:
gdata <- data.frame(y = rsinmad(3000, shape1 = exp(1), scale = exp(2), shape3 = exp(1)))  # A special case!
fit <- vglm(y ~ 1, genbetaII(lss = FALSE), data = gdata, trace = TRUE)
fit <- vglm(y ~ 1, data = gdata, trace = TRUE, genbetaII(ishape1.a = 3, iscale = 7, ishape3.q = 2.3))
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)

## End(Not run)
```
Description

Estimation of the 3-parameter generalized gamma distribution proposed by Stacy (1962).

Usage

gengamma.stacy(lscale = "loge", ld = "loge", lk = "loge",
   iscale = NULL, id = NULL, ik = NULL, zero = NULL)

Arguments

lscale, ld, lk  Parameter link function applied to each of the positive parameters b, d and k, respectively. See Links for more choices.

iscale, id, ik  Initial value for b, d and k, respectively. The defaults mean an initial value is determined internally for each.

zero  An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2,3}. The default value means none are modelled as intercept-only terms. See CommonVGAMffArguments for more information.

Details

The probability density function can be written

\[ f(y; b, d, k) = db^{-d} k y^{dk-1} \exp\left[-(y/b)^d\right] / \Gamma(k) \]

for scale parameter \( b > 0 \), and \( d > 0 \), \( k > 0 \), and \( y > 0 \). The mean of \( Y \) is \( b \times \Gamma(k + 1/d) / \Gamma(k) \) (returned as the fitted values), which equals \( bk \) if \( d = 1 \).

There are many special cases, as given in Table 1 of Stacey and Mihran (1965). In the following, the parameters are in the order \( b, d, k \). The special cases are: Exponential \( f(y; b, 1, 1) \), Gamma \( f(y; b, 1, k) \), Weibull \( f(y; b, d, 1) \), Chi Squared \( f(y; 2, 1, a/2) \) with \( a \) degrees of freedom, Chi \( f(y; \sqrt{2}, 2, a/2) \) with \( a \) degrees of freedom, Half-normal \( f(y; \sqrt{2}, 2, 1/2) \), Circular normal \( f(y; \sqrt{2}, 2, 1) \), Spherical normal \( f(y; \sqrt{2}, 2, 3/2) \), Rayleigh \( f(y; c\sqrt{2}, 2, 1) \) where \( c > 0 \).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.
Warning

Several authors have considered maximum likelihood estimation for the generalized gamma distribution and have found that the Newton-Raphson algorithm does not work very well and that the existence of solutions to the log-likelihood equations is sometimes in doubt. Although Fisher scoring is used here, it is likely that the same problems will be encountered. It appears that large samples are required, for example, the estimator of $k$ became asymptotically normal only with 400 or more observations. It is not uncommon for maximum likelihood estimates to fail to converge even with two or three hundred observations. With covariates, even more observations are needed to increase the chances of convergence. Using covariates is not advised unless the sample size is at least a few thousand.

Note

The notation used here differs from Stacy (1962) and Prentice (1974). Poor initial values may result in failure to converge so if there are covariates and there are convergence problems, try using the \texttt{zero} argument (e.g., \texttt{zero = 2:3}) or the \texttt{ik} argument.

Author(s)

T. W. Yee

References


See Also

\texttt{rgengamma.stacy}, \texttt{gammaQ}, \texttt{gammaR}, \texttt{prenticeWT}, \texttt{simulateNvlm}.

Examples

```r
k <- exp(-1); Scale = exp(1)
gdata <- data.frame(y = rgamma(1000, shape = k, scale = Scale))
fit <- vglm(y ~ 1, gengamma.stacy, data = gdata, trace = TRUE)
coef(fit, matrix = TRUE)
```
The Generalized Gamma Distribution

Description
Density, distribution function, quantile function and random generation for the generalized gamma distribution with scale parameter `scale`, and parameters `d` and `k`.

Usage

dgengamma.stacy(x, scale = 1, d = 1, k = 1, log = FALSE)
pengamma.stacy(q, scale = 1, d = 1, k = 1, lower.tail = TRUE, log.p = FALSE)
qgengamma.stacy(p, scale = 1, d = 1, k = 1, lower.tail = TRUE, log.p = FALSE)
rgengamma.stacy(n, scale = 1, d = 1, k = 1)

Arguments

- `x`, `q`: vector of quantiles.
- `p`: vector of probabilities.
- `n`: number of observations. Same as in `runif`.
- `scale`: the (positive) scale parameter `b`.
- `d`, `k`: the (positive) parameters `d` and `k`.
- `log`: Logical. If `log = TRUE` then the logarithm of the density is returned.
- `lower.tail`, `log.p`: Same meaning as in `pnorm` or `qnorm`.

Details

See `gengamma.stacy`, the VGAM family function for estimating the generalized gamma distribution by maximum likelihood estimation, for formulae and other details. Apart from `n`, all the above arguments may be vectors and are recycled to the appropriate length if necessary.

Value

dgengamma.stacy gives the density, pengamma.stacy gives the distribution function, qgengamma.stacy gives the quantile function, and rgengamma.stacy generates random deviates.

Author(s)

T. W. Yee and Kai Huang

References

genpoisson

Generalized Poisson distribution

Description

Estimation of the two-parameter generalized Poisson distribution.

Usage

```r
genpoisson(llambda = "rhobit", ltheta = "loge",
            ilambda = NULL, itheta = NULL,
            use.approx = TRUE, imethod = 1, ishrinkage = 0.95, zero = -1)
```

Arguments

- `llambda`, `ltheta`
  Parameter link functions for λ and θ. See Links for more choices. The λ parameter lies at least within the interval \([-1, 1]\]; see below for more details, and an alternative link is `rhobit`. The θ parameter is positive, therefore the default is the log link.

- `ilambda`, `itheta`
  Optional initial values for λ and θ. The default is to choose values internally.

- `use.approx`
  Logical. If TRUE then an approximation to the expected information matrix is used, otherwise Newton-Raphson is used.

- `imethod`
  An integer with value 1 or 2 or 3 which specifies the initialization method for the parameters. If failure to converge occurs try another value and/or else specify a value for `ilambda` and/or `itheta`.

- `ishrinkage`, `zero`
  See CommonVGAMffArguments for information.
Details

The generalized Poisson distribution has density

\[ f(y) = \theta (\theta + \lambda y)^{y-1} \exp(-\theta - \lambda y)/y! \]

for \( \theta > 0 \) and \( y = 0, 1, 2, \ldots \). Now \( \max(-1, -\theta/m) \leq \lambda \leq 1 \) where \( m(\geq 4) \) is the greatest positive integer satisfying \( \theta + m\lambda > 0 \) when \( \lambda < 0 \) [and then \( P(Y = y) = 0 \) for \( y > m \)]. Note the complicated support for this distribution means, for some data sets, the default link for \( \lambda \) will not always work, and some tinkering may be required to get it running.

As Consul and Famoye (2006) state on p.165, the lower limits on \( \lambda \) and \( m \geq 4 \) are imposed to ensure that there are at least 5 classes with nonzero probability when \( \lambda \) is negative.

An ordinary Poisson distribution corresponds to \( \lambda = 0 \). The mean (returned as the fitted values) is \( E(Y) = \theta/(1 - \lambda) \) and the variance is \( \theta/(1 - \lambda)^2 \).

For more information see Consul and Famoye (2006) for a summary and Consul (1989) for full details.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

Monitor convergence! This family function is fragile. Don’t get confused because theta (and not \( \lambda \)) here really matches more closely with \( \lambda \) of dpois.

Note

This family function handles multiple responses. This distribution is potentially useful for dispersion modelling. Convergence problems may occur when \( \lambda \) is very close to 0 or 1. If a failure occurs then you might want to try something like \( \lambda = \text{extlogit(min = -0.9, max = 1)} \) to handle the LHS complicated constraint, and if that doesn’t work, try \( \lambda = \text{extlogit(min = -0.8, max = 1)} \), etc.

Author(s)

T. W. Yee

References


See Also

poissonff, dpois, dgenpois, rhobit, extlogit.
Examples

gdata <- data.frame(x2 = runif(nn <- 200))
gdata <- transform(gdata, y1 = rpois(nn, exp(2 - x2)))  # Poisson data
fit <- vglm(y1 ~ x2, genpoisson, data = gdata, trace = TRUE)
coef(fit, matrix = TRUE)
summary(fit)

Description

Density, distribution function, quantile function and random generation for the generalized Rayleigh
distribution.

Usage

dgenray(x, scale = 1, shape, log = FALSE)
pgenray(q, scale = 1, shape, lower.tail = TRUE, log.p = FALSE)
qgenray(p, scale = 1, shape, lower.tail = TRUE, log.p = FALSE)
rgenray(n, scale = 1, shape)

Arguments

x, q        vector of quantiles.
p        vector of probabilities.
n        number of observations. If length(n) > 1 then the length is taken to be the
number required.
scale, shape        positive scale and shape parameters.
log        Logical. If log = TRUE then the logarithm of the density is returned.
lower.tail, log.p
        Same meaning as in pnorm or qnorm.

Details

See genrayleigh, the VGAM family function for estimating the parameters, for the formula of the
probability density function and other details.

Value

dgenray gives the density, pgenray gives the distribution function, qgenray gives the quantile
function, and rgenray generates random deviates.

Note

We define scale as the reciprocal of the scale parameter used by Kundu and Raqab (2005).
Author(s)
Kai Huang and J. G. Lauder and T. W. Yee

See Also
Genrayleigh, rayleigh.

Examples

```r
# Not run:
shape <- 0.5; Scale <- 1; nn <- 50;
x <- seq(-0.10, 3.0, len = nn)
plot(x, dgenray(x, shape, scale = Scale), type = "l", las = 1, ylim = c(0, 1.2),
ylab = paste("dgenray(shape = ", shape, ", scale = ", Scale, ")"),
col = "blue", cex.main = 0.8,
main = "Blue is density, orange is cumulative distribution function",
sub = "Purple lines are the 10, 20, ..., 90 percentiles")
lines(x, pgenray(x, shape, scale = Scale), col = "orange")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qgenray(probs, shape, scale = Scale)
lines(Q, qgenray(Q, shape, scale = Scale), col = "purple", lty = 3, type = "h")
lines(Q, qgenray(Q, shape, scale = Scale), col = "purple", lty = 3, type = "h")
abline(h = probs, col = "purple", lty = 3)
max(abs(pgenray(Q, shape, scale = Scale) - probs)) # Should be 0
```

## End(Not run)

---

### genrayleigh

**Generalized Rayleigh Distribution Family Function**

**Description**

Estimates the two parameters of the generalized Rayleigh distribution by maximum likelihood estimation.

**Usage**

```r
genrayleigh(ls_scale = "log", ls_shape = "log",
iscale = NULL, ishape = NULL,
tol12 = 1e-05, nsimEIM = 300, zero = 2)
```

**Arguments**

- `lscale`, `ls_shape` Link function for the two positive parameters, scale and shape. See [Links](#) for more choices.
- `iscale`, `ishape` Numeric. Optional initial values for the scale and shape parameters.
- `nsimEIM`, `zero` See [CommonVGAMArguments](#).
- `tol12` Numeric and positive. Tolerance for testing whether the second shape parameter is either 1 or 2. If so then the working weights need to handle these singularities.
Details

The generalized Rayleigh distribution has density function

$$f(y; b = \text{scale}, s = \text{shape}) = \left(2sy/b^2\right)e^{-\left(y/b\right)^2}\left(1 - e^{-\left(y/b\right)^2}\right)^{s-1}$$

where $y > 0$ and the two parameters, $b$ and $s$, are positive. The mean cannot be expressed nicely so the median is returned as the fitted values. Applications of the generalized Rayleigh distribution include modeling strength data and general lifetime data. Simulated Fisher scoring is implemented.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

We define scale as the reciprocal of the scale parameter used by Kundu and Raqab (2005).

Author(s)

J. G. Lauder and T. W. Yee

References


See Also

dgenray, rayleigh.

Examples

```r
Scale <- exp(1); shape <- exp(1)
rdata <- data.frame(y = rgenray(n = 1000, scale = Scale, shape = shape))
fit <- vglm(y ~ 1, genrayleigh, data = rdata, trace = TRUE)
c(with(rdata, mean(y)), head(fitted(fit), 1))
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)
```
Geometric (Truncated and Untruncated) Distributions

Description

Maximum likelihood estimation for the geometric and truncated geometric distributions.

Usage

geometric(link = "logit", expected = TRUE, imethod = 1,
          iprob = NULL, zero = NULL)

truncgeometric(upper.limit = Inf,
               link = "logit", expected = TRUE, imethod = 1,
               iprob = NULL, zero = NULL)

Arguments

link Parameter link function applied to the probability parameter \( p \), which lies in the unit interval. See Links for more choices.

expected Logical. Fisher scoring is used if expected = TRUE, else Newton-Raphson.

iprob, imethod, zero

See CommonVGAMffArguments for more details.

upper.limit Numeric. Upper values. As a vector, it is recycled across responses first. The default value means both family functions should give the same result.

Details

A random variable \( Y \) has a 1-parameter geometric distribution if \( P(Y = y) = p(1 - p)^y \) for \( y = 0, 1, 2, \ldots \). Here, \( p \) is the probability of success, and \( Y \) is the number of (independent) trials that are fails until a success occurs. Thus the response \( Y \) should be a non-negative integer. The mean of \( Y \) is \( E(Y) = (1 - p)/p \) and its variance is \( Var(Y) = (1 - p)/p^2 \). The geometric distribution is a special case of the negative binomial distribution (see negbinomial). The geometric distribution is also a special case of the Borel distribution, which is a Lagrangian distribution. If \( Y \) has a geometric distribution with parameter \( p \) then \( Y + 1 \) has a positive-geometric distribution with the same parameter. Multiple responses are permitted.

For truncgeometric(), the (upper) truncated geometric distribution can have response integer values from 0 to upper.limit. It has density \( \text{prob} \times (1 - \text{prob})^y / [1 - (1 - \text{prob})^{(1 + \text{upper.limit})}] \).

For a generalized truncated geometric distribution with integer values \( L \) to \( U \), say, subtract \( L \) from the response and feed in \( U - L \) as the upper limit.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.
Author(s)

T. W. Yee. Help from Viet Hoang Quoc is gratefully acknowledged.

References


See Also

`negbinomial`, `Geometric`, `betageometric`, `expgeometric`, `zageometric`, `zigeometric`, `rbetageom`, `simulate.vlm`.

Examples

gdata <- data.frame(x2 = runif(nn <- 1000) - 0.5)
gdata <- transform(gdata, x3 = runif(nn) - 0.5, x4 = runif(nn) - 0.5)
gdata <- transform(gdata, eta = -1.0 - 1.0 * x2 + 2.0 * x3)
gdata <- transform(gdata, prob = logit(eta, inverse = TRUE))
gdata <- transform(gdata, y1 = rgeom(nn, prob))
with(gdata, table(y1))
fit1 <- vglm(y1 ~ x2 + x3 + x4, geometric, data = gdata, trace = TRUE)
coef(fit1, matrix = TRUE)
summary(fit1)

# Truncated geometric (between 0 and upper.limit)
upper.limit <- 5
tdata <- subset(gdata, y1 <= upper.limit)
nrow(tdata) # Less than nn
fit2 <- vglm(y1 ~ x2 + x3 + x4, truncgeometric(upper.limit),
            data = tdata, trace = TRUE)
coef(fit2, matrix = TRUE)

# Generalized truncated geometric (between lower.limit and upper.limit)
lower.limit <- 1
upper.limit <- 8
gdata <- subset(gdata, lower.limit <= y1 & y1 <= upper.limit)
with(gdata, table(y1))
nrow(gdata) # Less than nn
fit3 <- vglm(y1 ~ lower.limit - x2 + x3 + x4,
            truncgeometric(upper.limit - lower.limit),
            data = gdata, trace = TRUE)
coef(fit3, matrix = TRUE)
Description

Retrieve one component of the list .smart.prediction from smartpredenv.

Usage

get.smart()  

details

get.smart is used in "read" mode within a smart function: it retrieves parameters saved at the time of fitting, and is used for prediction. get.smart is only used in smart functions such as sm.poly; get.smart.prediction is only used in modelling functions such as lm and glm. The function get.smart gets only a part of .smart.prediction whereas get.smart.prediction gets the entire .smart.prediction.

Value

Returns with one list component of .smart.prediction from smartpredenv, in fact, .smart.prediction[[.smart.predictionCounter]].  
The whole procedure mimics a first-in first-out stack (better known as a queue).

Side Effects

The variable .smart.prediction.counter in smartpredenv is incremented beforehand, and then written back to smartpredenv.

See Also

get.smart.prediction.

Examples

print(sm.min1)

g.get.smart.prediction  
Retrieves " .smart.prediction"

description

Retrieves .smart.prediction from smartpredenv.

Usage

g.get.smart.prediction()  

details

A smart modelling function such as lm allows smart functions such as sm.bs to write to a data structure called .smart.prediction in smartpredenv. At the end of fitting, get.smart.prediction retrieves this data structure. It is then attached to the object, and used for prediction later.
Value

Returns with the list .smart.prediction from smartpredenv.

See Also

gmean, glm.

Examples

## Not run:
fit$smart <- get.smart.prediction()  # Put at the end of lm()

## End(Not run)

dev

*Generalized Extreme Value Distribution Family Function*

Description

Maximum likelihood estimation of the 3-parameter generalized extreme value (GEV) distribution.

Usage

devev(1ocation = "identitylink", lscale = "loge", lshape = logoff(offset = 0.5),
percentiles = c(95, 99), iscale=0, ishape = NULL,
imethod = 1, gshape = c(-0.45, 0.45), tolshape0 = 0.001,
type.fitted = c("percentiles", "mean"), giveWarning = TRUE, zero = 2:3)
edev(1ocation = "identitylink", lscale = "loge", lshape = logoff(offset = 0.5),
percentiles = c(95, 99), iscale=0, ishape = NULL,
imethod = 1, gshape = c(-0.45, 0.45), tolshape0 = 0.001,
type.fitted = c("percentiles", "mean"), giveWarning = TRUE, zero = 2:3)

Arguments

-location, lscale, lshape
  Parameter link functions for \(\mu\), \(\sigma\) and \(\xi\) respectively. See Links for more choices.
  For the shape parameter, the default logoff link has an offset called \(A\) below;
  and then the linear/additive predictor is \(\log(\xi + A)\) which means that \(\xi > -A\).
  For technical reasons (see Details) it is a good idea for \(A = 0.5\).

percentiles
  Numeric vector of percentiles used for the fitted values. Values should be between
  0 and 100. This argument is ignored if type.fitted = "mean".

type.fitted
  See CommonVGAMffArguments for information. The default is to use the percentiles
  argument. If "mean" is chosen, then the mean \(\mu + \sigma(1 - \xi) / \xi\) is returned
  as the fitted values, and these are only defined for \(\xi < 1\).
iscale, ishape Numeric. Initial value for $\sigma$ and $\xi$. A NULL means a value is computed internally. The argument ishape is more important than the other two because they are initialized from the initial $\xi$. If a failure to converge occurs, or even to obtain initial values occurs, try assigning ishape some value (positive or negative; the sign can be very important). Also, in general, a larger value of iscale is better than a smaller value.

imethod Initialization method. Either the value 1 or 2. Method 1 involves choosing the best $\xi$ on a course grid with endpoints gshape. Method 2 is similar to the method of moments. If both methods fail try using ishape.

gshape Numeric, of length 2. Range of $\xi$ used for a grid search for a good initial value for $\xi$. Used only if imethod equals 1.

tolshape0, giveWarning Passed into dgev when computing the log-likelihood.

zero An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \{1,2,3\} corresponding respectively to $\mu$, $\sigma$, $\xi$. If zero = NULL then all linear/additive predictors are modelled as a linear combination of the explanatory variables. For many data sets having zero = 3 is a good idea.

Details

The GEV distribution function can be written

$$G(y) = \exp(-[(y - \mu)/\sigma]_{+}^{-1/\xi})$$

where $\sigma > 0$, $-\infty < \mu < \infty$, and $1 + \xi(y - \mu)/\sigma > 0$. Here, $x_+ = \max(x,0)$. The $\mu, \sigma, \xi$ are known as the location, scale and shape parameters respectively. The cases $\xi > 0$, $\xi < 0$, $\xi = 0$ correspond to the Frechet, Weibull, and Gumbel types respectively. It can be noted that the Gumbel (or Type I) distribution accommodates many commonly-used distributions such as the normal, lognormal, logistic, gamma, exponential and Weibull.

For the GEV distribution, the $k$th moment about the mean exists if $\xi < 1/k$. Provided they exist, the mean and variance are given by $\mu + \sigma\{\Gamma(1 - \xi) - 1\}/\xi$ and $\sigma^2\{\Gamma(1 - 2\xi) - \Gamma^2(1 - \xi)\}/\xi^2$ respectively, where $\Gamma$ is the gamma function.

Smith (1985) established that when $\xi \geq -0.5$, the maximum likelihood estimators are completely regular. To have some control over the estimated $\xi$ try using lshape = logoff(offset = 0.5), say, or lshape = extlogit(min = -0.5, max = 0.5), say.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

Currently, if an estimate of $\xi$ is too close to zero then an error will occur for gev() with multivariate responses. In general, egev() is more reliable than gev().

Fitting the GEV by maximum likelihood estimation can be numerically fraught. If $1 + \xi(y - \mu)/\sigma \leq 0$ then some crude evasive action is taken but the estimation process can still fail. This is particularly
the case if `vgam` with `s` is used; then smoothing is best done with `vglm` with regression splines (`bs` or `ns`) because `vglm` implements half-stepsizing whereas `vgam` doesn’t (half-stepsizing helps handle the problem of straying outside the parameter space.)

Note

The **VGAM** family function `gev` can handle a multivariate (matrix) response. If so, each row of the matrix is sorted into descending order and NAs are put last. With a vector or one-column matrix response using egev will give the same result but be faster and it handles the \( \xi = 0 \) case. The function `gev` implements Tawn (1988) while egev implements Prescott and Walden (1980).

The shape parameter \( \xi \) is difficult to estimate accurately unless there is a lot of data. Convergence is slow when \( \xi \) is near \(-0.5\). Given many explanatory variables, it is often a good idea to make sure \( \xi = 3 \). The range restrictions of the parameter \( \xi \) are not enforced; thus it is possible for a violation to occur.

Successful convergence often depends on having a reasonably good initial value for \( \xi \). If failure occurs try various values for the argument `ishape`, and if there are covariates, having \( \text{zero} = 3 \) is advised.

**Author(s)**

T. W. Yee

**References**


See Also

rgev, gumbel, egumbel, guplot, rlplot.egev, gpd, weibullR, frechet, extlogit, oxtemp, venice.

**Examples**

```r
## Not run:
# Multivariate example
fit1 <- vgam(cbind(r1, r2) ~ s(year, df = 3), gev(zero = 2:3),
             data = venice, trace = TRUE)
coef(fit1, matrix = TRUE)
head(fitted(fit1))
par(mfrow = c(1, 2), las = 1)
plot(fit1, se = TRUE, lcol = "blue", scol = "forestgreen",
     main = "Fitted mu(year) function (centered)", cex.main = 0.8)
```
with(venice, matplot(year, depvar(fit1)[, 1:2], ylab = "Sea level (cm)",
    col = 1:2, main = "Highest 2 annual sea levels", cex.main = 0.8))
with(venice, lines(year, fitted(fit1)[,1], lty = "dashed", col = "blue"))
legend("topleft", lty = "dashed", col = "blue", "fitted 95 percentile")

# Univariate example
(fit <- vglm(maxtemp ~ 1, egev, data = oxtmp, trace = TRUE))
head(fitted(fit))
coef(fit, matrix = TRUE)
Coef(fit)
vcov(fit)
vcov(fit, untransform = TRUE)
sqrt(diag(vcov(fit))))  # Approximate standard errors
rlplot(fit)

## End(Not run)

dgev(x, location = 0, scale = 1, shape = 0, log = FALSE, tolshape0 =
    sqrt(.Machine$double.eps), oobounds.log = -Inf, giveWarning = FALSE)
pgev(q, location = 0, scale = 1, shape = 0, lower.tail = TRUE, log.p = FALSE)
qgev(p, location = 0, scale = 1, shape = 0, lower.tail = TRUE, log.p = FALSE)
rgev(n, location = 0, scale = 1, shape = 0)

Arguments

- x, q: vector of quantiles.
- p: vector of probabilities.
- n: number of observations. If length(n) > 1 then the length is taken to be the number required.
- location: the location parameter \( \mu \).
- scale: the (positive) scale parameter \( \sigma \). Must consist of positive values.
- shape: the shape parameter \( \xi \).
- log: Logical. If log = TRUE then the logarithm of the density is returned.
- lower.tail, log.p: Same meaning as in punif or qunif.
tolshape0   Positive numeric. Threshold/tolerance value for resting whether ξ is zero. If the absolute value of the estimate of ξ is less than this value then it will be assumed zero and a Gumbel distribution will be used.

oobounds, log, giveWarning

Numeric and logical. The GEV distribution has support in the region satisfying 1+shape*(x-location)/scale > 0. Outside that region, the logarithm of the density is assigned oobounds, log, which equates to a zero density. It should not be assigned a positive number, and ideally is very negative. Since egev uses this function it is necessary to return a finite value outside this region so as to allow for half-stepping. Both arguments are in support of this. This argument and others match those of egev.

Details

See gev, the VGAM family function for estimating the two parameters by maximum likelihood estimation, for formulae and other details. Apart from n, all the above arguments may be vectors and are recycled to the appropriate length if necessary.

Value
dgev gives the density, pgev gives the distribution function, qgev gives the quantile function, and rgev generates random deviates.

Note

The default value of ξ = 0 means the default distribution is the Gumbel.
Currently, these functions have different argument names compared with those in the evd package.

Author(s)

T. W. Yee

References


See Also
gev, egev, vglm.control.

Examples

```r
## Not run: loc <- 2; sigma <- 1; xi <- -0.4
x <- seq(loc - 3, loc + 3, by = 0.01)
plot(x, dgev(x, loc, sigma, xi), type = "l", col = "blue", ylim = c(0,1),
    main = "Blue is density, red is cumulative distribution function",
    sub = "Purple are 5,10,...,95 percentiles", ylab = "", las = 1)
abline(h = 0, col = "blue", lty = 2)
lines(qgev(seq(0.05, 0.95, by = 0.05), loc, sigma, xi),
```

General Electric and Westinghouse Data

Description

General Electric and Westinghouse capital data.

Usage

data(gew)

Format

A data frame with 20 observations on the following 7 variables. All variables are numeric vectors. Variables ending in .g correspond to General Electric and those ending in .w are Westinghouse.

year The observations are the years from 1934 to 1953
invest.g, invest.w investment figures. These are $I = \text{Gross investment} = \text{additions to plant and equipment plus maintenance and repairs in millions of dollars deflated by } P_1$.
capital.g, capital.w capital stocks. These are $C = \text{The stock of plant and equipment} = \text{accumulated sum of net additions to plant and equipment deflated by } P_1 \text{ minus depreciation allowance deflated by } P_3$.
value.g, value.w market values. These are $F = \text{Value of the firm} = \text{price of common and preferred shares at December 31 (or average price of December 31 and January 31 of the following year) times number of common and preferred shares outstanding plus total book value of debt at December 31 in millions of dollars deflated by } P_2$.

Details

These data are a subset of a table in Boot and de Wit (1960), also known as the Grunfeld data. It is used a lot in econometrics, e.g., for seemingly unrelated regressions (see Surff).

Here, $P_1 = \text{Implicit price deflator of producers durable equipment (base 1947)}$, $P_2 = \text{Implicit price deflator of G.N.P. (base 1947)}$, $P_3 = \text{Depreciation expense deflator} = \text{ten years moving average of wholesale price index of metals and metal products (base 1947)}$. 

```r
dgev(qgev(seq(0.05, 0.95, by = 0.05), loc, sigma, xi), loc, sigma, xi),
col = "purple", lty = 3, type = "h")
lines(x, pgev(x, loc, sigma, xi), type = "l", col = "red")
abline(h = 0, lty = 2)
pgev(qgev(seq(0.05, 0.95, by = 0.05), loc, sigma, xi), loc, sigma, xi)
```

## End(Not run)
golf

Source

References

See Also

Examples
```r
str(gew)
```

---

### Description
Computes the gamma-ordinal transformation, including its inverse and the first two derivatives.

### Usage
```r
golf(theta, lambda = 1, cutpoint = NULL, inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)
```

### Arguments
- **theta**: Numeric or character. See below for further details.
- **lambda, cutpoint**: The former is the shape parameter in gamma2. cutpoint is optional; if NULL then cutpoint is ignored from the GOLF definition. If given, the cutpoints should be non-negative integers. If golf() is used as the link function in cumulative then, if the cutpoints are known, then one should choose reverse = TRUE, parallel = FALSE ~ -1. If the cutpoints are unknown, then choose reverse = TRUE, parallel = TRUE.
- **inverse, deriv, short, tag**: Details at Links.

### Details
The gamma-ordinal link function (GOLF) can be applied to a parameter lying in the unit interval. Its purpose is to link cumulative probabilities associated with an ordinal response coming from an underlying 2-parameter gamma distribution.
See Links for general information about VGAM link functions.
Value
See Yee (2012) for details.

Warning
Prediction may not work on `vglm` or `vgam` etc. objects if this link function is used.

Note
Numerical values of theta too close to 0 or 1 or out of range result in large positive or negative values, or maybe 0 depending on the arguments. Although measures have been taken to handle cases where theta is too close to 1 or 0, numerical instabilities may still arise.

In terms of the threshold approach with cumulative probabilities for an ordinal response this link function corresponds to the gamma distribution (see `gamma2`) that has been recorded as an ordinal response using known cutpoints.

Author(s)
Thomas W. Yee

References

See Also
Links, `gamma2`, `polf`, `nbolf`, `cumulative`.

Examples
```r
golf("p", lambda = 1, short = FALSE)
golf("p", lambda = 1, tag = TRUE)

p <- seq(0.02, 0.98, len = 201)
y <- golf(p, lambda = 1)
y. <- golf(p, lambda = 1, deriv = 1)
max(abs(golf(y, lambda = 1, inv = TRUE) - p)) # Should be 0

## Not run: par(mfrow = c(2, 1), las = 1)
plot(p, y, type = "l", col = "blue", main = "golf()")
abline(h = 0, v = 0.5, col = "orange", lty = "dashed")
plot(p, y., type = "l", col = "blue",
     main = "(Reciprocal of) first GOLF derivative")

## End(Not run)

# Another example
gdata <- data.frame(x2 = sort(runif(nn <- 1000)))
gdata <- transform(gdata, x3 = runif(nn))
gdata <- transform(gdata, mymu = exp( 3 + x1 * x2 - 2 * x3))
```
The Gompertz Distribution

Description

Density, cumulative distribution function, quantile function and random generation for the Gompertz distribution.

Usage

dgompertz(x, scale = 1, shape, log = FALSE)
pgompertz(q, scale = 1, shape, lower.tail = TRUE, log.p = FALSE)
qgompertz(p, scale = 1, shape, lower.tail = TRUE, log.p = FALSE)
rgompertz(n, scale = 1, shape)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. Same as in runif.
log Logical. If log = TRUE then the logarithm of the density is returned.
lower.tail, log.p Same meaning as in pnorm or qnorm.
scale, shape positive scale and shape parameters.
Details

See `gompertz` for details.

Value

dgompertz gives the density, pgompertz gives the cumulative distribution function, qgompertz gives the quantile function, and rgompertz generates random deviates.

Author(s)

T. W. Yee and Kai Huang

See Also

`gompertz`, `dgumbel`, `dmakeham`.

Examples

```r
probs <- seq(0.01, 0.99, by = 0.01)
Shape <- exp(1); Scale <- exp(1)
max(abs(pgompertz(qgompertz(p = probs, Scale, shape = Shape),
                Scale, shape = Shape) - probs))  # Should be 0

## Not run: x <- seq(-0.1, 1.0, by = 0.001)
plot(x, dgompertz(x, Scale, shape = Shape), type = "l", col = "blue", las = 1,
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Purple lines are the 10,20,...,90 percentiles",
     ylab = "")
abline(h = 0, col = "blue", lty = 2)
lines(x, pgompertz(x, Scale, shape = Shape), col = "orange")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qgompertz(probs, Scale, shape = Shape)
lines(Q, dgompertz(Q, Scale, shape = Shape), col = "purple",
      lty = 3, type = "h")
pgompertz(Q, Scale, shape = Shape) - probs  # Should be all zero
abline(h = probs, col = "purple", lty = 3)
## End(Not run)
```

Description

Maximum likelihood estimation of the 2-parameter Gompertz distribution.

Usage

```r
gompertz(lscale = "loge", lshape = "loge",
         iscale = NULL, ishape = NULL,
         nsimEIM = 500, zero = NULL, nowarning = FALSE)
```
Arguments

nowarning Logical. Suppress a warning? Ignored for VGAM 0.9-7 and higher.

lshape, lscale Parameter link functions applied to the shape parameter a, scale parameter scale. All parameters are positive. See Links for more choices.

ishape, iscale Optional initial values. A NULL means a value is computed internally.

nsimEIM, zero See CommonVGAMffArguments.

Details

The Gompertz distribution has a cumulative distribution function

\[ F(x; \alpha, \beta) = 1 - \exp\left[-(\alpha/\beta) \times (\exp(\beta x) - 1)\right] \]

which leads to a probability density function

\[ f(x; \alpha, \beta) = \alpha \exp(\beta x) \exp\left[-(\alpha/\beta) \times (\exp(\beta x) - 1)\right] \]

for \( \alpha > 0, \beta > 0, x > 0 \). Here, \( \beta \) is called the scale parameter scale, and \( \alpha \) is called the shape parameter (one could refer to \( \alpha \) as a location parameter and \( \beta \) as a shape parameter—see Lenart (2012)). The mean is involves an exponential integral function. Simulated Fisher scoring is used and multiple responses are handled.

The Makeham distribution has an additional parameter compared to the Gompertz distribution. If \( X \) is defined to be the result of sampling from a Gumbel distribution until a negative value \( Z \) is produced, then \( X = -Z \) has a Gompertz distribution.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

The same warnings in makeham apply here too.

Author(s)

T. W. Yee

References


See Also
dgompertz, makeham, simulate.vlm.
Examples

```r
## Not run:
gdata <- data.frame(x2 = runif(nn <- 1000))
gdata <- transform(gdata, eta1 = -1,
                   eta2 = -1 + 0.2 * x2,
                   ceta1 = 1,
                   ceta2 = -1 + 0.2 * x2)
gdata <- transform(gdata, shape1 = exp(eta1),
                   shape2 = exp(eta2),
                   scale1 = exp(ceta1),
                   scale2 = exp(ceta2))
gdata <- transform(gdata, y1 = rgompertz(nn, scale = scale1, shape = shape1),
                   y2 = rgompertz(nn, scale = scale2, shape = shape2))

fit1 <- vglm(y1 ~ 1, gompertz, data = gdata, trace = TRUE)
fit2 <- vglm(y2 ~ x2, gompertz, data = gdata, trace = TRUE)
coef(fit1, matrix = TRUE)
Coef(fit1)
summary(fit1)
coef(fit2, matrix = TRUE)
summary(fit2)
## End(Not run)
```

---

**gpd**

*Generalized Pareto Distribution Family Function*

**Description**

Maximum likelihood estimation of the 2-parameter generalized Pareto distribution (GPD).

**Usage**

```r
gpd(threshold = 0, lscale = "loge", lshape = logoff(offset = 0.5),
    percentiles = c(90, 95), iscale = NULL, ishape = NULL,
    tolshape0 = 0.001, type.fitted = c("percentiles", "mean"),
    giveWarning = TRUE, imethod = 1, zero = -2)
```

**Arguments**

- `threshold` Numeric, values are recycled if necessary. The threshold value(s), called $\mu$ below.
- `lscale` Parameter link function for the scale parameter $\sigma$. See [Links](#) for more choices.
- `lshape` Parameter link function for the shape parameter $\xi$. See [Links](#) for more choices. The default constrains the parameter to be greater than $-0.5$ because if $\xi \leq -0.5$ then Fisher scoring does not work. See the Details section below for more information.
For the shape parameter, the default logoff link has an offset called \( A \) below; and then the second linear/additive predictor is \( \log(\xi + A) \) which means that \( \xi > -A \). The working weight matrices are positive definite if \( A = 0.5 \).

**percentiles** Numeric vector of percentiles used for the fitted values. Values should be between 0 and 100. See the example below for illustration. This argument is ignored if `type.fitted = "mean"`.

**type.fitted** See CommonVGAMffArguments for information. The default is to use the percentiles argument. If "mean" is chosen, then the mean \( \mu + \sigma/(1 - \xi) \) is returned as the fitted values, and these are only defined for \( \xi < 1 \).

**iscale, ishape** Numeric. Optional initial values for \( \sigma \) and \( \xi \). The default is to use `imethod` and compute a value internally for each parameter. Values of `ishape` should be between \(-0.5, 1\). Values of `iscale` should be positive.

**tolshape0, giveWarning** Passed into `dgpd` when computing the log-likelihood.

**imethod** Method of initialization, either 1 or 2. The first is the method of moments, and the second is a variant of this. If neither work, try assigning values to arguments `ishape` and/or `iscale`.

**zero** An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. For one response, the value should be from the set \{1,2\} corresponding respectively to \( \sigma \) and \( \xi \). It is often a good idea for the \( \sigma \) parameter only to be modelled through a linear combination of the explanatory variables because the shape parameter is probably best left as an intercept only: `zero = 2`. Setting `zero = NULL` means both parameters are modelled with explanatory variables. See CommonVGAMffArguments for more details.

**Details**

The distribution function of the GPD can be written

\[
G(y) = 1 - [1 + \xi(y - \mu)/\sigma]_{+}^{-1/\xi}
\]

where \( \mu \) is the location parameter (known, with value `threshold`), \( \sigma > 0 \) is the scale parameter, \( \xi \) is the shape parameter, and \( h_{+} = \max(h, 0) \). The function \( 1 - G \) is known as the *survivor function*. The limit \( \xi \to 0 \) gives the *shifted exponential* as a special case:

\[
G(y) = 1 - \exp[-(y - \mu)/\sigma].
\]

The support is \( y > \mu \) for \( \xi > 0 \), and \( \mu < y < \mu - \sigma/\xi \) for \( \xi < 0 \).

Smith (1985) showed that if \( \xi <= -0.5 \) then this is known as the nonregular case and problems/difficulties can arise both theoretically and numerically. For the (regular) case \( \xi > -0.5 \) the classical asymptotic theory of maximum likelihood estimators is applicable; this is the default. Although for \( \xi < -0.5 \) the usual asymptotic properties do not apply, the maximum likelihood estimator generally exists and is superefficient for \( -1 < \xi < -0.5 \), so it is “better” than normal. When \( \xi < -1 \) the maximum likelihood estimator generally does not exist as it effectively becomes a two parameter problem.

The mean of \( Y \) does not exist unless \( \xi < 1 \), and the variance does not exist unless \( \xi < 0.5 \). So if you want to fit a model with finite variance use `lshape = "extlogit"`. 


Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam. However, for this VGAM family function, vglm is probably preferred over vgam when there is smoothing.

Warning

Fitting the GPD by maximum likelihood estimation can be numerically fraught. If $1 + \xi (y - \mu) / \sigma \leq 0$ then some crude evasive action is taken but the estimation process can still fail. This is particularly the case if vgam with s is used. Then smoothing is best done with vglm with regression splines (bs or ns) because vglm implements half-stepsizing whereas vgam doesn’t. Half-stepsizing helps handle the problem of straying outside the parameter space.

Note

The response in the formula of vglm and vgam is $y$. Internally, $y - \mu$ is computed. This VGAM family function can handle a multiple responses, which is inputted as a matrix. The response stored on the object is the original uncentred data.

With functions rgpd, dgpd, etc., the argument location matches with the argument threshold here.

Author(s)

T. W. Yee

References


See Also

rgpd, meplot, gev, paretoff, vglm, vgam, s.

Examples

# Simulated data from an exponential distribution (xi = 0)
Threshold <- 0.5
gdata <- data.frame(yl = Threshold + rexp(n = 3000, rate = 2))
fit <- vglm(y1 ~ 1, gpd(threshold = Threshold), data = gdata, trace = TRUE)
head(fitted(fit))
summary(depvar(fit)) # The original uncentred data
coef(fit, matrix = TRUE) # xi should be close to 0
Coef(fit)
summary(fit)
head(fit@extra$threshold) # Note the threshold is stored here

# Check the 90 percentile
ii <- depvar(fit) < fitted(fit)[1, "90%"]
100 * table(ii) / sum(table(ii)) # Should be 90%

# Check the 95 percentile
ii <- depvar(fit) < fitted(fit)[1, "95%"]
100 * table(ii) / sum(table(ii)) # Should be 95%

## Not run: plot(depvar(fit), col = "blue", las = 1,
# main = "Fitted 90% and 95% quantiles")
matlines(1:length(depvar(fit)), fitted(fit), lty = 2:3, lwd = 2)
## End(Not run)

Another example

gdata <- data.frame(x2 = runif(nn <- 2000))
Threshold <- 0; xi <- exp(-0.8) - 0.5
gdata <- transform(gdata, y2 = rgpd(nn, scale = exp(1 + 0.1*x2), shape = xi))
fit <- vglm(y2 ~ x2, gpd(Threshold), data = gdata, trace = TRUE)
coef(fit, matrix = TRUE)

## Not run: # Nonparametric fits
# Not so recommended:
fit1 <- vgam(y2 ~ s(x2), gpd(Threshold), data = gdata, trace = TRUE)
par(mfrow = c(2, 1))
plot(fit1, se = TRUE, scl = "blue")
# More recommended:
fit2 <- vglm(y2 ~ sm.bs(x2), gpd(Threshold), data = gdata, trace = TRUE)
plot(as(fit2, "vgam"), se = TRUE, scl = "blue")
## End(Not run)

---

**The Generalized Pareto Distribution**

**Description**

Density, distribution function, quantile function and random generation for the generalized Pareto distribution (GPD) with location parameter location, scale parameter scale and shape parameter shape.

**Usage**

dgpd(x, location = 0, scale = 1, shape = 0, log = FALSE,
tolshape = sqrt(.Machine$double.eps),
oobounds.log = -Inf, giveWarning = FALSE)
p pgpd(q, location = 0, scale = 1, shape = 0,
Arguments

- **x, q** vector of quantiles.
- **p** vector of probabilities.
- **n** number of observations. If `length(n) > 1` then the length is taken to be the number required.
- **location** the location parameter \( \mu \).
- **scale** the (positive) scale parameter \( \sigma \).
- **shape** the shape parameter \( \xi \).
- **log** Logical. If `log = TRUE` then the logarithm of the density is returned.
- **lower.tail, log.p** Same meaning as in `punif` or `qunif`.
- **tolshape0** Positive numeric. Threshold/tolerance value for resting whether \( \xi \) is zero. If the absolute value of the estimate of \( \xi \) is less than this value then it will be assumed zero and an exponential distribution will be used.
- **oobounds.log, giveWarning** Numeric and logical. The GPD distribution has support in the region satisfying \((x - location)/scale > 0\) and \(1 + shape*(x - location)/scale > 0\). Outside that region, the logarithm of the density is assigned `oobounds.log`, which equates to a zero density. It should not be assigned a positive number, and ideally is very negative. Since `gpd` uses this function it is necessary to return a finite value outside this region so as to allow for half-stepping. Both arguments are in support of this. This argument and others match those of `gpd`.

Details

See `gpd`, the `VGAM` family function for estimating the two parameters by maximum likelihood estimation, for formulae and other details. Apart from `n`, all the above arguments may be vectors and are recycled to the appropriate length if necessary.

Value

dgpd gives the density, pgpd gives the distribution function, qgpd gives the quantile function, and rgpd generates random deviates.

Note

The default values of all three parameters, especially \( \xi = 0 \), means the default distribution is the exponential.

Currently, these functions have different argument names compared with those in the `evd` package.
Description

A 4-column matrix.

Usage

data(grain.us)

Format

The columns are:

- `wheat.flour` numeric
- `corn` numeric
- `wheat` numeric
- `rye` numeric
Details

Monthly averages of grain prices in the United States for wheat flour, corn, wheat, and rye for the period January 1961 through October 1972. The units are US dollars per 100 pound sack for wheat flour, and per bushel for corn, wheat and rye.

Source


References


Examples

```r
### Not run:
cgrain <- scale(grain.us, scale = FALSE)  # Center the time series only
fit <- vglm(cgrain ~ 1, rrar(Rank = c(4, 1)),
            epsilon = 1e-3, stepsize = 0.5, trace = TRUE, maxit = 50)
summary(fit)
### End(Not run)
```

grc

Row-Column Interaction Models including Goodman’s RC Association Model and Unconstrained Quadratic Ordination

Description

Fits a Goodman’s RC association model (GRC) to a matrix of counts, and more generally, row-column interaction models (RCIMs). RCIMs allow for unconstrained quadratic ordination (UQO).

Usage

```r
grc(y, Rank = 1, Index.corner = 2:(1 + Rank),
   str0 = 1, summary.arg = FALSE, h.step = 1e-04, ...)
rcim(y, family = poissonff, Rank = 0, M1 = NULL,
     weights = NULL, which.linpred = 1,
     Index.corner = ifelse(is.null(str0), 0, max(str0)) + 1:Rank,
     rprefix = "Row.", cprefix = "Col.", iprefix = "X2."
     offset = 0, str0 = if (Rank) 1 else NULL,
     summary.arg = FALSE, h.step = 0.0001,
     rbaseline = 1, cbaseline = 1,
     has.intercept = TRUE, M = NULL,
     rindex = 2:nrow(y), cindex = 2:ncol(y), iindex = 2:nrow(y), ...)
```
Arguments

y  
For grc(): a matrix of counts. For rcim(): a general matrix response depending on family. Output from table() is acceptable; it is converted into a matrix. Note that y should be at least 3 by 3 in dimension.

family  
A VGAM family function. By default, the first linear/additive predictor is fitted using main effects plus an optional rank-Rank interaction term. Not all family functions are suitable or make sense. All other linear/additive predictors are fitted using an intercept-only, so it has a common value over all rows and columns. For example, zipoissonff may be suitable for counts but not zipoisson because of the ordering of the linear/additive predictors. If the VGAM family function does not have an infos slot then M needs to be inputted (the number of linear predictors for an ordinary (usually univariate) response, aka M). The VGAM family function also needs to be able to handle multiple responses (currently not all of them can do this).

Rank  
An integer from the set \{0, \ldots, \min(nrow(y), ncol(y))\}. This is the dimension of the fit in terms of the interaction. For grc() this argument must be positive. A value of 0 means no interactions (i.e., main effects only); each row and column is represented by an indicator variable.

weights  
Prior weights. Fed into rrvglm or vglm.

which.linpred  
Single integer. Specifies which linear predictor is modelled as the sum of an intercept, row effect, column effect plus an optional interaction term. It should be one value from the set 1:M.

Index.corner  
A vector of Rank integers. These are used to store the Rank by Rank identity matrix in the A matrix; corner constraints are used.

rprefix, cprefix, iprefix  
Character, for rows and columns and interactions respectively. For labelling the indicator variables.

offset  
Numeric. Either a matrix of the right dimension, else a single numeric expanded into such a matrix.

str0  
Ignored if \text{Rank} < 0, else an integer from the set \{1, \ldots, \min(nrow(y), ncol(y))\}, specifying the row that is used as the structural zero. Passed into rrvglm.control if \text{Rank} < 0. Set str0 = NULL for none.

summary.arg  
Logical. If TRUE then a summary is returned. If TRUE then y may be the output (fitted object) of grc().

h.step  
A small positive value that is passed into summary.rrvglm(). Only used when summary.arg = TRUE.

...  
Arguments that are passed into rrvglm.control().

M1  
The number of linear predictors of the VGAM family function for an ordinary (univariate) response. Then the number of linear predictors of the rcim() fit is usually the number of columns of y multiplied by M1. The default is to evaluate the infos slot of the VGAM family function to try to evaluate it; see vglmff-class. If this information is not yet supplied by the family function then the value needs to be inputted manually using this argument.
Baseliner reference levels for the rows and columns. Currently stored on the object but not used.

Logical. Include an intercept?

is the usual VGAM $M$, viz. the number of linear/additive predictors in total. Also, $c$ means column index, and these point to the columns of $y$ which are part of the vector of linear/additive predictor main effects.

For family = multinomial it is necessary to input these arguments as $M = ncol(y) - 1$ and $c$ = 2:$(ncol(y)-1)$.

$r$ means row index, and these are similar to $c$. $i$ means interaction index, and these are similar to $c$.

Details

Goodman’s RC association model fits a reduced-rank approximation to a table of counts. A Poisson model is assumed. The log of each cell mean is decomposed as an intercept plus a row effect plus a column effect plus a reduced-rank component. The latter can be collectively written $A \times C$, the product of two ‘thin’ matrices. Indeed, $A$ and $C$ have rank columns. By default, the first column and row of the interaction matrix $A \times C$ is chosen to be structural zeros, because $str0 = 1$. This means the first row of $A$ are all zeros.

This function uses options to set up the row and column indicator variables. In particular, Equation (4.5) of Yee and Hastie (2003) is used. These are called Row. and Col. (by default) followed by the row or column number.

The function rcim() is more general than grc(). Its default is a no-interaction model of grc(), i.e., rank-0 and a Poisson distribution. This means that each row and column has a dummy variable associated with it. The first row and first column are baseline. The power of rcim() is that many VGAM family functions can be assigned to its family argument. For example, uninormal fits something in between a 2-way ANOVA with and without interactions, alaplaceR with rank = 0 is something like medpolish. Others include zipoissonff and negbinomial. Hopefully one day all VGAM family functions will work when assigned to the family argument, although the result may not have meaning.

Unconstrained quadratic ordination (UQO) can be performed using rcim() and grc(). This has been called unconstrained Gaussian ordination in the literature, however the word Gaussian has two meanings which is confusing: it is better to use quadratic because the bell-shape response surface is meant. UQO is similar to CQO (cqo) except there are no environmental/explanatory variables. Here, a GLM is fitted to each column (species) that is a quadratic function of hypothetical latent variables or gradients. Thus each row of the response has an associated site score, and each column of the response has an associated optimum and tolerance matrix. UQO can be performed here under the assumption that all species have the same tolerance matrices. See Yee and Hadi (2014) for details. It is not recommended that presence/absence data be inputted because the information content is so low for each site-species cell. The example below uses Poisson counts.

Value

An object of class "grc", which currently is the same as an "rrvglm" object. Currently, a rank-0 rcim() object is of class rcim@-class, else of class "rcim" (this may change in the future).
Warning

The function rrcim() is experimental at this stage and may have bugs. Quite a lot of expertise is needed when fitting and in its interpretation thereof. For example, the constraint matrices applies the reduced-rank regression to the first (see which.linpred) linear predictor and the other linear predictors are intercept-only and have a common value throughout the entire data set. This means that, by default, family = zipoissonff is appropriate but not family = zipoisson. Else set family = zipoisson and which.linpred = 2. To understand what is going on, do examine the constraint matrices of the fitted object, and reconcile this with Equations (4.3) to (4.5) of Yee and Hastie (2003).

The functions temporarily create a permanent data frame called .grc.df or .rcim.df, which used to be needed by summary.rrvglm(). Then these data frames are deleted before exiting the function. If an error occurs then the data frames may be present in the workspace.

Note

These functions set up the indicator variables etc. before calling rrvglm or vglm. The . . . is passed into rrvglm.control or vglm.control. This means, e.g., Rank = 1 is default for grc().

The data should be labelled with rownames and colnames. Setting trace = TRUE is recommended to monitor convergence. Using criterion = "coefficients" can result in slow convergence.

If summary = TRUE then y can be a "grc" object, in which case a summary can be returned. That is, grc(y, summary = TRUE) is equivalent to summary(grc(y)). It is not possible to plot a grc(y, summary = TRUE) or rcim(y, summary = TRUE) object.

Author(s)

Thomas W. Yee, with assistance from Alfian F. Hadi.

References


See Also

rrvglm, rrvglm.control, rrvglm-class, summary.grc, moffset, Rrcim, Select, Qvar, plotrcim, cqq, multinomial, alcoff, crashi, auuc, olym08, olym12, poissonff, medpolish.

Examples

# Example 1: Undergraduate enrolments at Auckland University in 1990
fitted(grc1 <- grc(auuc))
summary(grc1)
grc2 <- grc(auuc, Rank = 2, Index.corner = c(2, 5))
fitted(grc2)
summary(grc2)

model3 <- rcim(auuc, Rank = 1, fam = multinomial, 
            M = ncol(auuc)-1, cindex = 2:(ncol(auuc)-1), trace = TRUE)
fitted(model3)
summary(model3)

# Median polish but not 100 percent reliable. Maybe call alaplace2(...
# Not run:
rcim0 <- rcim(auuc, fam = alaplace(tau = 0.5), trace = FALSE, maxit = 500)
round(fitted(rcim0), digits = 0)
round(100 * (fitted(rcim0) - auuc) / auuc, digits = 0) # Discrepancy
depvar(rcim0)
round(coef(rcim0, matrix = TRUE), digits = 2)
Coef(rcim0, matrix = TRUE)
# constraints(rcim0)
names(constraints(rcim0))

# Compare with medpolish():
(med.a <- medpolish(auuc))
fv <- med.a$overall + outer(med.a$row, med.a$col, "+")
round(100 * (fitted(rcim0) - fv) / fv) # Hopefully should be all 0s
## (Not run)

# Example 2: 2012 Summer Olympic Games in London
## Not run: top10 <- head(olymp12, 10)
grc1.oly12 <- with(top10, grc(cbind(gold, silver, bronze)))
round(fitted(grc1.oly12))
round(resid(grc1.oly12, type = "response"), digits = 1) # Response residuals
summary(grc1.oly12)
Coef(grc1.oly12)
## (Not run)

# Example 3: Unconstrained quadratic ordination (UQO); see Yee and Hadi (2014)
## Not run:
n <- 100; p <- 5; S <- 10
pdata <- rcqo(n, p, S, es.opt = FALSE, eq.max = FALSE,
             eq.tol = TRUE, sd.latvar = 0.75) # Poisson counts
true.nu <- attr(pdata, "latvar") # The 'truth'; site scores
attr(pdata, "tolerances") # The 'truth'; tolerances
Y <- Select(pdata, "y", sort = FALSE) # Y matrix (n x S); the "y" vars
uqo.rcim1 <- rcim(Y, Rank = 1,
            str0 = NULL, # Delta covers entire n x M matrix
            iindex = 1:nrow(Y), # RRR covers the entire Y
            has.intercept = FALSE) # Suppress the intercept
# Plot 1
par(mfrow = c(2, 2))
plot(attr(pdata, "optimums"), Coef(uqo.rciml)@A,
   col = "blue", type = "p", main = "(a) UQO optimums",
   xlab = "True optimums", ylab = "Estimated (UQO) optimums")
mylm <- lm(Coef(uqo.rciml)@A ~ attr(pdata, "optimums"))
abline(coef = coef(mylm), col = "orange", lty = "dashed")

# Plot 2
fill.val <- NULL  # Choose this for the new parameterization
plot(attr(pdata, "latvar"), c(fill.val, concoef(uqo.rciml)),
   las = 1, col = "blue", type = "p", main = "(b) UQO site scores",
   xlab = "True site scores", ylab = "Estimated (UQO) site scores")
mylm <- lm(c(fill.val, concoef(uqo.rciml)) ~ attr(pdata, "latvar"))
abline(coef = coef(mylm), col = "orange", lty = "dashed")

# Plots 3 and 4
myform <- attr(pdata, "formula")
plut <- cpo(myform, family = poissonff,
   eq.tol = FALSE, trace = FALSE, data = pdata)
clut <- cpo(Select(pdata, "y", sort = FALSE) ~ scale(latvar(uqo.rciml)),
   family = poissonff, eq.tol = FALSE, trace = FALSE, data = pdata)
lvplot(plut, lcol = 1:5, y = TRUE, pcol = 1:5, pch = 1:5, pcex = 0.5,
   main = "(c) CQO fitted to the original data",
   xlab = "Estimated (CQO) site scores")
lvplot(clut, lcol = 1:5, y = TRUE, pcol = 1:5, pch = 1:5, pcex = 0.5,
   main = "(d) CQO fitted to the scaled UQO site scores",
   xlab = "Estimated (UQO) site scores")

## End(Not run)

gumbel  

## Gumbel Distribution Family Function

description

Maximum likelihood estimation of the 2-parameter Gumbel distribution.

Usage

gumbel(llocation = "identitylink", lscale = "loge",
   iscale = NULL, R = NA, percentiles = c(95, 99),
   mpv = FALSE, zero = NULL)

egumbel(llocation = "identitylink", lscale = "loge",
   iscale = NULL, R = NA, percentiles = c(95, 99),
   mpv = FALSE, zero = NULL)
Arguments

llocation, lscale
Parameter link functions for $\mu$ and $\sigma$. See Links for more choices.

iscale
Numeric and positive. Optional initial value for $\sigma$. Recycled to the appropriate length. In general, a larger value is better than a smaller value. A NULL means an initial value is computed internally.

R
Numeric. Maximum number of values possible. See Details for more details.

percentiles
Numeric vector of percentiles used for the fitted values. Values should be between 0 and 100. This argument uses the argument $R$ if assigned. If percentiles = NULL then the mean will be returned as the fitted values.

mpv
Logical. If mpv = TRUE then the median predicted value (MPV) is computed and returned as the (last) column of the fitted values. This argument is ignored if percentiles = NULL. See Details for more details.

zero
An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The value (possibly values) must be from the set {1, 2} corresponding respectively to $\mu$ and $\sigma$. By default all linear/additive predictors are modelled as a linear combination of the explanatory variables. See CommonVGAMffArguments for more details.

Details

The Gumbel distribution is a generalized extreme value (GEV) distribution with shape parameter $\xi = 0$. Consequently it is more easily estimated than the GEV. See gev for more details.

The quantity $R$ is the maximum number of observations possible, for example, in the Venice data below, the top 10 daily values are recorded for each year, therefore $R = 365$ because there are about 365 days per year. The MPV is the value of the response such that the probability of obtaining a value greater than the MPV is 0.5 out of $R$ observations. For the Venice data, the MPV is the sea level such that there is an even chance that the highest level for a particular year exceeds the MPV. If mpv = TRUE then the column labelled "MPV" contains the MPVs when fitted() is applied to the fitted object.

The formula for the mean of a response $Y$ is $\mu + \sigma \times Euler$ where $Euler$ is a constant that has value approximately equal to 0.5772. The formula for the percentiles are (if $R$ is not given) $\mu - \sigma \times \log(-\log(P/100))$ where $P$ is the percentile argument value(s). If $R$ is given then the percentiles are $\mu - \sigma \times \log[R(1 - P/100)].$

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

When $R$ is not given (the default) the fitted percentiles are that of the data, and not of the overall population. For example, in the example below, the 50 percentile is approximately the running median through the data, however, the data are the highest sea level measurements recorded each year (it therefore equates to the median predicted value or MPV).
**Note**

egumbel() only handles a univariate response, and is preferred to gumbel() because it is faster.
gumbel() can handle a multivariate response, i.e., a matrix with more than one column. Each row of the matrix is sorted into descending order. Missing values in the response are allowed but require na.action = na.pass. The response matrix needs to be padded with any missing values. With a multivariate response one has a matrix y, say, where y[, 2] contains the second order statistics, etc.

**Author(s)**

T. W. Yee

**References**


**See Also**

rgumbel, dgumbel2, cens.gumbel, guplot, gev, egev, venice.

**Examples**

```r
# Example 1: Simulated data
gdata <- data.frame(y1 = rgumbel(n = 1000, loc = 100, scale = exp(1)))
fit1 <- vglm(y1 ~ 1, egumbel(perc = NULL), data = gdata, trace = TRUE)
coef(fit1, matrix = TRUE)
Coef(fit1)
head(fitted(fit1))
with(gdata, mean(y1))

# Example 2: Venice data
(fit2 <- vglm(cbind(r1, r2, r3, r4, r5) ~ year, data = venice,
gumbel(R = 365, mpv = TRUE), trace = TRUE))
head(fitted(fit2))
coef(fit2, matrix = TRUE)
sqrt(diag(vcov(summary(fit2)))))  # Standard errors

# Example 3: Try a nonparametric fit
# Use the entire data set, including missing values
# Same as as.matrix(venice[, paste0("r", 1:10)):
```
Y <- Select(venice, "r", sort = FALSE)
fit3 <- vgam(Y ~ s(year, df = 3), gumbel(R = 365, mpv = TRUE),
             data = venice, trace = TRUE, na.action = na.pass)
depvar(fit3)[4:5, ] # NAs used to pad the matrix

## Not run: # Plot the component functions
par(mfrow = c(2, 3), mar = c(6, 4, 1, 2) + 0.3, xpd = TRUE)
plot(fit3, se = TRUE, lcol = "blue", scl = "limegreen", lty = 1,
     lwd = 2, slwd = 2, sly = "dashed")

# Quantile plot --- plots all the fitted values
qtplot(fit3, mpv = TRUE, lcol = c(1, 2, 5), tcol = c(1, 2, 5), lwd = 2,
       pcol = "blue", tadj = 0.1, ylab = "Sea level (cm)"
)

# Plot the 99 percentile only
year <- - venice["year"]
matplot(year, Y, ylab = "Sea level (cm)", type = "n")
matpoints(year, Y, pch = "x", col = "blue")
lines(year, fitted(fit3)[, "99%"], lwd = 2, col = "orange")

# Check the 99 percentiles with a smoothing spline.
# Nb. (1-0.99) * 365 = 3.65 is approx. 4, meaning the 4th order
# statistic is approximately the 99 percentile.
plot(year, Y[, 4], ylab = "Sea level (cm)", type = "n",
     main = "Orange is 99 percentile, Green is a smoothing spline")
points(year, Y[, 4], pch = "4", col = "blue")
lines(year, fitted(fit3)[, "99%"], lty = 1, col = "orange")
lines(smooth.spline(year, Y[, 4], df = 4), col = "limegreen", lty = 2)

## End(Not run)

---

**Gumbel-II**

*The Gumbel-II Distribution*

**Description**

Density, cumulative distribution function, quantile function and random generation for the Gumbel-II distribution.

**Usage**

```r
dgumbelII(x, scale = 1, shape, log = FALSE)
pgumbelII(q, scale = 1, shape, lower.tail = TRUE, log.p = FALSE)
qgumbelII(p, scale = 1, shape, lower.tail = TRUE, log.p = FALSE)
rngumbelII(n, scale = 1, shape)
```

**Arguments**

- `x, q` vector of quantiles.
p vector of probabilities.
n number of observations. Same as in runif.
log Logical. If log = TRUE then the logarithm of the density is returned.
lower.tail, log.p Same meaning as in pnorm or qnorm.
shape, scale positive shape and scale parameters.

Details
See gumbelII for details.

Value
dgumbelII gives the density, pgumbelII gives the cumulative distribution function, qgumbelII gives the quantile function, and rgumbelII generates random deviates.

Author(s)
T. W. Yee and Kai Huang

See Also
gumbelII, dgumbe.

Examples
probs <- seq(0.01, 0.99, by = 0.01)
Scale <- exp(1); Shape <- exp(0.5);
max(abs(pgumbelII(gumbelII(p = probs, shape = Shape, Scale),
shape = Shape, Scale) - probs)) # Should be 0

## Not run: x <- seq(-0.1, 10, by = 0.01);
plot(x, dgumbelII(x, shape = Shape, Scale), type = "l", col = "blue", las = 1,
main = "Blue is density, orange is cumulative distribution function",
sub = "Purple lines are the 10,20,...,90 percentiles",
ylab = "", ylim = 0:1)
abline(h = 0, col = "blue", lty = 2)
lines(x, pgumbelII(x, shape = Shape, Scale), col = "orange")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- gumbelII(probs, shape = Shape, Scale)
lines(Q, pgumbelII(Q, Scale, Shape), col = "purple", lty = 3, type = "h")
pgumbelII(Q, shape = Shape, Scale) - probs # Should be all zero
abline(h = probs, col = "purple", lty = 3)
## End(Not run)
**gumbelII**

*Gumbel-II Distribution Family Function*

**Description**

Maximum likelihood estimation of the 2-parameter Gumbel-II distribution.

**Usage**

```r
gumbelII(lscale = "loge", lshape = "loge", iscale = NULL, ishape = NULL, 
    probs.y = c(0.2, 0.5, 0.8), perc.out = NULL, imethod = 1, 
    zero = -1, nowarning = FALSE)
```

**Arguments**

- `nowarning`: Logical. Suppress a warning?
- `lshape`, `lscale`: Parameter link functions applied to the (positive) shape parameter (called `s` below) and (positive) scale parameter (called `b` below). See Links for more choices.
- `ishape`, `iscale`: Optional initial values for the shape and scale parameters.
- `imethod`: See `weibullR`.
- `zero`, `probs.y`: Details at CommonVGAMffArguments.
- `perc.out`: If the fitted values are to be quantiles then set this argument to be the percentiles of these, e.g., 50 for median.

**Details**

The Gumbel-II density for a response `Y` is

\[ f(y; b, s) = sy^{s-1} \exp[-(y/b)^s]/(b^s) \]

for \( b > 0, s > 0, y > 0 \). The cumulative distribution function is

\[ F(y; b, s) = \exp[-(y/b)^{-s}]. \]

The mean of `Y` is \( b \Gamma(1 - 1/s) \) (returned as the fitted values) when \( s > 1 \), and the variance is \( b^2 \Gamma(1 - 2/s) \) when \( s > 2 \). This distribution looks similar to `weibullR`, and is due to Gumbel (1954).

This VGAM family function currently does not handle censored data. Fisher scoring is used to estimate the two parameters. Probably similar regularity conditions hold for this distribution compared to the Weibull distribution.

**Value**

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.
Note

See weibullR. This VGAM family function handles multiple responses.

Author(s)

T. W. Yee

References

Gumbel, E. J. (1954). Statistical theory of extreme values and some practical applications. Applied
Mathematics Series, volume 33, U.S. Department of Commerce, National Bureau of Standards,
USA.

See Also
dgumbelII, gumbel.gev.

Examples

```r
gdata <- data.frame(x2 = runif(nn <- 1000))
gdata <- transform(gdata, heta1 = +1,
                   heta2 = -1 + 0.1 * x2,
                   ceta1 = 0,
                   ceta2 = 1)
gdata <- transform(gdata, shape1 = exp(heta1),
                   shape2 = exp(heta2),
                   scale1 = exp(ceta1),
                   scale2 = exp(ceta2))
gdata <- transform(gdata,
                   y1 = rgumbelII(nn, scale = scale1, shape = shape1),
                   y2 = rgumbelII(nn, scale = scale2, shape = shape2))
fit <- vglm(cbind(y1, y2) ~ x2,
            gumbelII(zero = c(1, 2, 3)), data = gdata, trace = TRUE)
coef(fit, matrix = TRUE)
vcov(fit)
summary(fit)
```

---

gumbelUC  The Gumbel Distribution

Description

Density, distribution function, quantile function and random generation for the Gumbel distribution
with location parameter `location` and scale parameter `scale`.
Usage

dgumbel(x, location = 0, scale = 1, log = FALSE)
pgumbel(q, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
qgumbel(p, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
rgumbel(n, location = 0, scale = 1)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. If length(n) > 1 then the length is taken to be the number required.
location the location parameter \( \mu \). This is not the mean of the Gumbel distribution (see Details below).
scale the scale parameter \( \sigma \). This is not the standard deviation of the Gumbel distribution (see Details below).
log Logical. If log = TRUE then the logarithm of the density is returned.
lower.tail, log.p
Same meaning as in punif or qunif.

Details

The Gumbel distribution is a special case of the generalized extreme value (GEV) distribution where the shape parameter \( \xi = 0 \). The latter has 3 parameters, so the Gumbel distribution has two. The Gumbel distribution function is

\[
G(y) = \exp \left( - \exp \left( -\frac{y - \mu}{\sigma} \right) \right)
\]

where \(-\infty < y < \infty\), \(-\infty < \mu < \infty\) and \(\sigma > 0\). Its mean is

\[\mu - \sigma \cdot \gamma\]

and its variance is

\[\sigma^2 \cdot \pi^2 / 6\]

where \(\gamma\) is Euler’s constant (which can be obtained as \(-\text{digamma}(1)\)).

See gumbel, the VGAM family function for estimating the two parameters by maximum likelihood estimation, for formulae and other details. Apart from \(n\), all the above arguments may be vectors and are recycled to the appropriate length if necessary.

Value

dgumbel gives the density, pgumbel gives the distribution function, qgumbel gives the quantile function, and rgumbel generates random deviates.

Note

The VGAM family function gumbel can estimate the parameters of a Gumbel distribution using maximum likelihood estimation.
guplot

**Author(s)**

T. W. Yee

**References**


**See Also**

`gumbel`, `egumbel`, `gev`, `dgompertz`.

**Examples**

```r
mu <- 1; sigma <- 2;
y <- rgumbel(n = 100, loc = mu, scale = sigma)
c(mean(y), mu - sigma * digamma(1))  # Sample and population means
c(var(y), sigma^2 * pi^2 / 6)  # Sample and population variances

## Not run: x <- seq(-2.5, 3.5, by = 0.01)
loc <- 0; sigma <- 1
plot(x, dgumbel(x, loc, sigma), type = "l", col = "blue", ylim = c(0, 1),
     main = "Blue is density, red is cumulative distribution function",
     sub = "Purple are 5,10,...,95 percentiles", ylab = "", las = 1)
abline(h = 0, col = "blue", lty = 2)
lines(qgumbel(seq(0.05, 0.95, by = 0.05), loc, sigma),
      dgumbel(qgumbel(seq(0.05, 0.95, by = 0.05), loc, sigma), loc, sigma),
      col = "purple", lty = 3, type = "h")
lines(x, pgumbel(x, loc, sigma), type = "l", col = "red")
abline(h = 0, lty = 2)
## End(Not run)
```

---

**guplot**

**Gumbel Plot**

**Description**

Produces a Gumbel plot, a diagnostic plot for checking whether the data appears to be from a Gumbel distribution.

**Usage**

```r
guplot(object, ...)
guplot.default(y, main = "Gumbel Plot",
    xlab = "Reduced data", ylab = "Observed data", type = "p", ...)
guplot.vlm(object, ...)
```
Arguments

- **y**: A numerical vector. NAs etc. are not allowed.
- **main**: Character. Overall title for the plot.
- **xlab**: Character. Title for the x axis.
- **ylab**: Character. Title for the y axis.
- **type**: Type of plot. The default means points are plotted.
- **object**: An object that inherits class "vlm", usually of class `vglm-class` or `vgam-class`.
- **...**: Graphical argument passed into `plot`. See `par` for an exhaustive list. The arguments `xlim` and `ylim` are particularly useful.

Details

If Y has a Gumbel distribution then plotting the sorted values \( y_i \) versus the reduced values \( r_i \) should appear linear. The reduced values are given by

\[
 r_i = -\log(-\log(p_i))
\]

where \( p_i \) is the \( i \)th plotting position, taken here to be \((i - 0.5)/n\). Here, \( n \) is the number of observations. Curvature upwards/downwards may indicate a Frechet/Weibull distribution, respectively. Outliers may also be detected using this plot.

The function `guplot` is generic, and `guplot.default` and `guplot.vlm` are some methods functions for Gumbel plots.

Value

A list is returned invisibly with the following components.

- **x**: The reduced data.
- **y**: The sorted y data.

Note

The Gumbel distribution is a special case of the GEV distribution with shape parameter equal to zero.

Author(s)

T. W. Yee

References


See Also

gumbel, egumbel, gev, venice.
Examples

```r
## Not run: guplot(rnorm(500), las = 1) -> ii
names(ii)

guplot(with(venice, r1), col = "blue") # Venice sea levels data

## End(Not run)
```

hatvalues

Hat Values and Regression Deletion Diagnostics

Description

When complete, a suite of functions that can be used to compute some of the regression (leave-one-out deletion) diagnostics, for the VGLM class.

Usage

```r
hatvalues(model, ...)
hatvaluesvlm(model, type = c("diagonal", "matrix", "centralBlocks"), ...)
hatplot(model, ...)
hatplot.vlm(model, multiplier = c(2, 3), lty = "dashed",
           xlab = "Observation", ylab = "Hat values", ylim = NULL, ...)
dfbetavlm(model, maxit.new = 1,
          trace.new = FALSE,
          smallno = 1.0e-8, ...)
```

Arguments

- `model` - an R object, typically returned by `vglm`.
- `type` - Character. The default is the first choice, which is a $nM \times nM$ matrix. If `type = "matrix"` then the entire hat matrix is returned. If `type = "centralBlocks"` then $n$ central $M \times M$ block matrices, in matrix-band format.
- `multiplier` - Numeric, the multiplier. The usual rule-of-thumb is that values greater than two or three times the average leverage (at least for the linear model) should be checked.
- `lty, xlab, ylab, ylim` - Graphical parameters, see `par` etc. The default of `ylim = c(0, max(hatvalues(model)))` which means that if the horizontal dashed lines cannot be seen then there are no particularly influential observations.
- `maxit.new, trace.new, smallno` - Having `maxit.new = 1` will give a one IRLS step approximation from the ordinary solution (and no warnings!). Else having `maxit.new = 10`, say, should usually mean convergence will occur for all observations when they are removed one-at-a-time. Else having `maxit.new = 2`, say, should usually mean some lack of convergence will occur when observations are removed one-at-a-time. Setting
trace.new = TRUE will produce some running output at each IRLS iteration and for each individual row of the model matrix. The argument smallno multiplies each value of the original prior weight (often unity); setting it identically to zero will result in an error, but setting a very small value effectively removes that observation.

... further arguments, for example, graphical parameters for hatplot.vlm().

Details

The invocation hatvalues(vglmObject) should return a \( n \times M \) matrix of the diagonal elements of the hat (projection) matrix of a vglm object. To do this, the QR decomposition of the object is retrieved or reconstructed, and then straightforward calculations are performed.

The invocation hatplot(vglmObject) should plot the diagonal of the hat matrix for each of the \( M \) linear/additive predictors. By default, two horizontal dashed lines are added; hat values higher than these ought to be checked.

Note

It is hoped, soon, that the full suite of functions described at influence.measures will be written for VGLMs. This will enable general regression deletion diagnostics to be available for the entire VGLM class.

Author(s)

T. W. Yee.

See Also

vglm, cumulative, influence.measures.

Examples

# Proportional odds model, p.179, in McCullagh and Nelder (1989)

pneumo <- transform(pneumo, let = log(exposure.time))

fit <- vglm(cbind(normal, mild, severe) ~ let, cumulative, data = pneumo)

hatvalues(fit)  # n \times M matrix, with positive values

all.equal(sum(hatvalues(fit)), fit@rank)  # Should be TRUE

## Not run: par(mfrow = c(1, 2))

hatplot(fit, ylim = c(0, 1), las = 1, col = "blue")

## End(Not run)
Description

A hormone assay data set from Carroll and Ruppert (1988).

Usage

data(hormone)

Format

A data frame with 85 observations on the following 2 variables.

\( X \) a numeric vector, suitable as the x-axis in a scatter plot. The reference method.

\( Y \) a numeric vector, suitable as the y-axis in a scatter plot. The test method.

Details

The data is given in Table 2.4 of Carroll and Ruppert (1988), and was downloaded from http://www.stat.tamu.edu/~carroll. The book describes the data as follows. The data are the results of two assay methods for hormone data; the scale of the data as presented is not particularly meaningful, and the original source of the data refused permission to divulge further information. As in a similar example of Leurgans (1980), the old or reference method is being used to predict the new or test method. The overall goal is to see whether we can reproduce the test-method measurements with the reference-method measurements. Thus calibration might be of interest for the data.

References


See Also

uninormal, rrvglm.
Examples

```r
## Not run:
data(hormone)
summary(hormone)

modelI <- rrvglm(Y ~ 1 + X, data = hormone, trace = TRUE,
                  uninormal(zero = NULL, lsd = "identitylink", imethod = 2))

# Alternative way to fit modelI
modelI.other <- vglm(Y ~ 1 + X, data = hormone, trace = TRUE,
                     uninormal(zero = NULL, lsd = "identitylink"))

# Inferior to modelI
modelII <- vglm(Y ~ 1 + X, data = hormone, trace = TRUE,
                family = uninormal(zero = NULL))

logLik(modelI)
logLik(modelII) # Less than logLik(modelI)

# Reproduce the top 3 equations on p.65 of Carroll and Ruppert (1988).
# They are called Equations (1)--(3) here.

# Equation (1)
hormone <- transform(hormone, rX = 1 / X)
clist <- list("(Intercept)" = diag(2), X = diag(2), rX = rbind(0, 1))
fit1 <- vglm(Y ~ 1 + X + rX, family = uninormal(zero = NULL),
             constraints = clist, data = hormone, trace = TRUE)
coef(fit1, matrix = TRUE)
summary(fit1) # Actually, the intercepts do not seem significant
plot(Y ~ X, hormone, col = "blue")
lines(fitted(fit1) ~ X, hormone, col = "orange")

# Equation (2)
fit2 <- rrvglm(Y ~ 1 + X, uninormal(zero = NULL), hormone, trace = TRUE)
coef(fit2, matrix = TRUE)
plot(Y ~ X, hormone, col = "blue")
lines(fitted(fit2) ~ X, hormone, col = "red")
# Add + 2 SEs
lines(fitted(fit2) + 2 * exp(predict(fit2)[, "loge(sd)"] - X, hormone, col = "orange")
lines(fitted(fit2) - 2 * exp(predict(fit2)[, "loge(sd)"] - X, hormone, col = "orange")

# Equation (3)
# Does not fit well because the loge link for the mean is not good.
fit3 <- rrvglm(Y ~ 1 + X, maxit = 300, data = hormone, trace = TRUE,
               uninormal(lmean = "loge", zero = NULL))
coef(fit3, matrix = TRUE)
plot(Y ~ X, hormone, col = "blue") # Does not look okay.
lines(exp(predict(fit3)[, 1]) ~ X, hormone, col = "red")
# Add + 2 SEs
```
hspider

Hunting Spider Data

Description

Abundance of hunting spiders in a Dutch dune area.

Usage

data(hspider)

Format

A data frame with 28 observations (sites) on the following 18 variables.

WaterCon  Log percentage of soil dry mass.
BareSand  Log percentage cover of bare sand.
FallTwig  Log percentage cover of fallen leaves and twigs.
CoveMoss  Log percentage cover of the moss layer.
CoveHerb  Log percentage cover of the herb layer.
ReflLux   Reflection of the soil surface with cloudless sky.
Alopacce  Abundance of Alopecosa accentuata.
Alopcune  Abundance of Alopecosa cuneata.
Alopfabr  Abundance of Alopecosa fabrilis.
Arctlute  Abundance of Arctosa lutetiana.
Arctperi  Abundance of Arctosa perita.
Auloalbi  Abundance of Aulonia albimana.
Pardlugu  Abundance of Pardosa lugubris.
Pardmont  Abundance of Pardosa monticola.
Pardnigr  Abundance of Pardosa nigriceps.
Pardpull  Abundance of Pardosa pullata.
Trocterr  Abundance of Trochosa terricola.
Zoraspin  Abundance of Zora spinimana.
Details

The data, which originally came from Van der Aart and Smeek-Enserink (1975) consists of abundances (numbers trapped over a 60 week period) and 6 environmental variables. There were 28 sites.

This data set has been often used to illustrate ordination, e.g., using canonical correspondence analysis (CCA). In the example below, the data is used for constrained quadratic ordination (CQO; formerly called canonical Gaussian ordination or CGO), a numerically intensive method that has many superior qualities. See cqo for details.

References


Examples

```
summary(hspider)

## Not run:
# Standardize the environmental variables:
hsider[, 1:6] <- scale(subset(hspider, select = WaterCon:RefLux))

# Fit a rank-1 binomial CAO
hsbin <- hspider # Binary species data
hsbin[, -(1:6)] <- as.numeric(hsbin[, -(1:6)] > 0)
set.seed(123)
ahsb1 <- cao(cbind(Alopcune, Arctlute, Auloalbi, Zoraspin) ~ WaterCon + Reflux,
               family = binomialff(multiple.responses = TRUE),
               df1.nl = 2.2, Bestof = 3, data = hsbin)
par(mfrow = 2:1, las = 1)
lvplot(ahsb1, type = "predictors", llwd = 2, ylab = "logit p", lcol = 1:9)
persp(ahsb1, rug = TRUE, col = 1:10, lwd = 2)
coef(ahsb1)

## End(Not run)
```

huber2  

---

huber2  

**Huber’s least favourable distribution family function**

Description

M-estimation of the two parameters of Huber's least favourable distribution. The one parameter case is also implemented.
Usage

huber1(iloocation = "identitylink", k = 0.862, imethod = 1)
huber2(iloocation = "identitylink", lscale = "log", k = 0.862, imethod = 1, zero = 2)

Arguments

iloocation, lscale
Link functions applied to the location and scale parameters. See Links for more choices.
k
Tuning constant. See rhuber for more information.
imethod, zero
See CommonVGAMffArguments for information. The default value of zero means the scale parameter is modelled as an intercept-only.

Details

Huber’s least favourable distribution family function is popular for resistant/robust regression. The center of the distribution is normal and its tails are double exponential.

By default, the mean is the first linear/additive predictor (returned as the fitted values; this is the location parameter), and the log of the scale parameter is the second linear/additive predictor. The Fisher information matrix is diagonal; Fisher scoring is implemented.

The VGAM family function huber1() estimates only the location parameter. It assumes a scale parameter of unit value.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

Warning: actually, huber2() may be erroneous since the first derivative is not continuous when there are two parameters to estimate. huber1() is fine in this respect.

The response should be univariate.

Author(s)

T. W. Yee. Help was given by Arash Ardalan.

References


See Also

rhuber, uninormal, gaussianff, laplace, CommonVGAMffArguments.
Examples

```r
set.seed(1231); NN <- 30; coef1 <- 1; coef2 <- 10
hdata <- data.frame(x2 = sort(runif(NN)))
hdata <- transform(hdata, y = rhuber(NN, mu = coef1 + coef2 * x2))

hdata$x2[1] <- 0.0  # Add an outlier
hdata$y[1] <- 10

fit.huber2 <- vglm(y ~ x2, huber2(imethod = 3), data = hdata, trace = TRUE)
fitted <- vglm(y ~ x2, huber2(imethod = 3), data = hdata, trace = TRUE)

coef(fit.huber2, matrix = TRUE)
summary(fit.huber2)

## Not run:  # Plot the results
plot(y ~ x2, data = hdata, col = "blue", las = 1)
lines(fitted(fit.huber2) ~ x2, data = hdata, col = "darkgreen", lwd = 2)

fit.lm <- lm(y ~ x2, data = hdata)  # Compare to a LM:
lines(fitted(fit.lm) ~ x2, data = hdata, col = "lavender", lwd = 3)

# Compare to truth:
lines(coef1 + coef2 * x2 ~ x2, data = hdata, col = "orange", lwd = 2, lty = "dashed")

legend("bottomright", legend = c("truth", "huber", "lm"),
  col = c("orange", "darkgreen", "lavender"),
  lty = c("dashed", "solid", "solid"), lwd = c(2, 2, 3))
## End(Not run)
```

---

**Table 1 of Huggins (1989)**

---

Description

Simulated capture data set for the linear logistic model depending on an occasion covariate and an individual covariate for 10 trapping occasions and 20 individuals.

Usage

```r
data(Huggins89table1)
data(Huggins89.t1)
```

Format

The format is a data frame.
Details

Table 1 of Huggins (1989) gives this toy data set. Note that variables $t_1,...,t_{10}$ are occasion-specific variables. They correspond to the response variables $y_1,...,y_{10}$ which have values 1 for capture and 0 for not captured.

Both Huggins89table1 and Huggins89.t1 are identical. The latter used variables beginning with $z$, not $t$, and may be withdrawn very soon.

References


Examples

```r
Huggins89table1 <- transform(Huggins89table1, x3.tij = t01,
                           T02 = t02, T03 = t03, T04 = t04, T05 = t05, T06 = t06,
                           T07 = t07, T08 = t08, T09 = t09, T10 = t10)
small.table1 <- subset(Huggins89table1,
                       y01 + y02 + y03 + y04 + y05 + y06 + y07 + y08 + y09 + y10 > 0)
# fit.tbh is the bottom equation on p.133.
# It is a M_tbh model.
fit.tbh <-
  vglm(cbind(y01, y02, y03, y04, y05, y06, y07, y08, y09, y10) ~ x2 + x3.tij,
       xij = list(x3.tij ~ t01 + t02 + t03 + t04 + t05 + t06 + t07 + t08 + t09 + t10 +
                   T02 + T03 + T04 + T05 + T06 + T07 + T08 + T09 + T10 - 1),
       posbernoulli.t(parallel.t = TRUE ~ x2 + x3.tij),
       data = small.table1, trace = TRUE,
       form2 = ~ x2 + x3.tij +
                   t01 + t02 + t03 + t04 + t05 + t06 + t07 + t08 + t09 + t10 +
                   T02 + T03 + T04 + T05 + T06 + T07 + T08 + T09 + T10 - 1),
       posbernoulli.b(parallel.b = FALSE),
       data = small.table1, trace = TRUE)
# These results differ a bit from Huggins (1989), probably because
# two animals had to be removed here (they were never caught):
coef(fit.tbh) # First element is the behavioural effect
sqrt(diag(vcov(fit.tbh)))) # SEs
constraints(fit.tbh, matrix = TRUE)
summary(fit.tbh, presid = FALSE)
fit.tbh@extra$N.hat # Estimate of the population site N; cf. 20.06
fit.tbh@extra$SE.N.hat # Its standard error; cf. 1.87 or 4.51

fit.th <- vglm(cbind(y01, y02, y03, y04, y05, y06, y07, y08, y09, y10) ~ x2,
              posbernoulli.t, data = small.table1, trace = TRUE)
coef(fit.th)
constraints(fit.th)
coef(fit.th, matrix = TRUE) # M_th model
summary(fit.th, presid = FALSE)
fit.th@extra$N.hat # Estimate of the population size N
fit.th@extra$SE.N.hat # Its standard error

fit.bh <- vglm(cbind(y01, y02, y03, y04, y05, y06, y07, y08, y09, y10) ~ x2,
               posbernoulli.b(parallel.b = FALSE), data = small.table1, trace = TRUE)
coef(fit.bh)
```
The `hunua` data frame has 392 rows and 18 columns. Altitude is explanatory, and there are binary responses (presence/absence = 1/0 respectively) for 17 plant species.

### Usage

```r
data(hunua)
```

### Format

This data frame contains the following columns:

- **agaaus**  Agathis australis, or Kauri
- **beitaw**  Beilschmiedia tawa, or Tawa
- **corlae**  Corynocarpus laeavigatus
- **cyadea**  Cyathea dealbata
- **cyamed**  Cyathea medullaris
- **daccup**  Dacrydium cupressinum
- **dacdac**  Dacrycarpus dacrydioides
- **eladen**  Elaeocarpus dentatus

---

### Description

The `hunua` data frame has 392 rows and 18 columns. Altitude is explanatory, and there are binary responses (presence/absence = 1/0 respectively) for 17 plant species.
hedarb  Hedycarya arborea
hohpop  Species name unknown
kniexc  Knightia excelsa, or Rewarewa
kuneri  Kunzea ericoides
lepsco  Leptospermum scoparium
metrob  Metrosideros robusta
neslan  Nestegis lanceolata
rhosap  Rhopalostylis sapida
vitluc  Vitex lucens, or Puriri
altitude  meters above sea level

Details

These were collected from the Hunua Ranges, a small forest in southern Auckland, New Zealand. At 392 sites in the forest, the presence/absence of 17 plant species was recorded, as well as the altitude. Each site was of area size 200m².

Source

Dr Neil Mitchell, University of Auckland.

See Also

waitakere.

Examples

# Fit a GAM using vgam() and compare it with the Waitakere Ranges one
fit.h <- vgam(agaaus ~ s(altitude, df = 2), binomialff, data = hunua)
## Not run:
plot(fit.h, se = TRUE, lcol = "orange", scol = "orange",
     llwd = 2, slwd = 2, main = "Orange is Hunua, Blue is Waitakere")
## End(Not run)
head(predict(fit.h, hunua, type = "response"))

fit.w <- vgam(agaaus ~ s(altitude, df = 2), binomialff, data = waitakere)
## Not run:
plot(fit.w, se = TRUE, lcol = "blue", scol = "blue", add = TRUE)
## End(Not run)
head(predict(fit.w, hunua, type = "response"))  # Same as above?
hyperg

Hypergeometric Family Function

Description

Family function for a hypergeometric distribution where either the number of white balls or the total number of white and black balls are unknown.

Usage

hyperg(N = NULL, D = NULL, lprob = "logit", iprob = NULL)

Arguments

N  Total number of white and black balls in the urn. Must be a vector with positive values, and is recycled, if necessary, to the same length as the response. One of N and D must be specified.

D  Number of white balls in the urn. Must be a vector with positive values, and is recycled, if necessary, to the same length as the response. One of N and D must be specified.

lprob  Link function for the probabilities. See Links for more choices.

iprob  Optional initial value for the probabilities. The default is to choose initial values internally.

Details

Consider the scenario from dhyper where there are \( N = m + n \) balls in an urn, where \( m \) are white and \( n \) are black. A simple random sample (i.e., without replacement) of \( k \) balls is taken. The response here is the sample proportion of white balls. In this document, \( N \) is \( N = m + n \), \( D \) is \( m \) (for the number of “defectives”, in quality control terminology, or equivalently, the number of marked individuals). The parameter to be estimated is the population proportion of white balls, viz. \( \text{prob} = m/(m + n) \).

Depending on which one of \( N \) and \( D \) is inputted, the estimate of the other parameter can be obtained from the equation \( \text{prob} = m/(m + n) \), or equivalently, \( \text{prob} = D/N \). However, the log-factorials are computed using \( \text{lgamma} \) and both \( m \) and \( n \) are not restricted to being integer. Thus if an integer \( N \) is to be estimated, it will be necessary to evaluate the likelihood function at integer values about the estimate, i.e., at \( \text{trunc}(\hat{N}) \) and \( \text{ceiling}(\hat{N}) \) where \( \hat{N} \) is the (real) estimate of \( N \).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, vgam, rrvglm, cql, and cao.

Warning

No checking is done to ensure that certain values are within range, e.g., \( k \leq N \).
Note

The response can be of one of three formats: a factor (first level taken as success), a vector of proportions of success, or a 2-column matrix (first column = successes) of counts. The argument weights in the modelling function can also be specified. In particular, for a general vector of proportions, you will need to specify weights because the number of trials is needed.

Author(s)

Thomas W. Yee

References


See Also

dhyper, binomialff.

Examples

```r
nn <- 100
m <- 5 # Number of white balls in the population
k <- rep(4, len = nn) # Sample sizes
n <- 4 # Number of black balls in the population
y <- rhyper(nn = nn, m = m, n = n, k = k)
yprop <- y / k # Sample proportions

# N is unknown, D is known. Both models are equivalent:
fit <- vglm(cbind(y, k-y) ~ 1, hyperg(D = m), trace = TRUE, crit = "e")
fit <- vglm(yprop ~ 1, hyperg(D = m), weight = k, trace = TRUE, crit = "e")

# N is known, D is unknown. Both models are equivalent:
fit <- vglm(cbind(y, k-y) ~ 1, hyperg(N = m+n), trace = TRUE, crit = "l")
fit <- vglm(yprop ~ 1, hyperg(N = m+n), weight = k, trace = TRUE, crit = "l")

coef(fit, matrix = TRUE)
Coef(fit) # Should be equal to the true population proportion
unique(m / (m+n)) # The true population proportion
fit@extra
head(fitted(fit))
summary(fit)
```

**hypersecant**

Hyperbolic Secant Distribution Family Function

Description

Estimation of the parameter of the hyperbolic secant distribution.
Usage

hypersecant(link.theta = extlogit(min = -pi/2, max = pi/2), init.theta = NULL)
hypersecant01(link.theta = extlogit(min = -pi/2, max = pi/2), init.theta = NULL)

Arguments

link.theta Parameter link function applied to the parameter θ. See Links for more choices.
init.theta Optional initial value for θ. If failure to converge occurs, try some other value. The default means an initial value is determined internally.

Details

The probability density function of the hyperbolic secant distribution is given by

\[ f(y; \theta) = \exp(\theta y + \log(\cos(\theta)))/(2 \cosh(\pi y/2)), \]

for parameter \(-\pi/2 < \theta < \pi/2\) and all real \(y\). The mean of \(Y\) is \(\tan(\theta)\) (returned as the fitted values). Morris (1982) calls this model NEF-HS (Natural Exponential Family-Hyperbolic Secant). It is used to generate NEFs, giving rise to the class of NEF-GHS (G for Generalized).

Another parameterization is used for hypersecant01(): let \(Y = (\text{logit}U)/\pi\). Then this uses

\[ f(u; \theta) = (\cos(\theta)/\pi) \times u^{-0.5+\theta/\pi} \times (1 - u)^{-0.5-\theta/\pi}, \]

for parameter \(-\pi/2 < \theta < \pi/2\) and \(0 < u < 1\). Then the mean of \(U\) is \(0.5 + \theta/\pi\) (returned as the fitted values) and the variance is \((\pi^2 - 4\theta^2)/(8\pi^2)\).

For both parameterizations Newton-Raphson is same as Fisher scoring.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Author(s)

T. W. Yee

References


See Also

extlogit.
Examples

```r
hdata <- data.frame(x2 = rnorm(nn <- 200))
hdata <- transform(hdata, y = rnorm(nn))  # Not very good data!
fit1 <- vglm(y ~ x2, hypersecant, data = hdata, trace = TRUE, crit = "coef")
coef(fit1, matrix = TRUE)
fit1@misc$eargs

# Not recommended:
fit2 <- vglm(y ~ x2, hypersecant(link = "identitylink"), data = hdata, trace = TRUE)
coef(fit2, matrix = TRUE)
fit2@misc$eargs
```

### Hzeta

**Haight’s Zeta Distribution**

#### Description

Density, distribution function, quantile function and random generation for Haight’s zeta distribution with parameter \( \alpha \).

#### Usage

```r
dhzeta(x, alpha, log = FALSE)
phzeta(q, alpha, log.p = FALSE)
qhzeta(p, alpha)
rhzeta(n, alpha)
```

#### Arguments

- **x, q** Vector of quantiles. For the density, it should be a vector with positive integer values in order for the probabilities to be positive.
- **p** vector of probabilities.
- **n** number of observations. Same as `runif`.
- **alpha** The parameter value. Must contain positive values and is recycled to the length of \( x \) or \( p \) or \( q \) if necessary.
- **log** Logical. If `log = TRUE` then the logarithm of the density is returned.
- **log.p** Same meaning as in `pnorm` or `qnorm`.

#### Details

The probability function is

\[
f(x) = (2x - 1)^{(-\alpha)} - (2x + 1)^{(-\alpha)},
\]

where \( \alpha > 0 \) and \( x = 1, 2, \ldots \).
Value

dhzeta gives the density, phzeta gives the distribution function, qhzeta gives the quantile function, and rhzeta generates random deviates.

Note

Given some response data, the VGAM family function hzeta estimates the parameter alpha.

Author(s)

T. W. Yee and Kai Huang

See Also

hzeta, zeta, zetaff, simulate.vlm.

Examples

dhzeta(1:20, 2.1)
rhzeta(20, 2.1)

round(1000 * dhzeta(1:8, 2))
table(rhzeta(1000, 2))

## Not run: alpha <- 1.1; x <- 1:10
plot(x, dhzeta(x, alpha = alpha), type = "h", ylim = 0:1, lwd = 2,
     sub = paste("alpha =", alpha, las = 1, col = "blue", ylab = "Probability",
               main = "Haight's zeta: blue = density; orange = distribution function")
lines(x+0.1, phzeta(x, alpha = alpha), col = "orange", lty = 3, lwd = 2,
      type = "h")

## End(Not run)
hzeta

Arguments

- **link**: Parameter link function for the parameter. See Links for more choices. Here, a log-log link keeps the parameter greater than one, meaning the mean is finite.
- **ialpha**: Optional initial value for the (positive) parameter. The default is to obtain an initial value internally. Use this argument if the default fails.
- **nsimEIM**: See CommonVGAMffArguments for more information.

Details

The probability function is

\[ f(y) = (2y - 1)^{(-\alpha)} - (2y + 1)^{(-\alpha)}, \]

where the parameter \( \alpha > 0 \) and \( y = 1, 2, \ldots \). The function \texttt{dhzeta} computes this probability function. The mean of \( Y \), which is returned as fitted values, is \( (1 - 2^{-\alpha})\zeta(\alpha) \) provided \( \alpha > 1 \), where \( \zeta \) is Riemann's zeta function. The mean is a decreasing function of \( \alpha \). The mean is infinite if \( \alpha \leq 1 \), and the variance is infinite if \( \alpha \leq 2 \).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as \texttt{vglm} and \texttt{vgam}.

Author(s)

T. W. Yee

References


See Also

- \texttt{hzeta}, \texttt{zeta}, \texttt{zetaff}, \texttt{loglog}, \texttt{simulate.vlm}.

Examples

```r
alpha <- exp(exp(-0.1))  # The parameter
hdata <- data.frame(y = rhzeta(n = 1000, alpha))
fit <- vglm(y ~ 1, hzeta, data = hdata, trace = TRUE, crit = "coef")
coef(fit, matrix = TRUE)
Coef(fit)  # Useful for intercept-only models; should be same as alpha
c(with(hdata, mean(y)), head(fitted(fit), 1))
summary(fit)
```
Index from Array to Matrix

**Description**
Maps the elements of an array containing symmetric positive-definite matrices to a matrix with sufficient columns to hold them (called matrix-band format.)

**Usage**
```
iam(j, k, M, both = FALSE, diag = TRUE)
```

**Arguments**
- **j**: An integer from the set \{1:M\} giving the row number of an element.
- **k**: An integer from the set \{1:M\} giving the column number of an element.
- **M**: The number of linear/additive predictors. This is the dimension of each positive-definite symmetric matrix.
- **both**: Logical. Return both the row and column indices? See below for more details.
- **diag**: Logical. Return the indices for the diagonal elements? If FALSE then only the strictly upper triangular part of the matrix elements are used.

**Details**
Suppose we have \(n\) symmetric positive-definite square matrices, each \(M\) by \(M\), and these are stored in an array of dimension \(c(n, M, M)\). Then these can be more compactly represented by a matrix of dimension \(c(n, K)\) where \(K\) is an integer between \(M\) and \(M(M+1)/2\) inclusive. The mapping between these two representations is given by this function. It firstly enumerates by the diagonal elements, followed by the band immediately above the diagonal, then the band above that one, etc. The last element is \(H1, M\). This function performs the mapping from elements \((j, k)\) of symmetric positive-definite square matrices to the columns of another matrix representing such. This is called the matrix-band format and is used by the VGAM package.

**Value**
This function has a dual purpose depending on the value of both. If both=FALSE then the column number corresponding to the \(j\)-\(k\) element of the matrix is returned. If both = TRUE then \(j\) and \(k\) are ignored and a list with the following components are returned.

- **row.index**: The row indices of the upper triangular part of the matrix (This may or may not include the diagonal elements, depending on the argument diagonal).
- **col.index**: The column indices of the upper triangular part of the matrix (This may or may not include the diagonal elements, depending on the argument diagonal).
Note
This function is used in the weight slot of many VGAM family functions (see vglmff-class), especially those whose M is determined by the data, e.g., dirichlet, multinomial.

Author(s)
T. W. Yee

See Also
vglmff-class.

Examples
iam(1, 2, M = 3)  # The 4th column represents element (1,2) of a 3x3 matrix
iam(NULL, NULL, M = 3, both = TRUE)  # Return the row and column indices

    dirichlet()@weight

    M <- 4
    temp1 <- iam(NA, NA, M = M, both = TRUE)
    mat1 <- matrix(NA, M, M)
    mat1[cbind(temp1$row, temp1$col)] = 1:length(temp1$row)
    mat1  # More commonly used

    temp2 <- iam(NA, NA, M = M, both = TRUE, diag = FALSE)
    mat2 <- matrix(NA, M, M)
    mat2[cbind(temp2$row, temp2$col)] = 1:length(temp2$row)
    mat2  # Rarely used

identitylink  

Identity Link Function

Description
Computes the identity transformation, including its inverse and the first two derivatives.

Usage

    identitylink(theta, inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)
    negidentity(theta, inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)

Arguments

theta  
Numeric or character. See below for further details.

inverse, deriv, short, tag
Details at Links.
Details

The identity link function $g(\theta) = \theta$ should be available to every parameter estimated by the `VGAM` library. However, it usually results in numerical problems because the estimates lie outside the permitted range. Consequently, the result may contain `Inf`, `-Inf`, `NA` or `NaN`.

The function `negidentity` is the negative-identity link function and corresponds to $g(\theta) = -\theta$. This is useful for some models, e.g., in the literature supporting the `egev` function it seems that half of the authors use $\xi = -k$ for the shape parameter and the other half use $k$ instead of $\xi$.

Value

For `identitylink()`: for `deriv = 0`, the identity of theta, i.e., theta when `inverse = FALSE`, and if `inverse = TRUE` then theta. For `deriv = 1`, then the function returns $d \theta / d \eta$ as a function of theta if `inverse = FALSE`, else if `inverse = TRUE` then it returns the reciprocal.

For `negidentity()`: the results are similar to `identitylink()` except for a sign change in most cases.

Author(s)

Thomas W. Yee

References


See Also

`links`, `loge`, `logit`, `probit`, `powerlink`.

Examples

```r
identitylink((-5):5)
identitylink((-5):5, deriv = 1)
identitylink((-5):5, deriv = 2)
negidentity((-5):5)
negidentity((-5):5, deriv = 1)
negidentity((-5):5, deriv = 2)
```
inv.binomial

Usage

inv.binomial(lrho = extlogit(min = 0.5, max = 1),
     ilambda = "loge", irho = NULL, ilambda = NULL, zero = NULL)

Arguments

lrho, ilambda  Link function for the ρ and λ parameters. See Links for more choices.
irho, ilambda  Numeric. Optional initial values for ρ and λ.
zero           See CommonVGAMffArguments.

Details

The inverse binomial distribution of Yanagimoto (1989) has density function

\[ f(y; \rho, \lambda) = \frac{\lambda \Gamma(2y + \lambda)}{\Gamma(y + 1) \Gamma(y + \lambda + 1)} \{\rho(1 - \rho)\}^y \rho^\lambda \]

where \( y = 0, 1, 2, \ldots \) and \( \frac{1}{2} < \rho < 1 \), and \( \lambda > 0 \). The first two moments exist for \( \rho > \frac{1}{2} \); then the mean is \( \lambda(1 - \rho)/(2\rho - 1) \) (returned as the fitted values) and the variance is \( \lambda \rho(1 - \rho)/(2\rho - 1)^3 \). The inverse binomial distribution is a special case of the generalized negative binomial distribution of Jain and Consul (1971). It holds that \( \text{Var}(Y) > E(Y) \) so that the inverse binomial distribution is overdispersed compared with the Poisson distribution.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

This VGAM family function only works reasonably well with intercept-only models. Good initial values are needed; if convergence failure occurs use irho and/or ilambda.

Some elements of the working weight matrices use the expected information matrix while other elements use the observed information matrix. Yet to do: using the mean and the reciprocal of \( \lambda \) results in an EIM that is diagonal.

Author(s)

T. W. Yee

References

Inv.gaussian

The Inverse Gaussian Distribution

Description

Density, distribution function and random generation for the inverse Gaussian distribution.

Usage

dinv.gaussian(x, mu, lambda, log = FALSE)
pinv.gaussian(q, mu, lambda)
rinv.gaussian(n, mu, lambda)

Arguments

x, q vector of quantiles.
n number of observations. If length(n) > 1 then the length is taken to be the number required.
mu the mean parameter.
lambda the $\lambda$ parameter.
log Logical. If log = TRUE then the logarithm of the density is returned.

Details

See inv.gaussianff, the VGAM family function for estimating both parameters by maximum likelihood estimation, for the formula of the probability density function.

Value

dinv.gaussian gives the density, pinv.gaussian gives the distribution function, and rinv.gaussian generates random deviates.

See Also

negbinomial, poissonff.
Note
Currently qinv.gaussian is unavailable.

Author(s)
T. W. Yee

References

See Also
inv.gaussianff, waldff.

Examples
## Not run: x <- seq(-0.05, 4, len = 300)
plot(x, dinv.gaussian(x, mu = 1, lambda = 1), type = "l",
col = "blue", las = 1, main =
"blue is density, orange is cumulative distribution function")
abline(h = 0, col = "gray", lty = 2)
lines(x, pinv.gaussian(x, mu = 1, lambda = 1), type = "l", col = "orange")
## End(Not run)

inv.gaussianff

Inverse Gaussian Distribution Family Function

Description
Estimates the two parameters of the inverse Gaussian distribution by maximum likelihood estimation.

Usage
inv.gaussianff(lmu = "loge", llambda = "loge",
imethod = 1, ilambda = NULL,
parallel = FALSE, ishrinkage = 0.99, zero = NULL)
inv.gaussianff

Arguments

- lmu, llambda: Parameter link functions for the $\mu$ and $\lambda$ parameters. See Links for more choices.
- llambda, parallel: See CommonVGAMffArguments for more information. If parallel = TRUE then the constraint is not applied to the intercept.
- ilambda, ishrinkage, zero: See CommonVGAMffArguments for more information.

Details

The standard ("canonical") form of the inverse Gaussian distribution has a density that can be written as

$$f(y; \mu, \lambda) = \sqrt{\lambda/(2\pi y^3)} \exp \left(-\lambda(y - \mu)^2/(2\mu^2 y)\right)$$

where $y > 0$, $\mu > 0$, and $\lambda > 0$. The mean of $Y$ is $\mu$ and its variance is $\mu^3/\lambda$. By default, $\eta_1 = \log(\mu)$ and $\eta_2 = \log(\lambda)$. The mean is returned as the fitted values. This VGAM family function can handle multiple responses (inputted as a matrix).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Note

The inverse Gaussian distribution can be fitted (to a certain extent) using the usual GLM framework involving a scale parameter. This family function is different from that approach in that it estimates both parameters by full maximum likelihood estimation.

Author(s)

T. W. Yee

References


See Also

Inv.gaussian, waldff, bisa.

The R package SuppDists has several functions for evaluating the density, distribution function, quantile function and generating random numbers from the inverse Gaussian distribution.
inv.lomax

Examples

idata <- data.frame(x2 = runif(nn <- 1000))
idata <- transform(idata, mymu = exp(2 + 1 * x2),
                   Lambda = exp(2 + 1 * x2))
idata <- transform(idata, y = rinv.gaussian(nn, mu = mymu, lambda = Lambda))
fit1 <- vglm(y ~ x2, inv.gaussianff, data = idata, trace = TRUE)
rrig <- rrvglm(y ~ x2, inv.gaussianff, data = idata, trace = TRUE)
coef(fit1, matrix = TRUE)
coef(rrig, matrix = TRUE)
Coef(rrig)
summary(fit1)

---

Inv.lomax

*The Inverse Lomax Distribution*

Description

Density, distribution function, quantile function and random generation for the inverse Lomax distribution with shape parameter \( \alpha \) and scale parameter \( \lambda \).

Usage

```r
dinv.lomax(x, scale = 1, shape2.p, log = FALSE)
pinv.lomax(q, scale = 1, shape2.p, lower.tail = TRUE, log.p = FALSE)
qinv.lomax(p, scale = 1, shape2.p, lower.tail = TRUE, log.p = FALSE)
rinv.lomax(n, scale = 1, shape2.p)
```

Arguments

- **x**, **q**
  - vector of quantiles.
- **p**
  - vector of probabilities.
- **n**
  - number of observations. If length(n) > 1, the length is taken to be the number required.
- **shape2.p**
  - shape parameter.
- **scale**
  - scale parameter.
- **log**
  - Logical. If log = TRUE then the logarithm of the density is returned.
- **lower.tail**, **log.p**
  - Same meaning as in `pnorm` or `qnorm`.

Details

See `inv.lomax`, which is the *VGAM* family function for estimating the parameters by maximum likelihood estimation.
Value

dinv.lomax gives the density, pinv.lomax gives the distribution function, qinv.lomax gives the quantile function, and rinv.lomax generates random deviates.

Note

The inverse Lomax distribution is a special case of the 4-parameter generalized beta II distribution.

Author(s)

T. W. Yee

References


See Also

inv.lomax, genbetaII.

Examples

```r
idata <- data.frame(y = rinv.lomax(n = 1000, exp(2), exp(1)))
fit <- vglm(y ~ 1, inv.lomax, data = idata, trace = TRUE, crit = "coef")
coef(fit, matrix = TRUE)
Coef(fit)
```

inv.lomax

Inverse Lomax Distribution Family Function

Description

Maximum likelihood estimation of the 2-parameter inverse Lomax distribution.

Usage

```r
inv.lomax(1scale = "loge", lshape2.p = "loge", iscale = NULL,
          ishape2.p = NULL, imethod = 1, gscale = exp(-5:5), gshape2.p = exp(-5:5),
          probs.y = c(0.25, 0.5, 0.75), zero = -2)
```
inv.lomax

Arguments

- `lscale, lshape2.p`  
  Parameter link functions applied to the (positive) parameters $b$, and $p$. See Links for more choices.

- `iscale, ishape2.p, imethod, zero`  
  See CommonVGAMffArguments for information. For `imethod = 2` a good initial value for `ishape2.p` is needed to obtain a good estimate for the other parameter.

- `gscale, gshape2.p`  
  See CommonVGAMffArguments for information.

- `probs.y`  
  See CommonVGAMffArguments for information.

Details

The 2-parameter inverse Lomax distribution is the 4-parameter generalized beta II distribution with shape parameters $a = q = 1$. It is also the 3-parameter Dagum distribution with shape parameter $a = 1$, as well as the beta distribution of the second kind with $q = 1$. More details can be found in Kleiber and Kotz (2003).

The inverse Lomax distribution has density

$$f(y) = py^{p-1}/[b^p(1 + y/b)^{p+1}]$$

for $b > 0$, $p > 0$, $y \geq 0$. Here, $b$ is the scale parameter `scale`, and $p$ is a shape parameter. The mean does not seem to exist; the median is returned as the fitted values. This family function handles multiple responses.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

See the notes in genbetaII.

Author(s)

T. W. Yee

References


See Also

inv.lomax, genbetaII, betaII, dagum, sinmad, fisk, lomax, paralogistic, inv.paralogistic, simulate.vlm.
The Inverse Paralogistic Distribution

Description

Density, distribution function, quantile function and random generation for the inverse paralogistic distribution with shape parameters \(a\) and \(p\), and scale parameter \(\text{scale}\).

Usage

\[
dinv.paralogistic(x, \text{scale} = 1, \text{shape1.a}, \text{log} = \text{FALSE})
\]
\[
pinv.paralogistic(q, \text{scale} = 1, \text{shape1.a}, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE})
\]
\[
qinv.paralogistic(p, \text{scale} = 1, \text{shape1.a}, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE})
\]
\[
rinv.paralogistic(n, \text{scale} = 1, \text{shape1.a})
\]

Arguments

- \(x, q\) vector of quantiles.
- \(p\) vector of probabilities.
- \(n\) number of observations. If \(\text{length}(n) > 1\), the length is taken to be the number required.
- \(\text{shape1.a}\) shape parameter.
- \(\text{scale}\) scale parameter.
- \(\text{log}\) Logical. If \(\text{log} = \text{TRUE}\) then the logarithm of the density is returned.
- \(\text{lower.tail}, \text{log.p}\) Same meaning as in \text{pnorm} or \text{qnorm}.

Details

See \text{inv.paralogistic}, which is the \text{VGAM} family function for estimating the parameters by maximum likelihood estimation.

Value

\(dinv.paralogistic\) gives the density, \(pinv.paralogistic\) gives the distribution function, \(qinv.paralogistic\) gives the quantile function, and \(rinv.paralogistic\) generates random deviates.
Note

The inverse paralogistic distribution is a special case of the 4-parameter generalized beta II distribution.

Author(s)

T. W. Yee

References


See Also

inv.paralogistic, genbetaII.

Examples

```r
idata <- data.frame(y = rinv.paralogistic(n = 3000, exp(1), scale = exp(2)))
fit <- vglm(y - 1, inv.paralogistic(lss = FALSE, ishape1.a = 2.1),
            data = idata, trace = TRUE, crit = "coef")
coef(fit, matrix = TRUE)
Coef(fit)
```

## inv.paralogistic  Inverse Paralogistic Distribution Family Function

Description

Maximum likelihood estimation of the 2-parameter inverse paralogistic distribution.

Usage

```r
inv.paralogistic(lscale = "loge", lshape1.a = "loge", iscale = NULL,
                 ishape1.a = NULL, imethod = 1, lss = TRUE, gscale = exp(-5:5),
                 gshape1.a = exp(-5:5), probs.y = c(0.25, 0.5, 0.75),
                 zero = ifelse(lss, -2, -1))
```

Arguments

1 ss  See CommonVGAMffArguments for important information.
1 lshape1.a, lscale  Parameter link functions applied to the (positive) parameters a and scale. See Links for more choices.
1 iscale, ishape1.a, imethod, zero  See CommonVGAMffArguments for information. For imethod = 2 a good initial value for ishape1.a is needed to obtain a good estimate for the other parameter.
The 2-parameter inverse paralogistic distribution is the 4-parameter generalized beta II distribution with shape parameter $q = 1$ and $a = p$. It is the 3-parameter Dagum distribution with $a = p$. More details can be found in Kleiber and Kotz (2003).

The inverse paralogistic distribution has density

$$f(y) = a^2 y^{a^2 - 1} / \left[ b^{a^2} \left\{ 1 + (y/b)^a \right\}^{a+1} \right]$$

for $a > 0$, $b > 0$, $y \geq 0$. Here, $b$ is the scale parameter scale, and $a$ is the shape parameter. The mean is

$$E(Y) = b \Gamma(a + 1/a) \Gamma(1 - 1/a) / \Gamma(a)$$

provided $a > 1$; these are returned as the fitted values. This family function handles multiple responses.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

See the notes in genbetaII.

Author(s)

T. W. Yee

References


See Also

Inv.paralogistic, genbetaII, betaII, dagum, sinmad, fisk, inv.lomax, lomax, paralogistic, simulate.vlm.

Examples

```r
idata <- data.frame(y = rinv.paralogistic(n = 3000, exp(1)), scale = exp(2)))
fit <- vglm(y ~ 1, inv.paralogistic(1ss = FALSE), data = idata, trace = TRUE)
fit <- vglm(y ~ 1, inv.paralogistic(1method = 2, ishape1.a = 4),
            data = idata, trace = TRUE, crit = "coef")
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)
```
is.buggy

Does the fitted object suffer from a known bug?

Description

Checks to see if a fitted object suffers from some known bug.

Usage

is.buggy(object, ...)
is.buggy.vlm(object, each.term = FALSE, ...)

Arguments

object A fitted VGAM object, e.g., from vgam.
each.term Logical. If TRUE then a logical is returned for each term.
... Unused for now.

Details

It is known that vgam with s terms do not correctly handle constraint matrices (cmat, say) when crossprod(cmat) is not diagonal. This function detects whether this is so or not. Note that probably all VGAM family functions have defaults where all crossprod(cmat)s are diagonal, therefore do not suffer from this bug. It is more likely to occur if the user inputs constraint matrices using the constraints argument (and setting zero = NULL if necessary).

Value

The default is a single logical (TRUE if any term is TRUE), otherwise a vector of such with each element corresponding to a term. If the value is TRUE then I suggest replacing the VGAM by a similar model fitted by vglm and using regression splines, e.g., bs, ns.

Note

When the bug is fixed this function may be withdrawn, otherwise always return FALSEs!

Author(s)

T. W. Yee

See Also

vgam, vglm, bs, ns.
Examples

```r
fit1 <- vgam(cbind(agaus, kniexc) ~ s(altitude, df = c(3, 4)),
              binomial(multiple.responses = TRUE), data = hunua)
is.buggy(fit1) # Okay
is.buggy(fit1, each.term = TRUE) # No terms are buggy

fit2 <- vgam(cbind(agaus, kniexc) ~ s(altitude, df = c(3, 4)),
              binomial(multiple.responses = TRUE), data = hunua,
              constraints =
              list("(Intercept)" = diag(2),
                   "s(altitude, df = c(3, 4))" = matrix(c(1, 1, 0, 1), 2, 2))
is.buggy(fit2) # TRUE
is.buggy(fit2, each.term = TRUE)
constraints(fit2)

# fit2b is an approximate alternative to fit2:
fit2b <- vglm(cbind(agaus, kniexc) ~ bs(altitude, df = 3) + bs(altitude, df = 4),
              binomial(multiple.responses = TRUE), data = hunua,
              constraints =
              list("(Intercept)" = diag(2),
                   "bs(altitude, df = 3)" = rbind(1, 1),
                   "bs(altitude, df = 4)" = rbind(0, 1))
is.buggy(fit2b) # Okay
is.buggy(fit2b, each.term = TRUE)
constraints(fit2b)
```

---

is.parallel  Parallelism Constraint Matrices

Description

Returns a logical vector from a test of whether an object such as a matrix or VGLM object corresponds to a parallelism assumption.

Usage

```r
is.parallel.matrix(object, ...)  
is.parallel.vglm(object, type = c("term", "lm"), ...)
```

Arguments

- object: an object such as a constraint matrix or a `vglm` object.
- type: passed into `constraints`.
- ...: additional optional arguments. Currently unused.

Details

These functions may be useful for categorical models such as `propodds, cumulative, acat, cratio, sratio, multinomial`.
is.smart

Test For a Smart Object

Description

Tests an object to see if it is smart.

Usage

is.smart(object)

Arguments

object a function or a fitted model.

Details

If object is a function then this function looks to see whether object has the logical attribute "smart". If so then this is returned, else FALSE.

If object is a fitted model then this function looks to see whether object@smart.prediction or object$smart.prediction exists. If it does and it is not equal to list(smart.arg=FALSE) then a TRUE is returned, else FALSE. The reason for this is because, e.g., lm(...,smart=FALSE) and vglm(...,smart=FALSE), will return such a specific list.

Writers of smart functions manually have to assign this attribute to their smart function after it has been written.
is.zero

Value

Returns TRUE or FALSE, according to whether the object is smart or not.

Examples

```r
is.smart(sm.min1) # TRUE
is.smart(sm.poly) # TRUE
library(splines)
is.smart(sm.bs)   # TRUE
is.smart(sm.ns)   # TRUE
is.smart(tan)     # FALSE
## Not run:
udata <- data.frame(x2 = rnorm(9))
fit1 <- vglm(rnorm(9) ~ x2, uninform, data = udata)
is.smart(fit1)    # TRUE
fit2 <- vglm(rnorm(9) ~ x2, uninform, data = udata, smart = FALSE)
is.smart(fit2)    # FALSE
fit2@smart.prediction
## End(Not run)
```

is.zero Zero Constraint Matrices

Description

Returns a logical vector from a test of whether an object such as a matrix or VGLM object corresponds to a 'zero' assumption.

Usage

```r
is.zero.matrix(object, ...)
is.zero.vglm(object, ...)
```

Arguments

- `object`: an object such as a coefficient matrix of a `vglm` object, or a `vglm` object.
- `...`: additional optional arguments. Currently unused.

Details

These functions test the effect of the `zero` argument on a `vglm` object or the coefficient matrix of a `vglm` object. The latter is obtained by `coef(vglmObject, matrix = TRUE)`.

Value

A vector of logicals, testing whether each linear/additive predictor has the `zero` argument applied to it. It is TRUE if that linear/additive predictor is intercept-only, i.e., all other regression coefficients are set to zero.

No checking is done for the intercept term at all, i.e., that it was estimated in the first place.
See Also
correlations, vglm.

Examples

coalminers <- transform(coalminers, Age = (age - 42) / 5)
fit <- vglm(cbind(nBwK, nBwK, BwK) ~ Age, binom2.or(zero = NULL), coalminers)
is.zero(fit)
is.zero(coef(fit, matrix = TRUE))
Warning

If length(x) is large then the cost is $O(N^2)$, which is expensive! Under these circumstances it is not advisable to set exact = TRUE or max.n to a very large number.

See Also

`binormalcop, cor`.

Examples

```r
N <- 5000; x <- 1:N; y <- runif(N)
true.rho <- -0.8
ymat <- rbinorm(N, cov12 = true.rho) # Bivariate normal, aka N_2
x <- ymat[, 1]
y <- ymat[, 2]

# Not run: plot(x, y, col = "blue")

kendall.tau(x, y) # A random sample is taken here
kendall.tau(x, y) # A random sample is taken here

kendall.tau(x, y, exact = TRUE) # Costly if length(x) is large
kendall.tau(x, y, max.n = N) # Same as exact = TRUE

(rhohat <- sin(2 * pi / 2)) # This formula holds for N_2 actually
true.rho # rhohat should be near this value
```

Kumar

The Kumaraswamy Distribution

Description

Density, distribution function, quantile function and random generation for the Kumaraswamy distribution.

Usage

```r
dkumar(x, shape1, shape2, log = FALSE)
pkumar(q, shape1, shape2, lower.tail = TRUE, log.p = FALSE)
qkumar(p, shape1, shape2, lower.tail = TRUE, log.p = FALSE)
rkumar(n, shape1, shape2)
```

Arguments

- `x, q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If length(n) > 1 then the length is taken to be the number required.
shape1, shape2  positive shape parameters.

log Logical. If log = TRUE then the logarithm of the density is returned.

lower.tail, log.p
Same meaning as in pnorm or qnorm.

Details

See kumar, the VGAM family function for estimating the parameters, for the formula of the probability density function and other details.

Value
dkumar gives the density, pkumar gives the distribution function, qkumar gives the quantile function, and rkumar generates random deviates.

Author(s)

T. W. Yee and Kai Huang

See Also

kumar.

Examples

## Not run:
shape1 <- 2; shape2 <- 2; nn <- 201; # shape1 <- shape2 <- 0.5;
x <- seq(-0.05, 1.05, len = nn)
plot(x, dkumar(x, shape1, shape2), type = "l", las = 1, ylim = c(0,1.5),
     ylab = paste("fkumar(shape1 = ", shape1, ", shape2 = ", shape2, ")",
     col = "blue", cex.main = 0.8,
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Purple lines are the 10,20,...,90 percentiles")
lines(x, pkumar(x, shape1, shape2), col = "orange")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qkumar(probs, shape1, shape2)
lines(Q, dkumar(Q, shape1, shape2), col = "purple", lty = 3, type = "h")
lines(Q, pkumar(Q, shape1, shape2), col = "purple", lty = 3, type = "h")
abline(h = probs, col = "purple", lty = 3)
max(abs(pkumar(Q, shape1, shape2) - probs)) # Should be 0

## End(Not run)
Kumaraswamy Distribution Family Function

Description

Estimates the two parameters of the Kumaraswamy distribution by maximum likelihood estimation.

Usage

```r
kumar(lshape1 = "loge", lshape2 = "loge",
     ishape1 = NULL, ishape2 = NULL, grid.shape1 = c(0.4, 6.0),
     tol12 = 1.0e-4, zero = NULL)
```

Arguments

- `lshape1, lshape2`
  Link function for the two positive shape parameters, respectively, called `a` and `b` below. See Links for more choices.
- `ishape1, ishape2`
  Numeric. Optional initial values for the two positive shape parameters.
- `tol12`
  Numeric and positive. Tolerance for testing whether the second shape parameter is either 1 or 2. If so then the working weights need to handle these singularities.
- `grid.shape1`
  Lower and upper limits for a grid search for the first shape parameter.
- `zero`
  See CommonVGAMffArguments.

Details

The Kumaraswamy distribution has density function

\[
f(y; a = \text{shape1}, b = \text{shape2}) = aby^{a-1}(1 - y^a)^{b-1}\]

where \(0 < y < 1\) and the two shape parameters, `a` and `b`, are positive. The mean is \(b \times Beta(1 + 1/a, b)\) (returned as the fitted values) and the variance is \(b \times Beta(1 + 2/a, b) - (b \times Beta(1 + 1/a, b))^2\). Applications of the Kumaraswamy distribution include the storage volume of a water reservoir. Fisher scoring is implemented. Handles multiple responses (matrix input).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Author(s)

T. W. Yee
References


See Also

dkumar, betaff, simulate.vlm.

Examples

```r
shape1 <- exp(1); shape2 <- exp(2)
kdata <- data.frame(y = rkumar(n = 1000, shape1, shape2))
fit <- vglm(y ~ 1, kumar, data = kdata, trace = TRUE)
c(with(kdata, mean(y)), head(fitted(fit), 1))
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)
```

LakeO

Annual catches on Lake Otamangakau from October 1974 to October 1989

Description

Rainbow and brown trout catches by Mr Swainson at Lake Otamangakau in the central North Island of New Zealand during the 1970s and 1980s.

Usage

data(LakeO)

Format

A data frame with 15 observations on the following 5 variables.

- `year`: a numeric vector, the season began on 1 October of the year and ended 12 months later.
- `total.fish`: a numeric vector, the total number of fish caught during the season. Simply the sum of brown and rainbow trout.
- `brown`: a numeric vector, the number of brown trout (*Salmo trutta*) caught.
- `rainbow`: a numeric vector, the number of rainbow trout (*Oncorhynchus mykiss*) caught.
- `visits`: a numeric vector, the number of visits during the season that the angler made to the lake.

It is necessary to assume that the visits were of an equal time length in order to interpret the usual Poisson regressions.
Details

The data was extracted from the season summaries at Lake Otamangakau by Anthony Swainson for the seasons 1974–75 to 1988–89.

Mr Swainson was one of a small group of regular fly fishing anglers and kept a diary of his catches. Lake Otamangakau is a lake of area 1.8 squared km and has a maximum depth of about 12m, and is located in the central North Island of New Zealand. It is trout-infested and known for its trophy-sized fish.

See also trap0.

Source

Table 7.2 of the reference below. Thanks to Dr Michel Dedual for a copy of the report and for help reading the final year’s data. The report is available from TWY on request.

References


Examples

data(lake0)
lake0
summary(lake0)

lambertW(x, tolerance = 1e-10, maxit = 50)

Arguments

x A vector of reals.
tolerance Accuracy desired.
maxit Maximum number of iterations of third-order Halley’s method.

Description

Computes the Lambert W function for real values.
Details

The Lambert $W$ function is the root of the equation $W(z) \exp(W(z)) = z$ for complex $z$. It is multi-valued if $z$ is real and $z < -1/e$. For real $-1/e \leq z < 0$ it has two possible real values, and currently only the upper branch is computed.

Value

This function returns the principal branch of the $W$ function for real $z$. It returns $W(z) \geq -1$, and NA for $z < -1/e$.

Note

If convergence does not occur then increase the value of maxit and/or tolerance.

Yet to do: add an argument lbranch = TRUE to return the lower branch for real $-1/e \leq z < 0$; this would give $W(z) \leq -1$.

Author(s)

T. W. Yee

References


See Also

log, exp.

Examples

```r
## Not run:
curve(lambertW, -exp(-1), 3, xlab = c(-1, 3), ylab = c(-2, 1),
      las = 1, col = "orange")
abline(v = -exp(-1), h = -1, lwd = 2, lty = "dotted", col = "gray")
abline(h = 0, v = 0, lty = "dashed", col = "blue")
## End(Not run)
```

### Description

Maximum likelihood estimation of the 2-parameter classical Laplace distribution.

### Usage

```r
laplace(llocation = "identitylink", lscale = "loge",
        ilocation = NULL, iscale = NULL, imethod = 1, zero = 2)
```
Arguments

location, scale
Character. Parameter link functions for location parameter \(a\) and scale parameter \(b\). See Links for more choices.

location, scale
Optional initial values. If given, it must be numeric and values are recycled to the appropriate length. The default is to choose the value internally.

imethod
Initialization method. Either the value 1 or 2.

default
See CommonVGAMffArguments for more information.

Details

The Laplace distribution is often known as the double-exponential distribution and, for modelling, has heavier tail than the normal distribution. The Laplace density function is

\[
f(y) = \frac{1}{2b} \exp \left( -\frac{|y - a|}{b} \right)
\]

where \(-\infty < y < \infty\), \(-\infty < a < \infty\) and \(b > 0\). Its mean is \(a\) and its variance is \(2b^2\). This parameterization is called the classical Laplace distribution by Kotz et al. (2001), and the density is symmetric about \(a\).

For \(y \sim 1\) (where \(y\) is the response) the maximum likelihood estimate (MLE) for the location parameter is the sample median, and the MLE for \(b\) is \(\text{mean}(\text{abs}(y - \text{location}))\) (replace location by its MLE if unknown).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

This family function has not been fully tested. The MLE regularity conditions do not hold for this distribution, therefore misleading inferences may result, e.g., in the summary and vcov of the object.

Note

This family function uses Fisher scoring. Convergence may be slow for non-intercept-only models; half-stepping is frequently required.

Author(s)

T. W. Yee

References

See Also

dlape, alaplace2 (which differs slightly from this parameterization), exponential, median.

Examples

```r
data <- data.frame(y = rlaplace(nn <- 100, loc = 2, scale = exp(1)))
fit <- vglm(y ~ 1, laplace, data = data, trace = TRUE, crit = "l")
coef(fit, matrix = TRUE)
Coef(fit)
with(data, median(y))

ldata <- data.frame(x = runif(nn <- 100))
data <- transform(ldata, y = rlaplace(nn, loc = 2, scale = exp(-1 + 1*x)))
coef(vglm(y ~ x, laplace(loc = 0.2, imethod = 2, zero = 1), data = data,
               trace = TRUE), matrix = TRUE)
```

laplaceUC  

The Laplace Distribution

Description

Density, distribution function, quantile function and random generation for the Laplace distribution
with location parameter location and scale parameter scale.

Usage

```r
dlaplace(x, location = 0, scale = 1, log = FALSE)
plaplace(q, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
qlaplace(p, location = 0, scale = 1, lower.tail = TRUE, log.p = FALSE)
rlaplace(n, location = 0, scale = 1)
```

Arguments

- `x, q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. Same as in `runif`.
- `location` the location parameter `a`, which is the mean.
- `scale` the scale parameter `b`. Must consist of positive values.
- `log` Logical. If `log = TRUE` then the logarithm of the density is returned.
- `lower.tail, log.p` Same meaning as in `pnorm` or `qnorm`.
The Laplace distribution is often known as the double-exponential distribution and, for modelling, has heavier tail than the normal distribution. The Laplace density function is

\[ f(y) = \frac{1}{2b} \exp\left(-\frac{|y-a|}{b}\right) \]

where \(-\infty < y < \infty\), \(-\infty < a < \infty\) and \(b > 0\). The mean is \(a\) and the variance is \(2b^2\). See `laplace`, the VGAM family function for estimating the two parameters by maximum likelihood estimation, for formulae and details. Apart from \(n\), all the above arguments may be vectors and are recycled to the appropriate length if necessary.

**Value**

dlaplace gives the density, plaplace gives the distribution function, qlaplace gives the quantile function, and rlaplace generates random deviates.

**Author(s)**

T. W. Yee and Kai Huang

**References**


**See Also**

`laplace`.

**Examples**

```r
loc <- 1; b <- 2
y <- rlaplace(n = 100, loc = loc, scale = b)
mean(y)  # sample mean
loc     # population mean
var(y)  # sample variance
2 * b^2 # population variance

## Not run: loc <- 0; b <- 1.5; x <- seq(-5, 5, by = 0.01)
plot(x, dlaplace(x, loc, b), type = "l", col = "blue", ylim = c(0,1),
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Purple are 5,10,...,95 percentiles", las = 1, ylab = "")
abline(h = 0, col = "blue", lty = 2)
lines(qlaplace(seq(0.05,0.95,by = 0.05), loc, b),
      dlaplace(qlaplace(seq(0.05, 0.95, by = 0.05), loc, b), loc, b),
      col = "purple", lty = 3, type = "h")
lines(x, plaplace(x, loc, b), type = "l", col = "orange")
abline(h = 0, lty = 2)
## End(Not run)

plaplace(qlaplace(seq(0.05, 0.95, by = 0.05), loc, b), loc, b)
```
**Description**

Generic function for the *latent variables* of a model.

**Usage**

```r
latvar(object, ...)  
lv(object, ...)
```

**Arguments**

- `object`: An object for which the extraction of latent variables is meaningful.
- `...`: Other arguments fed into the specific methods function of the model. Sometimes they are fed into the methods function for `coef`.

**Details**

Latent variables occur in reduced-rank regression models, as well as in quadratic and additive ordination models. For the latter two, latent variable values are often called *site scores* by ecologists. Latent variables are linear combinations of the explanatory variables.

**Value**

The value returned depends specifically on the methods function invoked.

**Warning**

`latvar` and `lv` are identical, but the latter will be deprecated soon.

Latent variables are not really applicable to `vglm/vgam` models.

**Author(s)**

Thomas W. Yee

**References**


See Also

latvar.qrrvglm, latvar.rrvglm, latvar.cao, lvplot.

Examples

```r
## Not run:
hs[ , 1:6] <- scale(hs[ , 1:6])  # Standardized environmental vars
set.seed(123)
pl <- cao(chind(Pdlugu, Pardmont, Pardnigr, Pardpull, Zorasp) ~
        WaterCon + BareSand + FallTwig + CoveMass + CoveHerb + ReflLux,
        family = poissonff, data = hs, Rank = 1, df1.nl =
        c(Zorasp = 2.5, 3), Bestof = 3, Crowpositive = TRUE)

var(latvar(pl))  # Scaled to unit variance  # Scaled to unit variance
c(latvar(pl))  # Estimated site scores

## Not run
```

leipnik  

## Leipnik Distribution Family Function

### Description

Estimates the two parameters of a (transformed) Leipnik distribution by maximum likelihood estimation.

### Usage

```r
leipnik(lmu = "logit", llambda = "loge", imu = NULL, ilambda = NULL)
```

### Arguments

- `lmu`, `llambda` Link function for the $\mu$ and $\lambda$ parameters. See `Links` for more choices.
- `imu`, `ilambda` Numeric. Optional initial values for $\mu$ and $\lambda$.

### Details

The (transformed) Leipnik distribution has density function

$$f(y; \mu, \lambda) = \frac{(y(1-y))^{-\frac{1}{2}}}{\text{Beta}(\frac{\lambda+1}{2}, \frac{1}{2})} \left[ 1 + \frac{(y-\mu)^2}{y(1-y)} \right]^{-\frac{1}{2}}$$

where $0 < y < 1$ and $\lambda > -1$. The mean is $\mu$ (returned as the fitted values) and the variance is $1/\lambda$.

Jörgensen (1997) calls the above the **transformed** Leipnik distribution, and if $y = (x + 1)/2$ and $\mu = (\theta + 1)/2$, then the distribution of $X$ as a function of $x$ and $\theta$ is known as the the (untransformed) Leipnik distribution. Here, both $x$ and $\theta$ are in $(-1, 1)$. 
Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

If llambda="identitylink" then it is possible that the lambda estimate becomes less than −1, i.e., out of bounds. One way to stop this is to choose llambda = "loge", however, lambda is then constrained to be positive.

Note

Convergence may be slow or fail. Until better initial value estimates are forthcoming try assigning the argument ilambda some numerical value if it fails to converge. Currently, Newton-Raphson is implemented, not Fisher scoring. Currently, this family function probably only really works for intercept-only models, i.e., y ~ 1 in the formula.

Author(s)

T. W. Yee

References


See Also

mccullagh89.

Examples

ldata <- data.frame(y = rnorm(n = 2000, mean = 0.5, sd = 0.1))  # Not proper data
fit <- vglm(y ~ 1, leipnik(llambda = 1), data = ldata, trace = TRUE, checkwz = FALSE)
fit <- vglm(y ~ 1, leipnik(llambda = 1, llambda = logoff(offset = 1)),
    data = ldata, trace = TRUE, crit = "coef")
head(fitted(fit))
with(ldata, mean(y))
summary(fit)
coef(fit, matrix = TRUE)
Coef(fit)

sum(weights(fit))  # Sum of the prior weights
sum(weights(fit, type = "work"))  # Sum of the working weights
lerch  

**lerch**  

* Lerch Phi Function*

**Description**
Computes the Lerch transcendental Phi function.

**Usage**
```plaintext
lerch(x, s, v, tolerance = 1.0e-10, iter = 100)
```

**Arguments**
- **x, s, v** Numeric. This function recycles values of x, s, and v if necessary.
- **tolerance** Numeric. Accuracy required, must be positive and less than 0.01.
- **iter** Maximum number of iterations allowed to obtain convergence. If iter is too small then a result of NA may occur; if so, try increasing its value.

**Details**
The Lerch transcendental function is defined by

$$
\Phi(x, s, v) = \sum_{n=0}^{\infty} \frac{x^n}{(n + v)^s}
$$

where |x| < 1 and v ≠ 0, −1, −2, ... Actually, x may be complex but this function only works for real x. The algorithm used is based on the relation

$$
\Phi(x, s, v) = x^m \Phi(x, s, v + m) + \sum_{n=0}^{m-1} \frac{x^n}{(n + v)^s}.
$$

See the URL below for more information. This function is a wrapper function for the C code described below.

**Value**
Returns the value of the function evaluated at the values of x, s, v. If the above ranges of x and v are not satisfied, or some numeric problems occur, then this function will return a NA for those values.

**Warning**
This function has not been thoroughly tested and contains bugs, for example, the zeta function cannot be computed with this function even though \( \zeta(s) = \Phi(x = 1, s, v = 1) \). There are many sources of problems such as lack of convergence, overflow and underflow, especially near singularities. If any problems occur then a NA will be returned.
Note

There are a number of special cases, e.g., the Riemann zeta-function is given by \( \zeta(s) = \Phi(x = 1, s, v = 1) \). The special case of \( s = 1 \) corresponds to the hypergeometric 2F1, and this is implemented in the gsl package. The Lerch transcendentonal Phi function should not be confused with the Lerch zeta function though they are quite similar.

Author(s)

S. V. Aksenov and U. D. Jentschura wrote the C code. The R wrapper function was written by T. W. Yee.

References


See Also

zeta.

Examples

```r
## Not run:
s <- 2; v <- 1; x <- seq(-1.1, 1.1, length = 201)
plot(x, lerch(x, s = s, v = v), type = "l", col = "blue", las = 1,
     main = paste("lerch(x, s = ", s, ", v = ", v, ")", sep = ")
abline(v = 0, h = 1, lty = "dashed", col = "gray")

s <- rnorm(n = 100)
max(abs(zeta(s) - lerch(x = 1, s = s, v = 1))) # This fails (a bug); should be 0

## End(Not run)
```

---

**leukemia**

*Acute Myelogenous Leukemia Survival Data*

Description

Survival in patients with Acute Myelogenous Leukemia

Usage

```r
data(leukemia)
```
Format

time: survival or censoring time
status: censoring status
x: maintenance chemotherapy given? (factor)

Note

This data set has been transferred from survival and renamed from aml to leukemia.

Source


levy

Levy Distribution Family Function

Description

Estimates the scale parameter of the Levy distribution by maximum likelihood estimation.

Usage

levy(location = 0, lscale = "loge", iscale = NULL)

Arguments

location Location parameter. Must have a known value. Called \( a \) below.
lscale Parameter link function for the (positive) scale parameter \( b \). See Links for more choices.
iscale Initial value for the \( b \) parameter. By default, an initial value is chosen internally.

Details

The Levy distribution is one of three stable distributions whose density function has a tractable form. The formula for the density is

\[
f(y; b) = \sqrt{\frac{b}{2\pi}} \exp\left(\frac{-b}{2(y-a)}\right) \frac{1}{(y-a)^{3/2}}
\]

where \( a < y < \infty \) and \( b > 0 \). Note that if \( a \) is very close to \( \min(y) \) (where \( y \) is the response), then numerical problem will occur. The mean does not exist. The median is returned as the fitted values.
Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Author(s)

T. W. Yee

References


See Also


Examples

```r
nn <- 1000; loc1 <- 0; loc2 <- 10
myscale <- 1  # log link == 0 is the answer
ldata <- data.frame(y1 = loc1 +myscale/rnorm(nn)^2,  # Levy(myscale, a)
y2 = rlevy(nn, loc = loc2, scale = exp(+2)))

# Cf. Table 1.1 of Nolan for Levy(1,0)
with(ldata, sum(y1 > 1) / length(y1))  # Should be 0.6827
with(ldata, sum(y1 > 2) / length(y1))  # Should be 0.5205

fit1 <- vglm(y1 ~ 1, levy(location = loc1), data = ldata, trace = TRUE)
coef(fit1, matrix = TRUE)
Coef(fit1)
summary(fit1)
head(weights(fit1, type = "work"))

fit2 <- vglm(y2 ~ 1, levy(location = loc2), data = ldata, trace = TRUE)
coef(fit2, matrix = TRUE)
Coef(fit2)
c(median = with(ldata, median(y2)), fitted.median = head(fitted(fit2), 1))
```

lgamma1

Log-gamma Distribution Family Function

Description

Estimation of the parameter of the standard and nonstandard log-gamma distribution.

Usage

```r
lgamma1(lshape = "loge", ishape = NULL)
lgamma3(llocation = "identitylink", lscale = "loge", lshape = "loge",
    illocation = NULL, iscale = NULL, ishape = 1, zero = 2:3)
```
Arguments

1location, 1scale
Parameter link function applied to the location parameter \( a \) and the positive scale parameter \( b \). See \texttt{Links} for more choices.

1shape
Parameter link function applied to the positive shape parameter \( k \). See \texttt{Links} for more choices.

ishape
Initial value for \( k \). If given, it must be positive. If failure to converge occurs, try some other value. The default means an initial value is determined internally.

ilocation, iscale
Initial value for \( a \) and \( b \). The defaults mean an initial value is determined internally for each.

zero
An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \{1,2,3\}. The default value means none are modelled as intercept-only terms. See \texttt{CommonVGAMffArguments} for more information.

Details

The probability density function of the standard log-gamma distribution is given by

\[
f(y; k) = \exp[ky - \exp(y)]/\Gamma(k),
\]
for parameter \( k > 0 \) and all real \( y \). The mean of \( Y \) is \texttt{digamma}(k) (returned as the fitted values) and its variance is \texttt{trigamma}(k).

For the non-standard log-gamma distribution, one replaces \( y \) by \((y - a)/b\), where \( a \) is the location parameter and \( b \) is the positive scale parameter. Then the density function is

\[
f(y) = \exp[k(y - a)/b - \exp((y - a)/b)]/(b\Gamma(k)).
\]

The mean and variance of \( Y \) are \( a + b\*\texttt{digamma}(k) \) (returned as the fitted values) and \( b^2 \* \texttt{trigamma}(k) \), respectively.

Value

An object of class "\texttt{vglmff}" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, and \texttt{vgam}.

Note

The standard log-gamma distribution can be viewed as a generalization of the standard type 1 extreme value density: when \( k = 1 \) the distribution of \(-Y\) is the standard type 1 extreme value distribution.

The standard log-gamma distribution is fitted with \texttt{lngamma1} and the non-standard (3-parameter) log-gamma distribution is fitted with \texttt{lngamma3}.

Author(s)

T. W. Yee
The Log-Gamma Distribution

Density, distribution function, quantile function and random generation for the log-gamma distribution with location parameter \( \text{location} \), scale parameter \( \text{scale} \) and shape parameter \( k \).

Usage

\[
\begin{align*}
\text{dlgamma}(x, \text{location} = 0, \text{scale} = 1, \text{shape} = 1, \log = \text{FALSE}) \\
\text{plgamma}(q, \text{location} = 0, \text{scale} = 1, \text{shape} = 1, \\
\quad \text{lower.tail} = \text{TRUE}, \log.p = \text{FALSE}) \\
\text{qlgamma}(p, \text{location} = 0, \text{scale} = 1, \text{shape} = 1, \\
\quad \text{lower.tail} = \text{TRUE}, \log.p = \text{FALSE}) \\
\text{rlgamma}(n, \text{location} = 0, \text{scale} = 1, \text{shape} = 1)
\end{align*}
\]

Arguments

\[
\begin{align*}
x, q & \quad \text{vector of quantiles.} \\
p & \quad \text{vector of probabilities.} \\
n & \quad \text{number of observations. Same as \text{runif}.} \\
\text{location} & \quad \text{the location parameter } a.
\end{align*}
\]

Examples

```r
ldata <- data.frame(y = rlgamma(100, shape = exp(1)))
fit <- vglm(y ~ 1, lgamma1, data = ldata, trace = TRUE, crit = "coef")
summary(fit)
coef(fit, matrix = TRUE)
Coef(fit)

ldata <- data.frame(x2 = runif(nn <- 5000))  # Another example
ldata <- transform(ldata, loc = -1 + 2 * x2, Scale = exp(1))
ldata <- transform(ldata, y = rlgamma(nn, loc, scale = Scale, shape = exp(0)))
fit2 <- vglm(y ~ x2, lgamma3, data = ldata, trace = TRUE, crit = "c")
coef(fit2, matrix = TRUE)
```
scale  the (positive) scale parameter \( b \).
shape  the (positive) shape parameter \( k \).
log   Logical. If \( \log = \text{TRUE} \) then the logarithm of the density is returned.
lower.tail, log.p
        Same meaning as in \texttt{pnorm} or \texttt{qnorm}.

Details

See \texttt{lgamma1}, the \texttt{VGAM} family function for estimating the one parameter standard log-gamma distribution by maximum likelihood estimation, for formulae and other details. Apart from \( n \), all the above arguments may be vectors and are recycled to the appropriate length if necessary.

Value

dlgamma gives the density, plgamma gives the distribution function, qlgamma gives the quantile function, and rlgamma generates random deviates.

Note

The \texttt{VGAM} family function \texttt{lgamma3} is for the three parameter (nonstandard) log-gamma distribution.

Author(s)

T. W. Yee and Kai Huang

References


See Also

\texttt{lgamma1, prentice74}.

Examples

```r
# Not run:  loc <- 1; Scale <- 1.5; shape <- 1.4
x <- seq(-3.2, 5, by = 0.01)
plot(x, dgamma(x, loc = loc, Scale, shape = shape), type = "l",
     col = "blue", ylim = 0:1,
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Percentile are 5, 10, ..., 95 percentiles", las = 1, ylab = "")
abline(h = 0, col = "blue", lty = 2)
lines(qgamma(seq(0.05, 0.95, by = 0.05), loc = loc, Scale, shape = shape),
     dgamma(qgamma(seq(0.05, 0.95, by = 0.05), loc = loc, scale = Scale, shape = shape),
     loc = loc, Scale, shape = shape), col = "purple", lty = 3, type = "h")
lines(x, plgamma(x, loc = loc, Scale, shape = shape), col = "orange")
abline(h = 0, lty = 2)
# End(Not run)
```
Description

Density, cumulative distribution function, and random generation for the Lindley distribution.

Usage

dlind(x, theta, log = FALSE)
plind(q, theta, lower.tail = TRUE, log.p = FALSE)
rlind(n, theta)

Arguments

x, q
vector of quantiles.
n
number of observations. Same as in runif.
log
Logical. If log = TRUE then the logarithm of the density is returned.
theta
positive parameter.
lower.tail, log.p
Same meaning as in pnorm or qnorm.

Details

See lindley for details.

Value

dlind gives the density, plind gives the cumulative distribution function, and rlind generates random deviates.

Author(s)

T. W. Yee and Kai Huang

See Also

lindley.

Examples

theta <- exp(-1); x <- seq(0.0, 17, length = 700)
dlind(0:10, theta)
## Not run:
plot(x, dlind(x, theta), type = "l", las = 1, col = "blue",
     main = "dlind(x, theta = exp(-1))")
abline(h = 1, col = "grey", lty = "dashed")
## End(Not run)
**Description**

Estimates the (1-parameter) Lindley distribution by maximum likelihood estimation.

**Usage**

```r
lindley(link = "loge", itheta = NULL, zero = NULL)
```

**Arguments**

- `link` Link function applied to the (positive) parameter. See `Links` for more choices.
- `itheta, zero` See `CommonVGAMffArguments` for information.

**Details**

The density function is given by

\[ f(y; \theta) = \theta^2(1 + y) \exp(-\theta y)/(1 + \theta) \]

for \( \theta > 0 \) and \( y > 0 \). The mean of \( Y \) (returned as the fitted values) is \( \mu = (\theta + 2)/(\theta(\theta + 1)) \).

The variance is \( (\theta^2 + 4\theta + 2)/(\theta(\theta + 1))^2 \).

**Value**

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

**Note**

This VGAM family function can handle multiple responses (inputted as a matrix). Fisher scoring is implemented.

**Author(s)**

T. W. Yee

**References**


See Also
dlind, gammaR, simulate.vlm.

Examples
ldata <- data.frame(y = rbind(n = 1000, theta = exp(3)))
fit <- vglm(y ~ 1, lindley, data = ldata, trace = TRUE, crit = "coef")
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)

linkfun

Link functions

Description
Generic function for returning the link functions of a fitted object.

Usage
linkfun(object, ...)

Arguments
object An object which has parameter link functions.
... Other arguments fed into the specific methods function of the model.

Details
Fitted models in the VGAM have parameter link functions. This generic function returns these.

Value
The value returned depends specifically on the methods function invoked.

Author(s)
Thomas W. Yee

See Also
linkfun.vglm, multilogit, vglm.
Examples

pneumo <- transform(pneumo, let = log(exposure.time))
fit1 <- vglm(cbind(normal, mild, severe) ~ let, propodds, data = pneumo)
coef(fit1, matrix = TRUE)
linkfun(fit1)
linkfun(fit1, earg = TRUE)

fit2 <- vglm(cbind(normal, mild, severe) ~ let, multinomial, data = pneumo)
coef(fit2, matrix = TRUE)
linkfun(fit2)
linkfun(fit2, earg = TRUE)

Description

Returns the link functions, and parameter names, for vector generalized linear models (VGLMs).

Usage

linkfun.vglm(object, earg = FALSE, ...)

Arguments

object Object of class "vglm", i.e., a VGLM object.
earg Logical. Return the extra arguments associated with each link function? If TRUE then a list is returned.
... Arguments that might be used in the future.

Details

All fitted VGLMs have a link function applied to each parameter. This function returns these, and optionally, the extra arguments associated with them.

Value

Usually just a (named) character string, with the link functions in order. It is named with the parameter names. If earg = TRUE then a list with the following components.

link The default output.
earg The extra arguments, in order.

Note

Presently, the multinomial logit model has only one link function, multilogit, so a warning is not issued for that link. For other models, if the number of link functions does not equal M then a warning may be issued.
Author(s)

Thomas W. Yee

See Also

`linkfun`, `multilogit`, `vglm`.

Examples

```r
fit1 <- vgam(cbind(r1, r2) ~ s(year, df = 3), gev(zero = 2:3), data = venice)
coef(fit1, matrix = TRUE)
linkfun(fit1)
linkfun(fit1, earg = TRUE)
```

Description

The VGAM package provides a number of (parameter) link functions which are described in general here. Collectively, they offer the user considerable flexibility for modelling data.

Usage

```r
TypicalVGAMlinkFunction(theta, someParameter = 0,
bvalue = NULL,
inverse = FALSE, deriv = 0,
short = TRUE, tag = FALSE)
```

Arguments

- `theta`: Numeric or character. Actually this can be $\theta$ (default) or $\eta$, depending on the other arguments. If `theta` is character then `inverse` and `deriv` are ignored. The name `theta` should always be the name of the first argument.
- `someParameter`: Some parameter, e.g., an offset.
- `bvalue`: Boundary value, positive if given. If $0 < \theta$ then values of `theta` which are less than or equal to 0 can be replaced by `bvalue` before computing the link function value. Values of `theta` which are greater than or equal to 1 can be replaced by 1 minus `bvalue` before computing the link function value. The value `bvalue = .Machine$double.eps` is sometimes a reasonable value, or something slightly higher.
- `inverse`: Logical. If `TRUE` the inverse link value $\theta$ is returned, hence the argument `theta` is really $\eta$.
- `deriv`: Integer. Either 0, 1, or 2 specifying the order of the derivative.
- `short, tag`: Logical. These are used for labelling the `blurb` slot of a `vglmff-class` object. These arguments are used only if `theta` is character, and gives the formula for the link in character form. If `tag = TRUE` then the result is preceded by a little more information.
Details

Almost all VGAM link functions have something similar to the argument list as given above. In this help file we have $\eta = g(\theta)$ where $g$ is the link function, $\theta$ is the parameter and $\eta$ is the linear/additive predictor.

The following is a brief enumeration of all VGAM link functions.

For parameters lying between 0 and 1 (e.g., probabilities): \texttt{logit}, \texttt{probit}, \texttt{cloglog}, \texttt{cauchit}, \texttt{foldsqrt}, \texttt{logc}, \texttt{golf}, \texttt{polf}, \texttt{nbolf}.

For positive parameters (i.e., greater than 0): \texttt{loge}, \texttt{negloge}, \texttt{powerlink}.

For parameters greater than 1: \texttt{loge}.

For parameters between $-1$ and 1: \texttt{fisherz}, \texttt{rhobit}.

For parameters between $A$ and $B$: \texttt{extlogit}, \texttt{logoff} ($B = \infty$).

For unrestricted parameters (i.e., any value): \texttt{identity}, \texttt{negidentity}, \texttt{reciprocal}, \texttt{negreciprocal}.

Value

Returns one of the link function value or its first or second derivative, the inverse link or its first or second derivative, or a character description of the link.

Here are the general details. If $\text{inverse} = \text{FALSE}$ and $\text{deriv} = 0$ (default) then the ordinary link function $\eta = g(\theta)$ is returned. If $\text{inverse} = \text{FALSE}$ and $\text{deriv} = 1$ then it is $d\theta/d\eta$ as a function of $\theta$. If $\text{inverse} = \text{FALSE}$ and $\text{deriv} = 2$ then it is $d^2\theta/d\eta^2$ as a function of $\theta$.

If $\text{inverse} = \text{TRUE}$ and $\text{deriv} = 0$ then the inverse link function is returned, hence theta is really $\eta$. If $\text{inverse} = \text{TRUE}$ and $\text{deriv}$ is positive then the \texttt{reciprocal} of the same link function with ($\text{theta} = \text{theta}$, \texttt{someParameter}, $\text{inverse} = \text{TRUE}$, $\text{deriv} = \text{deriv}$) is returned.

Note

VGAM link functions are generally not compatible with other functions outside the package. In particular, they won’t work with \texttt{glm} or any other package for fitting GAMs.

From October 2006 onwards, all VGAM family functions will only contain one default value for each link argument rather than giving a vector of choices. For example, rather than binomialff\{link = c("logit", "probit")\} it is now binomialff\{link = "logit", ...\} No checking will be done to see if the user’s choice is reasonable. This means that the user can write his/her own VGAM link function and use it within any VGAM family function. Altogether this provides greater flexibility. The downside is that the user must specify the full name of the link function, by either assigning the link argument the full name as a character string, or just the name itself. See the examples below.

From August 2012 onwards, a major change in link functions occurred. Argument \texttt{esigma} (and the like such as \texttt{earg}) used to be in VGAM prior to version 0.9-0 (released during the 2nd half of 2012). The major change is that arguments such as \texttt{offset} that used to be passed in via those arguments can done directly through the link function. For example, gev\{lshape = "logoff", eshape = list\{offset = 0.5\}\} is replaced by gev\{lshape = logoff\{offset = 0.5\}\}. The \@misc slot no longer has link and earg components, but two other components replace these. Functions such as \texttt{dtheta.deta()}, \texttt{d2theta.deta2()}, \texttt{eta2theta()}, \texttt{theta2eta()} are modified.

Author(s)

T. W. Yee
Links

References


See Also

`TypicalVGAMfamilyFunction`, `linkfun`, `vglm`, `vgam`, `rrvglm`, `cgo`, `cao`.

Examples

```r
logit("a")
logit("a", short = FALSE)
logit("a", short = FALSE, tag = TRUE)

logoff(1:5, offset = 1)  # Same as log(1:5 + 1)
powerlink(1:5, power = 2)  # Same as (1:5)^2

## Not run:  # This is old and no longer works:
logoff(1:5, earg = list(offset = 1))
powerlink(1:5, earg = list(power = 2))

## End(Not run)

fit1 <- vgam(agaus ~ altitude, binomialff(link = "cloglog"), hunua)  # okay
fit2 <- vgam(agaus ~ altitude, binomialff(link = "cloglog"), hunua)  # okay

## Not run:
# This no longer works since "clog" is not a valid VGAM link function:
fit3 <- vgam(agaus ~ altitude, binomialff(link = "clog"), hunua)  # not okay

# No matter what the link, the estimated var-cov matrix is the same
y <- rbeta(n = 1000, shape1 = exp(0), shape2 = exp(1))
fit1 <- vglm(y ~ 1, betaR(lshape1 = "identitylink", lshape2 = "identitylink"),
            trace = TRUE, crit = "coef")
fit2 <- vglm(y ~ 1, betaR(lshape1 = logoff(offset = 1.1),
            lshape2 = logoff(offset = 1.1)),
            trace = TRUE, crit = "coef")
vcov(fit1, untransform = TRUE)
vcov(fit1, untransform = TRUE) - vcov(fit2, untransform = TRUE)  # Should be all 0s

# This is old:
fit1@misc$earg  # Some 'special' parameters
fit2@misc$earg  # Some 'special' parameters are here
}
```

```r
par(mfrow = c(2, 2))
p <- seq(0.01, 0.99, len = 200)
x <- seq(-4, 4, len = 200)
plot(p, logit(p), type = "l", col = "blue")
plot(x, logit(x, inverse = TRUE), type = "l", col = "blue")
plot(p, logit(p, deriv = 1), type = "l", col = "blue")  # reciprocal!
```
The Generalized Beta Distribution (Libby and Novick, 1982)

Description

Density, distribution function, quantile function and random generation for the generalized beta distribution, as proposed by Libby and Novick (1982).

Usage

dlino(x, shape1, shape2, lambda = 1, log = FALSE)
plino(q, shape1, shape2, lambda = 1, lower.tail = TRUE, log.p = FALSE)
qlino(p, shape1, shape2, lambda = 1, lower.tail = TRUE, log.p = FALSE)
rlino(n, shape1, shape2, lambda = 1)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. Same as in runif.
shape1, shape2, lambda
    see lino.
log Logical. If log = TRUE then the logarithm of the density is returned.
lower.tail, log.p
    Same meaning as in pnorm or qnorm.

Details

See lino, the VGAM family function for estimating the parameters, for the formula of the probability density function and other details.

Value

dlino gives the density, plino gives the distribution function, qlino gives the quantile function, and rlino generates random deviates.

Author(s)

T. W. Yee and Kai Huang

See Also

lino.
Examples

```r
## Not run:
lambda <- 0.4; shape1 <- exp(1.3); shape2 <- exp(1.3)
x <- seq(0.0, 1.0, len = 101)
plot(x, dlin(x, shape1 = shape1, shape2 = shape2, lambda = lambda),
     type = "l", col = "blue", las = 1, ylab = "",
     main = "Blue is density, red is cumulative distribution function",
     sub = "Purple lines are the 10, 20, ..., 90 percentiles")
abline(h = 0, col = "blue", lty = 2)
lines(x, plin(x, shape1 = shape1, shape2 = shape2, l = lambda), col = "red")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qlin(probs, shape1 = shape1, shape2 = shape2, lambda = lambda)
lines(Q, dlin(Q, shape1 = shape1, shape2 = shape2, lambda = lambda),
     col = "purple", lty = 3, type = "h")
plin(Q, shape1 = shape1, shape2 = shape2, l = lambda) - probs  # Should be all 0

## End(Not run)
```

---

### Generalized Beta Distribution Family Function

**Description**


**Usage**

```r
lino(lshape1 = "loge", lshape2 = "loge", 1lambda = "loge",
     ishape1 = NULL, ishape2 = NULL, ilambda = 1, zero = NULL)
```

**Arguments**

- `lshape1, lshape2`
  Parameter link functions applied to the two (positive) shape parameters \(a\) and \(b\). See *Links* for more choices.

- `l1lambda`
  Parameter link function applied to the parameter \(\lambda\). See *Links* for more choices.

- `ishape1, ishape2, ilambda`
  Initial values for the parameters. A NULL value means one is computed internally. The argument `ilambda` must be numeric, and the default corresponds to a standard beta distribution.

- `zero`
  An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. Here, the values must be from the set \{1,2,3\} which correspond to \(a, b, \lambda\), respectively.
Details

Proposed by Libby and Novick (1982), this distribution has density

\[
f(y; a, b, \lambda) = \frac{\lambda^a y^{a-1} (1 - y)^{b-1}}{B(a, b) (1 - (1 - \lambda)y)^{a+b}}
\]

for \(a > 0, b > 0, \lambda > 0, 0 < y < 1\). Here \(B\) is the beta function (see \texttt{beta})\. The mean is a complicated function involving the Gauss hypergeometric function. If \(X\) has a \texttt{lino} distribution with parameters \(\text{shape1}, \text{shape2}, \lambda\)\texttt{ambda}, then \(Y = \lambda X / (1 - (1 - \lambda)X)\) has a standard beta distribution with parameters \(\text{shape1}, \text{shape2}\).

Since \(\log(\lambda) = 0\) corresponds to the standard beta distribution, a \texttt{summary} of the fitted model performs a t-test for whether the data belongs to a standard beta distribution (provided the \texttt{loge} link for \(\lambda\) is used; this is the default).

Value

An object of class “\texttt{vglmff}” (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, and \texttt{vgam}.

Note

The fitted values, which is usually the mean, have not been implemented yet. Currently the median is returned as the fitted values.

Although Fisher scoring is used, the working weight matrices are positive-definite only in a certain region of the parameter space. Problems with this indicate poor initial values or an ill-conditioned model or insufficient data etc.

This model is can be difficult to fit. A reasonably good value of \(\lambda\) seems to be needed so if the self-starting initial values fail, try experimenting with the initial value arguments. Experience suggests \(\lambda\) is better a little larger, rather than smaller, compared to the true value.

Author(s)

T. W. Yee

References


See Also

\texttt{lino, genbetaII}. 
Examples

```r
ldata <- data.frame(y1 = rbeta(n = 1000, exp(0.5), exp(1)))  # standard beta
fit <- vglm(y1 ~ 1, lino, data = ldata, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
head(fitted(fit))
summary(fit)

# Nonstandard beta distribution
ldata <- transform(ldata, y2 = rlnino(n = 1000, shape1 = exp(1),
    shape2 = exp(2), lambda = exp(1)))
fit2 <- vglm(y2 ~ 1, lino(lshape1 = "identitylink", lshape2 = "identitylink",
    ilamb = 10), data = ldata, trace = TRUE)
coef(fit2, matrix = TRUE)
```

lirat

*Low-iron Rat Teratology Data*

Description

Low-iron rat teratology data.

Usage

data(lirat)

Format

A data frame with 58 observations on the following 4 variables.

N Litter size.
R Number of dead fetuses.
hb Hemoglobin level.
grp Group number. Group 1 is the untreated (low-iron) group, group 2 received injections on day 7 or day 10 only, group 3 received injections on days 0 and 7, and group 4 received injections weekly.

Details

The following description comes from Moore and Tsiatis (1991). The data comes from the experimental setup from Shepard et al. (1980), which is typical of studies of the effects of chemical agents or dietary regimens on fetal development in laboratory rats.

Female rats were put in iron-deficient diets and divided into 4 groups. One group of controls was given weekly injections of iron supplement to bring their iron intake to normal levels, while another group was given only placebo injections. Two other groups were given fewer iron-supplement injections than the controls. The rats were made pregnant, sacrificed 3 weeks later, and the total number of fetuses and the number of dead fetuses in each litter were counted.
For each litter the number of dead fetuses may be considered to be Binomial($N, p$) where $N$ is the litter size and $p$ is the probability of a fetus dying. The parameter $p$ is expected to vary from litter to litter, therefore the total variance of the proportions will be greater than that predicted by a binomial model, even when the covariates for hemoglobin level and experimental group are accounted for.

**Source**


**References**


**Examples**

```r
## Not run:
# cf. Figure 3 of Moore and Tsiatis (1991)
plot(R / N ~ hb, data = lirat, pch = as.character(grp), col = grp,
     las = 1, xlab = "Hemoglobin level", ylab = "Proportion Dead")
## End(Not run)
```

---

**lms.bc**

**LMS Quantile Regression with a Box-Cox transformation to a Gamma Distribution**

**Description**

LMS quantile regression with the Box-Cox transformation to the gamma distribution.

**Usage**

```r
lms.bc(percentiles = c(25, 50, 75), zero = c(1, 3),
       llambda = "identitylink", lmu = "identitylink", lsigma = "loge",
       idf.mu = 4, idf.sigma = 2, ilambda = 1, isigma = NULL)
```

**Arguments**

- `percentiles` A numerical vector containing values between 0 and 100, which are the quantiles. They will be returned as ‘fitted values’.
- `zero` See `lms.bcn`.
- `llambda, lmu, lsigma` See `lms.bcn`.
- `idf.mu, idf.sigma` See `lms.bcn`.
- `ilambda, isigma` See `lms.bcn`.
Details
Given a value of the covariate, this function applies a Box-Cox transformation to the response to best obtain a gamma distribution. The parameters chosen to do this are estimated by maximum likelihood or penalized maximum likelihood. Similar details can be found at `lms.bc`.

Value
An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Warning
This VGAM family function comes with the same warnings as `lms.bc`. Also, the expected value of the second derivative with respect to lambda may be incorrect (my calculations do not agree with the Lopatatzidis and Green manuscript.)

Note
Similar notes can be found at `lms.bc`.

Author(s)
Thomas W. Yee

References
Lopatatzidis A. and Green, P. J. (unpublished manuscript) Semiparametric quantile regression using the gamma distribution.

See Also
`lms.bc`, `lms.yjn`, `qtplot.lmsreg`, `deplot.lmsreg`, `cdf.lmsreg`, `bmi.nz`, `amlexponential`.

Examples
```r
# This converges, but deplot(fit) and qtplot(fit) do not work
fit0 <- vglm(BMI ~ sm.bs(age, df = 4), lms.bc, data = bmi.nz, trace = TRUE)
coef(fit0, matrix = TRUE)
## Not run:
par(mfrow = c(1, 1))
plotvgam(fit0, se = TRUE) # Plot mu function (only)
## End(Not run)

# Use a trick: fit0 is used for initial values for fit1.
fit1 <- vgam(BMI ~ s(age, df = c(4, 2)), etastart = predict(fit0),
  lms.bc(zero = 1), bmi.nz, trace = TRUE)
```
LMS quantile regression with the Box-Cox transformation to normality.

**Description**

LMS quantile regression with the Box-Cox transformation to normality.

**Usage**

```r
lms.bcn(percentiles = c(25, 50, 75), zero = c(1, 3),
llambda = "identitylink", lmu = "identitylink", lsigma = "loge",
idf.mu = 4, idf.sigma = 2, ilambda = 1,
isigma = NULL, tol0 = 0.001)
```

**Arguments**

- **percentiles**: A numerical vector containing values between 0 and 100, which are the quantiles. They will be returned as ‘fitted values’.
zero

An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \{1,2,3\}. The default value usually increases the chance of successful convergence. Setting `zero = NULL` means they all are functions of the covariates. For more information see `CommonVGAMffArguments`.

llambda, lmu, lsigma

Parameter link functions applied to the first, second and third linear/additive predictors. See `Links` for more choices, and `CommonVGAMffArguments`.

idf.mu

Degrees of freedom for the cubic smoothing spline fit applied to get an initial estimate of mu. See `vsmooth.spline`.

idf.sigma

Degrees of freedom for the cubic smoothing spline fit applied to get an initial estimate of sigma. See `vsmooth.spline`. This argument may be assigned `NULL` to get an initial value using some other algorithm.

ilambda

Initial value for lambda. If necessary, it is recycled to be a vector of length \(n\) where \(n\) is the number of (independent) observations.

isigma

Optional initial value for sigma. If necessary, it is recycled to be a vector of length \(n\). The default value, `NULL`, means an initial value is computed in the `@initialize` slot of the family function.

tol0

Small positive number, the tolerance for testing if lambda is equal to zero.

Details

Given a value of the covariate, this function applies a Box-Cox transformation to the response to best obtain normality. The parameters chosen to do this are estimated by maximum likelihood or penalized maximum likelihood.

In more detail, the basic idea behind this method is that, for a fixed value of \(x\), a Box-Cox transformation of the response \(Y\) is applied to obtain standard normality. The 3 parameters (\(\lambda, \mu, \sigma\), which start with the letters “L-M-S” respectively, hence its name) are chosen to maximize a penalized log-likelihood (with `vgam`). Then the appropriate quantiles of the standard normal distribution are back-transformed onto the original scale to get the desired quantiles. The three parameters may vary as a smooth function of \(x\).

The Box-Cox power transformation here of the \(Y\), given \(x\), is

\[
Z = [(Y/\mu(x))^{\lambda(x)} - 1]/(\sigma(x) \lambda(x))
\]

for \(\lambda(x) \neq 0\). (The singularity at \(\lambda(x) = 0\) is handled by a simple function involving a logarithm.) Then \(Z\) is assumed to have a standard normal distribution. The parameter \(\sigma(x)\) must be positive, therefore `VGAM` chooses \(\eta(x)^T = (\lambda(x), \mu(x), \log(\sigma(x)))\) by default. The parameter \(\mu\) is also positive, but while \(\log(\mu)\) is available, it is not the default because \(\mu\) is more directly interpretable.

Given the estimated linear/additive predictors, the 100\(\alpha\) percentile can be estimated by inverting the Box-Cox power transformation at the 100\(\alpha\) percentile of the standard normal distribution.

Of the three functions, it is often a good idea to allow \(\mu(x)\) to be more flexible because the functions \(\lambda(x)\) and \(\sigma(x)\) usually vary more smoothly with \(x\). This is somewhat reflected in the default value for the argument `zero`, viz. `zero = c(1,3)`.
Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

The computations are not simple, therefore convergence may fail. Set trace = TRUE to monitor convergence if it isn't set already. Convergence failure will occur if, e.g., the response is bimodal at any particular value of x. In case of convergence failure, try different starting values. Also, the estimate may diverge quickly near the solution, in which case try prematurely stopping the iterations by assigning maxits to be the iteration number corresponding to the highest likelihood value.

One trick is to fit a simple model and use it to provide initial values for a more complex model; see in the examples below.

Note

The response must be positive because the Box-Cox transformation cannot handle negative values. The LMS-Yeo-Johnson-normal method can handle both positive and negative values.

In general, the lambda and sigma functions should be more smoother than the mean function. Having zero = 1, zero = 3 or zero = c(1, 3) is often a good idea. See the example below.

Author(s)

Thomas W. Yee

References


See Also

lms.bcg, lms.yjn, qtplot.lmscreg, deplot.lmscreg, cdf.lmscreg, alaplace1, amlnormal, denorm, CommonVGAMffArguments.

Examples

```r
## Not run: require("VGAMdata")
mysubset <- subset(xs.nz, sex == "M" & ethnicity == "Maori" & study1)
mysubset <- transform(mysubset, BMI = weight / height^2)
BMIdata <- na.omit(mysubset)
BMIdata <- subset(BMIdata, BMI < 80 & age < 65,
                   select = c(age, BMI)) # Delete an outlier
summary(BMIdata)
```
fit <- vgam(BMI ~ s(age, df = c(4, 2)), lms.bcn(zero = 1), data = BMIdata)

par(mfrow = c(1, 2))
plot(fit, scol = "blue", se = TRUE)  # The two centered smooths

head(predict(fit))
head(fitted(fit))

head(cdf(fit))  # Person 46 is probably overweight, given his age
100 * colMeans(depvar(fit, drop = TRUE) < fitted(fit))  # Empirical proportions

# Convergence problems? Try this trick: fit0 is a simpler model used for fit1
fit0 <- vgam(BMI ~ s(age, df = 4), lms.bcn(zero = c(1, 3)), data = BMIdata)
fit1 <- vgam(BMI ~ s(age, df = c(4, 2)), lms.bcn(zero = 1), data = BMIdata, etastart = predict(fit0))

## End(Not run)

## Not run:
# Quantile plot
par(bty = "1", mar = c(5, 4, 4, 3) + 0.1, xpd = TRUE)
qtplot(fit, percentiles = c(5, 50, 90, 99), main = "Quantiles",
       xlim = c(15, 66), las = 1, ylab = "BMI", lwd = 2, lcol = 4)

# Density plot
ygrid <- seq(15, 43, len = 100)  # BMI ranges
par(mfrow = c(1, 1), lwd = 2)
(aa <- deplot(fit, x0 = 20, y = ygrid, xlab = "BMI", col = "black",
          main = "Density functions at Age = 20 (black), 42 (red) and 55 (blue)
          (aa <- deplot(fit, x0 = 42, y = ygrid, add = TRUE, lty = 2, col = "red"
          aa <- deplot(fit, x0 = 55, y = ygrid, add = TRUE, lty = 4, col = "blue",
          Attach = TRUE)

aa@post$deplot  # Contains density function values

## End(Not run)

---

**lms.yjn**

LMS Quantile Regression with a Yeo-Johnson Transformation to Normality

---

**Description**

LMS quantile regression with the Yeo-Johnson transformation to normality.

**Usage**

lms.yjn(percentiles = c(25, 50, 75), zero = c(1,3),
        llambda = "identitylink", lsigma = "loge",
        idf.mu = 4, idf.sigma = 2,
        ilambda = 1, isigma = NULL, rule = c(10, 5),
        ...)
yoffset = NULL, diagW = FALSE, iters.diagW = 6)

lms.yjn2(percentiles=c(25,50,75), zero=c(1,3),
    llambda = "identitylink", lmu = "identitylink", lsigma = "loge",
    idf.mu = 4, idf.sigma = 2, ilambda = 1.0,
    isigma = NULL, yoffset = NULL, nsimEIM = 250)

Arguments

percentiles A numerical vector containing values between 0 and 100, which are the quantiles. They will be returned as ‘fitted values’.

zero See lms.bc.

llambda, lmu, lsigma See lms.bc.

idf.mu, idf.sigma See lms.bc.

ilambda, isigma See lms.bc.

rule Number of abscissae used in the Gaussian integration scheme to work out elements of the weight matrices. The values given are the possible choices, with the first value being the default. The larger the value, the more accurate the approximation is likely to be but involving more computational expense.

yoffset A value to be added to the response y, for the purpose of centering the response before fitting the model to the data. The default value, NULL, means -median(y) is used, so that the response actually used has median zero. The yoffset is saved on the object and used during prediction.

diagW Logical. This argument is offered because the expected information matrix may not be positive-definite. Using the diagonal elements of this matrix results in a higher chance of it being positive-definite, however convergence will be very slow. If TRUE, then the first iters.diagW iterations will use the diagonal of the expected information matrix. The default is FALSE, meaning faster convergence.

iters.diagW Integer. Number of iterations in which the diagonal elements of the expected information matrix are used. Only used if diagW = TRUE.

nsimEIM See CommonVGAMffArguments for more information.

Details

Given a value of the covariate, this function applies a Yeo-Johnson transformation to the response to best obtain normality. The parameters chosen to do this are estimated by maximum likelihood or penalized maximum likelihood. The function lms.yjn2() estimates the expected information matrices using simulation (and is consequently slower) while lms.yjn() uses numerical integration. Try the other if one function fails.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.
Warning

The computations are not simple, therefore convergence may fail. In that case, try different starting values.

The generic function `predict`, when applied to a `lms.yjn` fit, does not add back the `yoffset` value.

Note

The response may contain both positive and negative values. In contrast, the LMS-Box-Cox-normal and LMS-Box-Cox-gamma methods only handle a positive response because the Box-Cox transformation cannot handle negative values.

Some other notes can be found at `lms.bcn`.

Author(s)

Thomas W. Yee

References


See Also

`lms.bcn`, `lms.bcg`, `qtplot.lmscreg`, `deplot.lmscreg`, `cdf.lmscreg`, `bmi.nz`, `amlnormal`.

Examples

```r
fit <- vgam(BMI ~ s(age, df = 4), lms.yjn, bmi.nz, trace = TRUE)
head(predict(fit))
head(fitted(fit))
head(bmi.nz)
# Person 1 is near the lower quartile of BMI amongst people his age
head(cdf(fit))

## Not run:
# Quantile plot
par(bty = "l", mar = c(5, 4, 4, 3) + 0.1, xpd = TRUE)
qtplot(fit, percentiles = c(5, 50, 90, 99), main = "Quantiles",
xlim = c(15, 90), las = 1, ylab = "BMI", lwd = 2, lcol = 4)

# Density plot
ygrid <- seq(15, 43, len = 100) # BMI ranges
par(mfrow = c(1, 1), lwd = 2)
(aa <- deplot(fit, x0 = 20, y = ygrid, xlab = "BMI", col = "black”,
```

main = "Density functions at Age = 20 (black), 42 (red) and 55 (blue)"

aa <- deplot(fit, x0 = 42, y = ygrid, add = TRUE, lty = 2, col = "red")

aa <- deplot(fit, x0 = 55, y = ygrid, add = TRUE, lty = 4, col = "blue",
            attach = TRUE)

with(aa@post, deplot)  # Contains density function values; == a@post$deplot

## End(Not run)

---

**Log**

---

### Logarithmic Distribution

**Description**

Density, distribution function, and random generation for the logarithmic distribution.

**Usage**

\[
dlog(x, \text{prob}, \text{log} = \text{FALSE})
\]

\[
plog(q, \text{prob}, \text{log.p} = \text{FALSE})
\]

\[
rlog(n, \text{prob}, \text{Smallno} = 1.0e-6)
\]

**Arguments**

- **x, q**
  Vector of quantiles. For the density, it should be a vector with positive integer values in order for the probabilities to be positive.

- **n**
  number of observations. Same as in `runif`.

- **prob**
  The parameter value \(c\) described in in `logff`. Here it is called `prob` because \(0 < c < 1\) is the range. For `rlog()` this parameter must be of length 1.

- **log, log.p**
  Logical. If `log.p = TRUE` then all probabilities \(p\) are given as `log(p)`.

- **Smallno**
  Numeric, a small value used by the rejection method for determining the upper limit of the distribution. That is, `plog(U, prob) > 1-Smallno` where \(U\) is the upper limit.

**Details**

The details are given in `logff`.

**Value**

`dlog` gives the density, `plog` gives the distribution function, and `rlog` generates random deviates.

**Note**

Given some response data, the `VGAM` family function `logff` estimates the parameter `prob`. For `plog()`, if argument `q` contains large values and/or `q` is long in length then the memory requirements may be very high. Very large values in `q` are handled by an approximation by Owen (1965).
log1pexp

Author(s)
T. W. Yee

References

See Also
logff.

Examples

dlog(1:20, 0.5)
rlog(20, 0.5)

## Not run: prob <- 0.8; x <- 1:10
plot(x, dlog(x, prob = prob), type = "h", ylim = 0:1,
     sub = "prob=0.8", las = 1, col = "blue", ylab = "Probability",
     main = "Logarithmic distribution: blue=density; orange=distribution function")
lines(x + 0.1, plog(x, prob = prob), col = "orange", lty = 3, type = "h")
## End(Not run)

log1pexp Logarithms with an Unit Offset and Exponential Term

Description
Computes \( \log(1 + \exp(x)) \) accurately.

Usage
log1pexp(x)

Arguments
x A vector of reals (numeric). Complex numbers not allowed since log1p does not handle these.

Details
Computes \( \log(1 + \exp(x)) \) accurately. An adjustment is made when \( x \) is positive and large in value.

Value
Returns \( \log(1 + \exp(x)) \).
logc

**Complementary-log Link Function**

**Description**
Computes the complementary-log transformation, including its inverse and the first two derivatives.

**Usage**

\[
\text{logc}(\text{theta}, \text{bvalue} = \text{NULL}, \text{inverse} = \text{FALSE}, \text{deriv} = 0, \\
\text{short} = \text{TRUE}, \text{tag} = \text{FALSE})
\]

**Arguments**
- **theta**: Numeric or character. See below for further details.
- **bvalue**: See Links.
- **inverse, deriv, short, tag**: Details at Links.

**Details**
The complementary-log link function is suitable for parameters that are less than unity. Numerical values of theta close to 1 or out of range result in Inf, -Inf, NA or NaN.

**Value**
- For \( \text{deriv} = 0 \), the log of \( \text{theta} \), i.e., \( \log(1-\text{theta}) \) when \( \text{inverse} = \text{FALSE} \), and if \( \text{inverse} = \text{TRUE} \) then \( 1-\exp(\text{theta}) \).
- For \( \text{deriv} = 1 \), then the function returns \( d\theta / d\eta \) as a function of \( \text{theta} \) if \( \text{inverse} = \text{FALSE} \), else if \( \text{inverse} = \text{TRUE} \) then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base \( e \).

**Note**
Numerical instability may occur when \( \text{theta} \) is close to 1. One way of overcoming this is to use \( \text{bvalue} \).

---

**See Also**
- `log1p, exp`.

**Examples**

```
x <- c(10, 50, 100, 200, 400, 500, 800, 1000, 1e4, 1e5, 1e20, Inf)
log1pexp(x)
log(1 + exp(x))  # Naive; suffers from overflow
x <- -c(10, 50, 100, 200, 400, 500, 800, 1000, 1e4, 1e5, 1e20, Inf)
log1pexp(x)
log(1 + exp(x))  # Naive; suffers from inaccuracy
```
**Author(s)**

Thomas W. Yee

**References**


**See Also**

`links, loge, cloglog, loglog, logoff`.

**Examples**

```r
## Not run:
logc(seq(-0.2, 1.1, by = 0.1)) # Has NAs

## End(Not run)
logc(seq(-0.2, 1.1, by = 0.1), bvalue = 1 - .Machine$double.eps) # Has no NAs
```

---

**loge**

*Log link function, and variants*

**Description**

Computes the log transformation, including its inverse and the first two derivatives.

**Usage**

```r
loge(theta, bvalue = NULL, inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)
negloge(theta, bvalue = NULL, inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)
logneg(theta, bvalue = NULL, inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)
```

**Arguments**

- `theta` Numeric or character. See below for further details.
- `bvalue` See `Links`.
- `inverse`, `deriv`, `short`, `tag`
  Details at `Links`.  

---
Details

The log link function is very commonly used for parameters that are positive. Here, all logarithms
are natural logarithms, i.e., to base $e$. Numerical values of theta close to 0 or out of range result in
Inf, -Inf, NA or NaN.

The function loge computes $\log(\theta)$ whereas negloge computes $-\log(\theta) = \log(1/\theta)$.

The function logneg computes $\log(-\theta)$, hence is suitable for parameters that are negative, e.g., a
trap-shy effect in posbernoulli.b.

Value

The following concerns loge. For deriv = 0, the log of theta, i.e., $\log(\theta)$ when inverse = FALSE,
and if inverse = TRUE then $\exp(\theta)$. For deriv = 1, then the function returns $d \theta / \eta$ as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the
reciprocal.

Note

This function is called loge to avoid conflict with the log function.

Numerical instability may occur when theta is close to 0 unless bvalue is used.

Author(s)

Thomas W. Yee

References

Hall.

See Also

Links, explink, logit, logc, loglog, log, logoff, lambertW, posbernoulli.b.

Examples

```r
## Not run:  loge(seq(-0.2, 0.5, by = 0.1))
loge(seq(-0.2, 0.5, by = 0.1), bvalue = .Machine$double.xmin)
negloge(seq(-0.2, 0.5, by = 0.1))
negloge(seq(-0.2, 0.5, by = 0.1), bvalue = .Machine$double.xmin)
## End(Not run)
logneg(seq(-0.5, -0.2, by = 0.1))
```
**logF**

*Natural Exponential Family Generalized Hyperbolic Secant Distribution Family Function*

**Description**

Maximum likelihood estimation of the 1-parameter log F distribution.

**Usage**

```r
call = TRUE

logf(lshape1 = "loge", lshape2 = "loge",
     ishape1 = NULL, ishape2 = 1, imethod = 1)
```

**Arguments**

- `lshape1, lshape2`
  Parameter link functions for the shape parameters. Called $\alpha$ and $\beta$ respectively. See **Links** for more choices.
- `ishape1, ishape2`
  Optional initial values for the shape parameters. If given, it must be numeric and values are recycled to the appropriate length. The default is to choose the value internally. See **CommonVGLMfArguments** for more information.
- `imethod`
  Initialization method. Either the value 1, 2, or .... See **CommonVGLMfArguments** for more information.

**Details**

The density for this distribution is

\[
f(y; \alpha, \beta) = \exp(\alpha y) / \left[ B(\alpha, \beta)(1 + e^y)^{\alpha+\beta} \right]
\]

where $y$ is real, $\alpha > 0$, $\beta > 0$, $B(.,.)$ is the beta function **beta**.

**Value**

An object of class "vglmff" (see **vglmff-class**). The object is used by modelling functions such as **vglm** and **vgam**.

**Author(s)**

Thomas W. Yee

**References**

See Also

dlogf, logff.

Examples

```r
nn <- 1000
ldata <- data.frame(y1 = rnorm(nn, m = +1, sd = exp(2)),  # Not proper data
                    x2 = rnorm(nn, m = -1, sd = exp(2)),
                    y2 = rnorm(nn, m = -1, sd = exp(2)))  # Not proper data
fit1 <- vglm(y1 ~ 1 , logF, data = ldata, trace = TRUE)
fit2 <- vglm(y2 ~ x2, logF, data = ldata, trace = TRUE)
coef(fit2, matrix = TRUE)
summary(fit2)
vcov(fit2)
head(fitted(fit1))
with(ldata, mean(y1))
max(abs(head(fitted(fit1)) - with(ldata, mean(y1))))
```

---

**logff**

*Logarithmic Distribution*

Description

Estimating the (single) parameter of the logarithmic distribution.

Usage

```r
logff(link = "logit", init.c = NULL, zero = NULL)
```

Arguments

- `link`  Parameter link function for the parameter $c$, which lies between 0 and 1. See `Links` for more choices and information.
- `init.c`  Optional initial value for the $c$ parameter. If given, it often pays to start with a larger value, e.g., 0.95. The default is to choose an initial value internally.
- `zero`  Details at `CommonVGAMffArguments`.

Details

The logarithmic distribution is a generalized power series distribution that is based specifically on the logarithmic series (scaled to a probability function). Its probability function is $f(y) = ac^y / y$, for $y = 1, 2, 3, \ldots$, where $0 < c < 1$, and $a = -1 / \log(1 - c)$. The mean is $ac / (1 - c)$ (returned as the fitted values) and variance is $ac(1 - ac) / (1 - c)^2$.

Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.
Note

The function \texttt{log} computes the natural logarithm. In the \texttt{VGAM} library, a link function with option \texttt{loge} corresponds to this.

Multiple responses are permitted.

The logarithmic distribution is sometimes confused with the log-series distribution. The latter was used by Fisher et al. for species abundance data and has two parameters.

Author(s)

T. W. Yee

References


See Also

\texttt{rlog, log, loge, logoff, explogff, simulate.vlm}.

Examples

```r
ldata <- data.frame(y = rlog(n = 1000, prob = logit(0.2, inverse = TRUE)))
fit <- vglm(y ~ 1, logff, data = ldata, trace = TRUE, crit = "c")
coef(fit, matrix = TRUE)
Coef(fit)
## Not run: with(ldata,
  hist(y, prob = TRUE, breaks = seq(0.5, max(y) + 0.5, by = 1),
       border = "blue")
x <- seq(1, with(ldata, max(y)), by = 1)
with(ldata, lines(x, dlog(x, Coef(fit)[1]), col = "orange", type = "h", lwd = 2))
## End(Not run)

# Example: Corbet (1943) butterfly Malaya data
corbet <- data.frame(nindiv = 1:24,
  ofreq = c(118, 74, 44, 24, 29, 22, 20, 19, 20, 15, 12,
            14, 6, 12, 6, 9, 9, 6, 10, 10, 11, 5, 3, 3))
fit <- vglm(nindiv ~ 1, logff, data = corbet, weights = ofreq)
coef(fit, matrix = TRUE)
chat <- Coef(fit)["c"]
pdf2 <- dlog(x = with(corbet, nindiv), prob = chat)
print(with(corbet, cbind(nindiv, ofreq, fitted = pdf2 * sum(ofreq))), digits = 1)
```
logistic

Logistic Distribution Family Function

Description

Estimates the location and scale parameters of the logistic distribution by maximum likelihood estimation.

Usage

logistic(llocation = "identitylink", scale.arg = 1, imethod = 1)
logistic(llocation = "identitylink", lscale = "loge",
   llocation = NULL, iscale = NULL, imethod = 1, zero = -2)

Arguments

llocation, lscale
Parameter link functions applied to the location parameter \( l \) and scale parameter \( s \). See Links for more choices, and CommonVGAMffArguments for more information.
scale.arg
Known positive scale parameter (called \( s \) below).
ilocation, iscale
See CommonVGAMffArguments for more information.
imethod, zero
See CommonVGAMffArguments for more information.

Details

The two-parameter logistic distribution has a density that can be written as

\[
f(y; l, s) = \frac{\exp[-(y - l)/s]}{s (1 + \exp[-(y - l)/s])^2}
\]

where \( s > 0 \) is the scale parameter, and \( l \) is the location parameter. The response \(-\infty < y < \infty\). The mean of \( Y \) (which is the fitted value) is \( l \) and its variance is \( \pi^2 s^2 / 3 \).

A logistic distribution with scale = 0.65 (see dlogis) resembles dt with df = 7; see logistic1 and studentt.

logistic1 estimates the location parameter only while logistic estimates both parameters. By default, \( \eta_1 = l \) and \( \eta_2 = \log(s) \) for logistic.

logistic can handle multiple responses.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.
logit

Note

Fisher scoring is used, and the Fisher information matrix is diagonal.

Author(s)

T. W. Yee

References


See Also

rlogis, logit, cumulative, bilogistic, simulate.vlm

Examples

# Location unknown, scale known
ldata <- data.frame(x2 = runif(nn <- 500))
ldata <- transform(ldata, y1 = rlogis(nn, loc = 1 + 5*x2, scale = exp(2)))
fit1 <- vglm(y1 ~ x2, logistic(scale = exp(2)), data = ldata, trace = TRUE)
coef(fit1, matrix = TRUE)

# Both location and scale unknown
ldata <- transform(ldata, y2 = rlogis(nn, loc = 1 + 5*x2, scale = exp(0 + 1*x2)))
fit2 <- vglm(cbind(y1, y2) ~ x2, logistic, data = ldata, trace = TRUE)
coef(fit2, matrix = TRUE)
vcov(fit2)
summary(fit2)

---

logit

Logit Link Function

Description

Computes the logit transformation, including its inverse and the first two derivatives.
Usage

logit(theta, bvalue = NULL, inverse = FALSE, deriv = 0,
     short = TRUE, tag = FALSE)

extlogit(theta, min = 0, max = 1, bminvalue = NULL, bmaxvalue = NULL,
         inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)

Arguments

theta Numeric or character. See below for further details.
bvalue, bminvalue, bmaxvalue
See Links. These are boundary values. For extlogit, values of theta less than or
equal to A or greater than or equal to B can be replaced by bminvalue and
bmaxvalue.
min, max For extlogit, min gives A, max gives B, and for out of range values, bminvalue
         and bmaxvalue.
inverse, deriv, short, tag
Details at Links.

Details

The logit link function is very commonly used for parameters that lie in the unit
interval. Numerical values of theta close to 0 or 1 or out of range result in Inf, -Inf, NA or NaN.
The extended logit link function extlogit should be used more generally for parameters that lie in
the interval (A, B), say. The formula is

$$\log((\theta - A)/(B - \theta))$$

and the default values for A and B correspond to the ordinary logit function. Numerical values
of theta close to A or B or out of range result in Inf, -Inf, NA or NaN. However these can be replaced
by values bminvalue and bmaxvalue first before computing the link function.

Value

For logit with deriv = 0, the logit of theta, i.e., \(\log(\theta/(1-\theta))\) when inverse = FALSE,
and if inverse = TRUE then \(\exp(\theta)/(1+\exp(\theta))\).

For deriv = 1, then the function returns \(\theta/d\) eta as a function of theta if inverse = FALSE,
else if inverse = TRUE then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base \(e\).

Note

Numerical instability may occur when theta is close to 1 or 0 (for logit), or close to A or B for
extlogit. One way of overcoming this is to use, e.g., bvalue.

In terms of the threshold approach with cumulative probabilities for an ordinal response this link
function corresponds to the univariate logistic distribution (see logistic).

Author(s)

Thomas W. Yee
logit

References

See Also
links, probit, cloglog, cauchit, logistic, , loge, multilogit.

Examples
p <- seq(0.01, 0.99, by = 0.01)
logit(p)
max(abs(logit(logit(p), inverse = TRUE) - p)) # Should be 0

p <- c(seq(-0.02, 0.02, by = 0.01), seq(0.97, 1.02, by = 0.01))
logit(p) # Has NAs
logit(p, bvalue = .Machine$double.eps) # Has no NAs

p <- seq(0.9, 2.2, by = 0.1)
extlogit(p, min = 1, max = 2,
    bminvalue = 1 + .Machine$double.eps,
    bmaxvalue = 2 - .Machine$double.eps) # Has no NAs

# Not run: par(mfrow = c(2,2), lwd = (mylwd <- 2))
y <- seq(-4, 4, length = 100)
p <- seq(0.01, 0.99, by = 0.01)
for (d in 0:1) {
    matplot(p, cbind(logit(p, deriv = d), probit(p, deriv = d)),
        type = "n", col = "purple", ylab = "transformation", las = 1,
        main = if (d == 0) "Some probability link functions"
          else "First derivative")
    lines(p, logit(p, deriv = d), col = "limegreen")
    lines(p, probit(p, deriv = d), col = "purple")
    lines(p, cloglog(p, deriv = d), col = "chocolate")
    lines(p, cauchit(p, deriv = d), col = "tan")
    if (d == 0) {
        abline(v = 0.5, h = 0, lty = "dashed")
        legend(0.45, 0.5, c("logit", "probit", "cloglog", "cauchit"),
            col = c("limegreen", "purple", "chocolate", "tan"), lwd = mylwd)
    } else
        abline(v = 0.5, lty = "dashed")
}

for (d in 0) {
    matplot(y, cbind(logit(y, deriv = d, inverse = TRUE),
        probit(y, deriv = d, inverse = TRUE)), las = 1,
        type = "n", col = "purple", xlab = "transformation", ylab = "p",
        main = if (d == 0) "Some inverse probability link functions"
          else "First derivative")
    lines(y, logit(y, deriv = d, inverse = TRUE), col = "limegreen")
    lines(y, probit(y, deriv = d, inverse = TRUE), col = "purple")
    lines(y, cloglog(y, deriv = d, inverse = TRUE), col = "chocolate")
loglaplace

Log-Laplace and Logit-Laplace Distribution Family Functions

Description

Maximum likelihood estimation of the 1-parameter log-Laplace and the 1-parameter logit-Laplace distributions. These may be used for quantile regression for counts and proportions respectively.

Usage

loglaplace(tau = NULL, llocation = "loge", ilocation = NULL, kappa = sqrt(tau/(1 - tau)), Scale.arg = 1, ishrinkage = 0.95, parallel.locat = FALSE, digt = 4, idf.mu = 3, rep0 = 0.5, minquantile = 0, maxquantile = Inf, imethod = 1, zero = NULL)

logitlaplace(tau = NULL, llocation = "logit", ilocation = NULL, kappa = sqrt(tau/(1 - tau)), Scale.arg = 1, ishrinkage = 0.95, parallel.locat = FALSE, digt = 4, idf.mu = 3, rep0 = 0.5, imethod = 1, zero = NULL)

Arguments

tau, kappa

See alaplace1.

llocation

Character. Parameter link functions for location parameter $\xi$. See Links for more choices. However, this argument should be left unchanged with count data because it restricts the quantiles to be positive. With proportions data llocation can be assigned a link such as logit, probit, cloglog, etc.

ilocation

Optional initial values. If given, it must be numeric and values are recycled to the appropriate length. The default is to choose the value internally.

parallel.locat

Logical. Should the quantiles be parallel on the transformed scale (argument llocation)? Assigning this argument to TRUE circumvents the seriously embarrassing quantile crossing problem.
Initialization method. Either the value 1, 2, or ....

See alaplace1.

Numeric, positive. Replacement values for 0s and 1s respectively. For count data, values of the response whose value is 0 are replaced by rep0; it avoids computing log(0). For proportions data values of the response whose value is 0 or 1 are replaced by min(rangey[1]/2, rep0[1]/w[y = 0]) and max((1 + rangey[2])/2, 1-rep0[1]) respectively; e.g., it avoids computing logit(0) or logit(1). Here, rangey[1] is the 2-vector range(y[(y > 0) & (y < 1)]) of the response.

Numeric. The minimum and maximum values possible in the quantiles. These argument are effectively ignored by default since loge keeps all quantiles positive. However, if location = logoff(offset = 1) then it is possible that the fitted quantiles have value 0 because minquantile = 0.

These VGAM family functions implement translations of the asymmetric Laplace distribution (ALD). The resulting variants may be suitable for quantile regression for count data or sample proportions. For example, a log link applied to count data is assumed to follow an ALD. Another example is a logit link applied to proportions data so as to follow an ALD. A positive random variable \( Y \) is said to have a log-Laplace distribution if \( Y = e^W \) where \( W \) has an ALD. There are many variants of ALDs and the one used here is described in alaplace1.

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

In the extra slot of the fitted object are some list components which are useful. For example, the sample proportion of values which are less than the fitted quantile curves, which is \( \text{sum}(wprior[y <= \text{location}]) / \text{sum}(wprior) \) internally. Here, wprior are the prior weights (called ssize below), y is the response and location is a fitted quantile curve. This definition comes about naturally from the transformed ALD data.

The VGAM family function logitlaplace1 will not handle a vector of just 0s and 1s as the response; it will only work satisfactorily if the number of trials is large.

See alaplace1 for other warnings. Care is needed with tau values which are too small, e.g., for count data the sample proportion of zeros must be less than all values in tau. Similarly, this also holds with logitlaplace1, which also requires all tau values to be less than the sample proportion of ones.

The form of input for logitlaplace1 as response is a vector of proportions (values in \([0, 1]\)) and the number of trials is entered into the weights argument of vglm/vgam. See Example 2 below. See alaplace1 for other notes in general.
Author(s)
Thomas W. Yee

References

See Also
alaplaceQ, dloglap.

Examples
# Example 1: quantile regression of counts with regression splines
set.seed(123); my.k <- exp(0)
data <- data.frame(x2 = sort(runif(n <- 500)))
mymu <- function(x) exp(1 + 3*sin(2*x) / (x+0.5)^2)
data <- transform(data, y = rnbinom(n, mu = mymu(x2), size = my.k))
mytau <- c(0.1, 0.25, 0.5, 0.75, 0.9); mydof = 3
# halfstepping is usual:
fitp <- vglm(y ~ sm.bs(x2, df = mydof), data = data, trace = TRUE,
            loglaplace(tau = mytau, parallel.locat = TRUE))

## Not run:
par(las = 1)  # Plot on a log1p() scale
mylwd <- 1.5

plot(jitter(log1p(y), factor = 1.5) ~ x2, data, col = "red", pch = "o",
      main = "Example 1; darkgreen=truth, blue=estimated", cex = 0.75)
with(data, matlines(x2, log1p(fitted(fitp))), col = "blue",
     lty = 1, lwd = mylwd))
finexgrid <- seq(0, 1, len = 201)
for (ii in 1:length(mytau))
  lines(finexgrid, col = "darkgreen", lwd = mylwd,
        log1p(qnbinom(p = mytau[ii], mu = mymu(finexgrid), si = my.k))))

## End(Not run)
fitp@extra  # Contains useful information

# Example 2: sample proportions
set.seed(123); nnn <- 1000; ssize <- 100  # ssize = 1 will not work!
adata <- data.frame(x2 = sort(runif(nnn)))
mymu <- function(x) logit(1.0 + 4*x, inv = TRUE)
data <- transform(adata, ssize = ssize,
The Log-Laplace Distribution

Description

Density, distribution function, quantile function and random generation for the 3-parameter log-Laplace distribution with location parameter location.ald, scale parameter scale.ald (on the log scale), and asymmetry parameter kappa.
Usage

dloglap(x, location.ald = 0, scale.ald = 1,  
tau = 0.5, kappa = sqrt(tau/(1-tau)), log = FALSE)
ploglap(q, location.ald = 0, scale.ald = 1,  
tau = 0.5, kappa = sqrt(tau/(1-tau)), lower.tail = TRUE, log.p = FALSE)
qloglap(p, location.ald = 0, scale.ald = 1,  
tau = 0.5, kappa = sqrt(tau/(1-tau)), lower.tail = TRUE, log.p = FALSE)
rloglap(n, location.ald = 0, scale.ald = 1,  
tau = 0.5, kappa = sqrt(tau/(1-tau)))

Arguments

x, q
  vector of quantiles.
p
  vector of probabilities.n
  number of observations. If length(n) > 1 then the length is taken to be the
  number required.

location.ald, scale.ald
  the location parameter \( \xi \) and the (positive) scale parameter \( \sigma \), on the log scale.

tau
  the quantile parameter \( \tau \). Must consist of values in \((0, 1)\). This argument is used
  to specify kappa and is ignored if kappa is assigned.

kappa
  the asymmetry parameter \( \kappa \). Must consist of positive values.

log
  if TRUE, probabilities \( p \) are given as \( \log(p) \).

lower.tail, log.p
  Same meaning as in \texttt{pnorm} or \texttt{qnorm}.

Details

A positive random variable \( Y \) is said to have a log-Laplace distribution if \( \log(Y) \) has an asymmetric
Laplace distribution (ALD). There are many variants of ALDs and the one used here is described
in \texttt{alaplace3}.

Value

dloglap gives the density, ploglap gives the distribution function, qloglap gives the quantile
function, and rloglap generates random deviates.

Author(s)

T. W. Yee and Kai Huang

References

Journal}, 3, 467–495.

See Also

dalap, alaplace3, loglaplace1.
Examples

```r
loc <- 0; sigma <- exp(0.5); kappa <- 1
x <- seq(-0.2, 5, by = 0.01)
## Not run:
plot(x, dloglap(x, loc, sigma, kappa = kappa), type = "l", col = "blue",
     main = "Blue is density, red is cumulative distribution function",
     ylim = c(0,1), sub = "Purple are 5,10,....,95 percentiles", las = 1, ylab = "")
abline(h = 0, col = "blue", lty = 2)
lines(qloglap(seq(0.05,0.95,by = 0.05), loc, sigma, kappa = kappa),
     dloglap(qloglap(seq(0.05,0.95,by = 0.05), loc, sigma, kappa = kappa),
     loc, sigma, kappa = kappa), col = "purple", lty = 3, type = "h")
lines(x, ploglap(x, loc, sigma, kappa = kappa), type = "l", col = "red")
abline(h = 0, lty = 2)
## End(Not run)
ploglap(qloglap(seq(0.05,0.95,by = 0.05), loc, sigma, kappa = kappa),
     loc, sigma, kappa = kappa)
```

logLik.vlm

**Extract Log-likelihood for VGLMs/VGAMs/etc.**

Description

Calculates the log-likelihood value or the element-by-element contributions of the log-likelihood.

Usage

```r
## S3 method for class 'vlm'
logLik(object, summation = TRUE, ...)
```

Arguments

- **object**: Some VGAM object, for example, having class `vglmff-class`.
- **summation**: Logical, apply `sum`? If `FALSE` then a n-vector or n-row matrix (with the number of responses as the number of columns) is returned. Each element is the contribution to the log-likelihood.
- **...**: Currently unused. In the future: other possible arguments fed into `logLik` in order to compute the log-likelihood.

Details

By default, this function returns the log-likelihood of the object. Thus this code relies on the log-likelihood being defined, and computed, for the object.

Value

Returns the log-likelihood of the object. If `summation = FALSE` then a n-vector or n-row matrix (with the number of responses as the number of columns) is returned. Each element is the contribution to the log-likelihood. The prior weights are assimilated within the answer.
Warning

Not all VGAM family functions have had the summation checked.

Note

Not all VGAM family functions currently have the summation argument implemented.

Author(s)

T. W. Yee.

See Also

VGLMs are described in \texttt{vglm-class}; VGAMs are described in \texttt{vgam-class}; RR-VGLMs are described in \texttt{rrvglm-class}; \texttt{AIC}.

Examples

\begin{verbatim}
zdata <- data.frame(x2 = runif(nn <- 50))
zdata <- transform(zdata, Ps01 = logit(-0.5, inverse = TRUE),
    Ps02 = logit(0.5, inverse = TRUE),
    lambda1 = loge(-0.5 + 2*x2, inverse = TRUE),
    lambda2 = loge(0.5 + 2*x2, inverse = TRUE))
zdata <- transform(zdata, y1 = rzipois(nn, lambda = lambda1, pstr0 = Ps01),
    y2 = rzipois(nn, lambda = lambda2, pstr0 = Ps02))

with(zdata, table(y1)) # Eyeball the data
with(zdata, table(y2))
fit2 <- vglm(cbind(y1, y2) ~ x2, zipoisson(zero = NULL), data = zdata)

logLik(fit2) # Summed over the two responses
sum(logLik(fit2, sum = FALSE)) # For checking purposes
(ll.matrix <- logLik(fit2, sum = FALSE)) # nn x 2 matrix
colSums(ll.matrix) # log-likelihood for each response
\end{verbatim}

\texttt{loglinb2}

\textit{Loglinear Model for Two Binary Responses}

Description

Fits a loglinear model to two binary responses.

Usage

\begin{verbatim}
loglinb2(exchangeable = FALSE, zero = 3)
\end{verbatim}
Arguments

exchangeable Logical. If TRUE, the two marginal probabilities are constrained to be equal. Should be set TRUE for ears, eyes, etc. data.

zero Which linear/additive predictor is modelled as an intercept only? A NULL means none of them.

Details

The model is

\[ P(Y_1 = y_1, Y_2 = y_2) = \exp(u_0 + u_1 y_1 + u_2 y_2 + u_{12} y_1 y_2) \]

where \( y_1 \) and \( y_2 \) are 0 or 1, and the parameters are \( u_1, u_2, u_{12} \). The normalizing parameter \( u_0 \) can be expressed as a function of the other parameters, viz.,

\[ u_0 = -\log[1 + \exp(u_1) + \exp(u_2) + \exp(u_1 + u_2 + u_{12})]. \]

The linear/additive predictors are \((\eta_1, \eta_2, \eta_3)^T = (u_1, u_2, u_{12})^T\).

Value

An object of class "vglmff" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, \texttt{rrvglm} and \texttt{vgam}.

When fitted, the \texttt{fitted.values} slot of the object contains the four joint probabilities, labelled as \((Y_1, Y_2) = (0,0), (0,1), (1,0), (1,1)\), respectively.

Note

The response must be a two-column matrix of ones and zeros only. This is more restrictive than \texttt{binom2.or}, which can handle more types of input formats. Note that each of the 4 combinations of the multivariate response need to appear in the data set.

Author(s)

Thomas W. Yee

References


See Also

\texttt{binom2.or}, \texttt{binom2.rho}, \texttt{loglinb3}. 
Examples

coalminers <- transform(coalminers, Age = (age - 42) / 5)

# Get the n x 4 matrix of counts
fit0 <- vglm(cbind(nBnW,nBW,BnW,BW) ~ Age, binom2.or, data = coalminers)
counts <- round(c(weights(fit0, type = "prior")) * depvar(fit0))

# Create a n x 2 matrix response for loglinb2()
# bwmat <- matrix(c(0,0,0,1,1,0,1,1), 4, 2, byrow = TRUE)
bwmat <- cbind(bln = c(0,0,1,1), wheeze = c(0,1,0,1))
matof1 <- matrix(1, nrow(counts), 1)
newminers <- data.frame(bln = kronecker(matof1, bwmat[, 1]),
                        wheeze = kronecker(matof1, bwmat[, 2]),
                        wt = c(t(counts)),
                        Age = with(coalminers, rep(age, rep(4, length(age)))))
newminers <- newminers[with(newminers, wt > 0),]

fit <- vglm(cbind(bln, wheeze) ~ Age, loglinb2(zero = NULL),
            weight = wt, data = newminers)
coef(fit, matrix = TRUE) # Same! (at least for the log odds-ratio)
summary(fit)

# Try reconcile this with McCullagh and Nelder (1989), p.234
(0.166-0.131) / 0.027458 # 1.275 is approximately 1.25

loglinb3

Loglinear Model for Three Binary Responses

Description

Fits a loglinear model to three binary responses.

Usage

loglinb3(exchangeable = FALSE, zero = 4:6)

Arguments

exchangeable Logical. If TRUE, the three marginal probabilities are constrained to be equal.
zero Which linear/additive predictor is modelled as an intercept only? A NULL means none.

Details

The model is \( P(Y_1 = y_1, Y_2 = y_2, Y_3 = y_3) = \)
\[ \exp(u_0 + u_1 y_1 + u_2 y_2 + u_3 y_3 + u_{12} y_1 y_2 + u_{13} y_1 y_3 + u_{23} y_2 y_3) \]
where \(y_1, y_2\) and \(y_3\) are 0 or 1, and the parameters are \(u_1, u_2, u_3, u_{12}, u_{13}, u_{23}\). The normalizing parameter \(u_0\) can be expressed as a function of the other parameters. Note that a third-order association parameter, \(u_{123}\) for the product \(y_1 y_2 y_3\), is assumed to be zero for this family function.

The linear/additive predictors are \((\eta_1, \eta_2, \ldots, \eta_6)^T = (u_1, u_2, u_3, u_{12}, u_{13}, u_{23})^T\).

**Value**

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

When fitted, the `fitted.values` slot of the object contains the eight joint probabilities, labelled as 
\((Y_1, Y_2, Y_3) = (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1),\) respectively.

**Note**

The response must be a three-column matrix of ones and zeros only. Note that each of the 8 combinations of the multivariate response need to appear in the data set, therefore data sets will need to be large in order for this family function to work.

**Author(s)**

Thomas W. Yee

**References**


**See Also**

`loglinb2`, `hunua`.

**Examples**

```r
fit <- vglm(cbind(cyadea, beitaw, kniexc) ~ altitude, loglinb3, data = hunua)
coef(fit, matrix = TRUE)
head(fitted(fit))
summary(fit)
```
loglog  

Log-log Link Function

Description

Computes the log-log transformation, including its inverse and the first two derivatives.

Usage

loglog(theta, bvalue = NULL, inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)

Arguments

theta  Numeric or character. See below for further details.
bvalue  Values of theta which are less than or equal to 1 can be replaced by bvalue before computing the link function value. The component name bvalue stands for “boundary value”. See Links for more information.

inverse, deriv, short, tag  
Details at Links.

Details

The log-log link function is commonly used for parameters that are greater than unity. Numerical values of theta close to 1 or out of range result in Inf, -Inf, NA or NaN.

Value

For deriv = 0, the log of theta, i.e., log(log(theta)) when inverse = FALSE, and if inverse = TRUE then exp(exp(theta)).

For deriv = 1, then the function returns \( \frac{d\theta}{d\eta} \) as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base \( e \).

Note

Numerical instability may occur when theta is close to 1 unless bvalue is used.

Author(s)

Thomas W. Yee

References

lognormal

See Also

Links, loge, logoff.

Examples

```r
x <- seq(0.8, 1.5, by = 0.1)
loglog(x) # Has NAs
loglog(x, bvalue = 1.0 + .Machine$double.eps) # Has no NAs

x <- seq(1.01, 10, len = 100)
loglog(x)
max(abs(loglog(loglog(x), inverse = TRUE) - x)) # Should be 0
```

Description

Maximum likelihood estimation of the (univariate) lognormal distribution.

Usage

```r
lognormal(lmeanlog = "identitylink", lsdlog = "loge", zero = 2)
```

Arguments

- `lmeanlog`: Parameter link functions applied to the mean and (positive) \( \sigma \) (standard deviation) parameter. Both of these are on the log scale. See Links for more choices.

- `lsdlog`: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. For `lognormal()`, the values must be from the set \{1,2\} which correspond to \( \mu \), \( \sigma \), respectively. See CommonVGAMffArguments for more information.

Details

A random variable \( Y \) has a 2-parameter lognormal distribution if \( \log(Y) \) is distributed \( N(\mu, \sigma^2) \). The expected value of \( Y \), which is

\[
E(Y) = \exp(\mu + 0.5\sigma^2)
\]

and not \( \mu \), make up the fitted values.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.
Author(s)

T. W. Yee

References


See Also

`rlnorm`, `uninormal`, `CommonVGAMffArguments`, `simulate.vlm`.

Examples

```r
ldata2 <- data.frame(x2 = runif(nn <- 1000))
ldata2 <- transform(ldata2, y1 = rlnorm(nn, mean = 1 + 2 * x2, sd = exp(-1)),
                    y2 = rlnorm(nn, mean = 1, sd = exp(-1 + x2)))
fit1 <- vglm(y1 ~ x2, lognormal(zero = 2), data = ldata2, trace = TRUE)
fit2 <- vglm(y2 ~ x2, lognormal(zero = 1), data = ldata2, trace = TRUE)
coef(fit1, matrix = TRUE)
coef(fit2, matrix = TRUE)
```

Description

Computes the log transformation with an offset, including its inverse and the first two derivatives.

Usage

```r
logoff(theta, offset = 0, inverse = FALSE, deriv = 0,
       short = TRUE, tag = FALSE)
```

Arguments

- `theta` Numeric or character. See below for further details.
- `offset` Offset value. See Links.
- `inverse`, `deriv`, `short`, `tag`

Details

The log-offset link function is very commonly used for parameters that are greater than a certain value. In particular, it is defined by \( \log(\theta + \text{offset}) \) where offset is the offset value. For example, if `offset = 0.5` then the value of `theta` is restricted to be greater than \(-0.5\). Numerical values of `theta` close to `-offset` or out of range result in `Inf`, `-Inf`, `NA` or `NaN`.

Links

The log-offset link function is commonly used in various statistical models, especially in VGAM packages. It allows for modeling parameters that must be positive, effectively shifting the parameter space to ensure positivity.

```r
ldata2 <- data.frame(x2 = runif(nn <- 1000))
ldata2 <- transform(ldata2, y1 = rlnorm(nn, mean = 1 + 2 * x2, sd = exp(-1)),
                    y2 = rlnorm(nn, mean = 1, sd = exp(-1 + x2)))
fit1 <- vglm(y1 ~ x2, lognormal(zero = 2), data = ldata2, trace = TRUE)
fit2 <- vglm(y2 ~ x2, lognormal(zero = 1), data = ldata2, trace = TRUE)
coef(fit1, matrix = TRUE)
coef(fit2, matrix = TRUE)
```
Value

For \( \text{deriv} = 0 \), the log of \( \theta + \text{offset} \), i.e., \( \log(\theta + \text{offset}) \) when \( \text{inverse} = \text{FALSE} \), and if \( \text{inverse} = \text{TRUE} \) then \( \exp(\theta) - \text{offset} \).

For \( \text{deriv} = 1 \), then the function returns \( d \theta / d \eta \) as a function of \( \theta \) if \( \text{inverse} = \text{FALSE} \), else if \( \text{inverse} = \text{TRUE} \) then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base \( e \).

Note

The default means this function is identical to \( \text{loge} \).

Numerical instability may occur when \( \theta \) is close to \( -\text{offset} \).

Author(s)

Thomas W. Yee

References


See Also

Links, \( \text{loge} \).

Examples

```r
## Not run:
logoff(seq(-0.2, 0.5, by = 0.1))
logoff(seq(-0.2, 0.5, by = 0.1), offset = 0.5)
  log(seq(-0.2, 0.5, by = 0.1) + 0.5)
## End(Not run)
```

Description

Density, distribution function, quantile function and random generation for the Lomax distribution with scale parameter \( \text{scale} \) and shape parameter \( q \).

Usage

```r
dlomax(x, scale = 1, shape3.q, log = FALSE)
plomax(q, scale = 1, shape3.q, lower.tail = TRUE, log.p = FALSE)
qlomax(p, scale = 1, shape3.q, lower.tail = TRUE, log.p = FALSE)
rlomax(n, scale = 1, shape3.q)
```
Arguments

- `x, q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
- `scale` scale parameter.
- `shape3.q` shape parameter.
- `log` Logical. If `log = TRUE` then the logarithm of the density is returned.
- `lower.tail, log.p` Same meaning as in `pnorm` or `qnorm`.

Details

See `lomax`, which is the VGAM family function for estimating the parameters by maximum likelihood estimation.

Value

dlomax gives the density, plomax gives the distribution function, qlomax gives the quantile function, and rlomax generates random deviates.

Note

The Lomax distribution is a special case of the 4-parameter generalized beta II distribution.

Author(s)

T. W. Yee and Kai Huang

References


See Also

`lomax, genbetaII`.

Examples

```r
probs <- seq(0.1, 0.9, by = 0.1)
max(abs(plomax(qlomax(p = probs, shape3.q = 1),
               shape3.q = 1) - probs)) # Should be 0

## Not run: par(mfrow = c(1, 2))
x <- seq(-0.01, 5, len = 401)
plot(x, dexp(x), type = "l", col = "black", ylab = "", ylim = c(0, 3),
     main = "Black is standard exponential, others are dlomax(x, shape3.q)")
```
The 2-parameter Lomax distribution is the 4-parameter generalized beta II distribution with shape parameters \( a = p = 1 \). It is probably more widely known as the Pareto (II) distribution. It is also the 3-parameter Singh-Maddala distribution with shape parameter \( a = 1 \), as well as the beta distribution of the second kind with \( p = 1 \). More details can be found in Kleiber and Kotz (2003).

The Lomax distribution has density

\[
f(y) = q / \left[ b \{1 + y/b\}^{1+q}\right]
\]
for $b > 0$, $q > 0$, $y \geq 0$. Here, $b$ is the scale parameter scale, and $q$ is a shape parameter. The cumulative distribution function is

$$F(y) = 1 - [1 + (y/b)]^{-q}.$$ 

The mean is

$$E(Y) = b/(q - 1)$$

provided $q > 1$; these are returned as the fitted values. This family function handles multiple responses.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

See the notes in genbetaII.

Author(s)

T. W. Yee

References


See Also

lomax, genbetaII, betaII, dagum, sinmad, fisk, inv.lomax, paralogistic, inv.paralogistic, simulate.vlm.

Examples

```r
ldata <- data.frame(y = rnorm(1000, scale = 1), y = rnorm(1000, scale = 2))
fit <- vglm(y ~ 1, lomax, data = ldata, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)
```
lqnorm

Minimizing the L-q norm Family Function

Description

Minimizes the L-q norm of residuals in a linear model.

Usage

lqnorm(qpower = 2, link = "identitylink",
       imethod = 1, imu = NULL, ishrinkage = 0.95)

Arguments

qpower A single numeric, must be greater than one, called q below. The absolute value of residuals are raised to the power of this argument, and then summed. This quantity is minimized with respect to the regression coefficients.

link Link function applied to the ‘mean’ \( \mu \). See Links for more details.

imethod Must be 1, 2 or 3. See CommonVGAMffArguments for more information. Ignored if imu is specified.

imu Numeric, optional initial values used for the fitted values. The default is to use imethod = 1.

ishrinkage How much shrinkage is used when initializing the fitted values. The value must be between 0 and 1 inclusive, and a value of 0 means the individual response values are used, and a value of 1 means the median or mean is used. This argument is used in conjunction with immethod = 3.

Details

This function minimizes the objective function

\[
\sum_{i=1}^{n} w_i (|y_i - \mu_i|)^q
\]

where \( q \) is the argument qpower, \( \eta_i = g(\mu_i) \) where \( g \) is the link function, and \( \eta_i \) is the vector of linear/additive predictors. The prior weights \( w_i \) can be inputted using the weights argument of vlm/vglm/vgam etc.; it should be just a vector here since this function handles only a single vector or one-column response.

Numerical problem will occur when \( q \) is too close to one. Probably reasonable values range from 1.5 and up, say. The value \( q = 2 \) corresponds to ordinary least squares while \( q = 1 \) corresponds to the MLE of a double exponential (Laplace) distribution. The procedure becomes more sensitive to outliers the larger the value of \( q \).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.
Warning

Convergence failure is common, therefore the user is advised to be cautious and monitor convergence!

Note

This VGAM family function is an initial attempt to provide a more robust alternative for regression and/or offer a little more flexibility than least squares. The @misc slot of the fitted object contains a list component called objectiveFunction which is the value of the objective function at the final iteration.

Author(s)

Thomas W. Yee

References


See Also

gaussianff.

Examples

```r
set.seed(123)
ldata <- data.frame(x = sort(runif(nn <- 10 )))
realfun <- function(x) 4 + 5*x
ldata <- transform(ldata, y = realfun(x) + rnorm(nn, sd = exp(-1)))
# Make the first observation an outlier
ldata <- transform(ldata, y = c(4*y[1], y[-1]), x = c(-1, x[-1]))
fit <- vglm(y ~ x, lqnorm(qpower = 1.2), data = ldata)
coef(fit, matrix = TRUE)
head(fitted(fit))
fit@misc$qpower
fit@misc$objectiveFunction

## Not run:
# Graphical check
with(ldata, plot(x, y, main = paste("LS = red, lqnorm = blue (qpower = ",
fit@misc$qpower", ", truth = black", sep = ",", col = "blue"))
lmfit <- lm(y ~ x, data = ldata)
with(ldata, lines(x, fitted(fit), col = "blue"))
with(ldata, lines(x, lmfit$fitted, col = "red"))
with(ldata, lines(x, realfun(x), col = "black"))
## End(Not run)
```
**Description**

*lrtest* is a generic function for carrying out likelihood ratio tests. The default method can be employed for comparing nested VGLMs (see details below).

**Usage**

```r
lrtest(object, ...) lrtest_vglm(object, ..., name = NULL)
```

**Arguments**

- `object` a `vglm` object. See below for details.
- `...` further object specifications passed to methods. See below for details.
- `name` a function for extracting a suitable name/description from a fitted model object. By default the name is queried by calling `formula`.

**Details**

*lrtest* is intended to be a generic function for comparisons of models via asymptotic likelihood ratio tests. The default method consecutively compares the fitted model object `object` with the models passed in `...`. Instead of passing the fitted model objects in `...`, several other specifications are possible. The updating mechanism is the same as for `waldtest`: the models in `...` can be specified as integers, characters (both for terms that should be eliminated from the previous model), update formulas or fitted model objects. Except for the last case, the existence of an `update` method is assumed. See `waldtest` for details.

Subsequently, an asymptotic likelihood ratio test for each two consecutive models is carried out: Twice the difference in log-likelihoods (as derived by the `logLik` methods) is compared with a Chi-squared distribution.

**Value**

An object of class "VGAManova" which contains a slot with the log-likelihood, degrees of freedom, the difference in degrees of freedom, likelihood ratio Chi-squared statistic and corresponding p value. These are printed by `stats::print.anova();` see `anova`.

**Warning**

Several `VGAM` family functions implement distributions which do not satisfying the usual regularity conditions needed for the LRT to work. No checking or warning is given for these.
Note

The code was adapted directly from `lmtest` (written by T. Hothorn, A. Zeileis, G. Millo, D. Mitchell) and made to work for VGLMs and S4. This help file also was adapted from `lmtest`.

Approximate LRTs might be applied to VGAMs, as produced by `vgam`, but it is probably better in inference to use `vglm` with regression splines (`bs` and `ns`). This methods function should not be applied to other models such as those produced by `rrvglm`, by `cqc`, by `cao`.

See Also

`lmtest`, `vglm`.

Examples

```r
setNseedHQI
pneumo <- transform(pneumo, let = log(exposure.time), x3 = runif(nrow(pneumo)))
fit1 <- vglm(cbind(normal, mild, severe) ~ let , propodds, data = pneumo)
fit2 <- vglm(cbind(normal, mild, severe) ~ let + x3, propodds, data = pneumo)
fit3 <- vglm(cbind(normal, mild, severe) ~ let , cumulative, data = pneumo)
# Various equivalent specifications of the LR test for testing x3
(ans1 <- lrtest(fit2, fit1))
ans2 <- lrtest(fit2, 2)
ans3 <- lrtest(fit2, "x3")
ans4 <- lrtest(fit2, . ~ . ~ x3)
c(all.equal(ans1, ans2), all.equal(ans1, ans3), all.equal(ans1, ans4))

# Doing it manually
(testStatistic <- 2 * (logLik(fit2) - logLik(fit1)))
(mypval <- pchisq(testStatistic, df = length(coef(fit2)) - length(coef(fit1)),
   lower.tail = FALSE))

(ans4 <- lrtest(fit3, fit1)) # Test proportional odds (parallelism) assumption
```

---

### lvplot

**Latent Variable Plot**

Generic function for a latent variable plot (also known as an ordination diagram by ecologists).

#### Usage

```r
lvplot(object, ...)
```

#### Arguments

- **object**
  - An object for a latent variable plot is meaningful.

- **...**
  - Other arguments fed into the specific methods function of the model. They usually are graphical parameters, and sometimes they are fed into the methods function for `Coef`. 
Details

Latent variables occur in reduced-rank regression models, as well as in quadratic and additive ordination. For the latter, latent variables are often called the site scores. Latent variable plots were coined by Yee (2004), and have the latent variable as at least one of its axes.

Value

The value returned depends specifically on the methods function invoked.

Note

Latent variables are not really applicable to \texttt{vglm}/\texttt{vgam} models.

Author(s)

Thomas W. Yee

References


See Also

\texttt{lvplot.qrrvglm}, \texttt{lvplot.cao}, \texttt{latvar}, \texttt{trplot}.

Examples

```r
## Not run:
hs = hspider[,1:6] # Standardized environmental vars
set.seed(123)
p1 <- cao(cbind(Pardlug, Pardmont, Pardnigr, Pardpull, Zoraspin) ~
  WaterCon + BareSand + FallTwig +
  CoveMoss + CoveHerb + RefILux,
  family = poissonff, data = hspider, Bestof = 3,
  df1.nl = c(Zoraspin = 2.5, 3), Crowlpositive = TRUE)
index <- 1:ncol(depvar(p1))
lvplot(p1, lcol = index, pcol = index, y = TRUE, las = 1)

## End(Not run)
```
Latent Variable Plot for QO models

Description

Produces an ordination diagram (latent variable plot) for quadratic ordination (QO) models. For rank-1 models, the x-axis is the first ordination/constrained/canonical axis. For rank-2 models, the x- and y-axis are the first and second ordination axes respectively.

Usage

```r
lvplot.qrrvglm(object, var.latvar = FALSE, refResponse = NULL, 
add = FALSE, show.plot = TRUE, 
rug = TRUE, y = FALSE, type = c("fitted.values", "predictors"), 
 xlab = paste("Latent Variable", if (Rank == 1) " 1" else " 2", sep = ""), 
ylab = if (Rank == 1) switch(type, predictors = "Predictors", 
 fithed.values = "Fitted values") else "Latent Variable 2", 
pceX = par()$cex, pcol = par()$col, pch = par()$pch, 
llty = par()$lty, lcol = par()$col, llwd = par()$lwd, 
label.arg = FALSE, adj.arg = -0.1, 
ellipse = 0.95, Absolute = FALSE, 
elty = par()$lty, ecol = par()$col, elwd = par()$lwd, egrid = 200, 
chull.arg = FALSE, clty = 2, ccol = par()$col, clwd = par()$lwd, 
ccex = " "
C = FALSE, OriginC = c("origin", "mean"), 
Clty = par()$lty, Ccol = par()$col, Clwd = par()$lwd, 
Ccex = par()$cex, Cadj.arg = -0.1, stretchC = 1, 
sites = FALSE, spch = NULL, scol = par()$col, scex = par()$cex, 
sfont = par()$font, check.ok = TRUE, ...)
```

Arguments

- `object`: A CQO object.
- `var.latvar`: Logical that is fed into `coef.qrrvglm`.
- `refResponse`: Integer or character that is fed into `coef.qrrvglm`.
- `add`: Logical. Add to an existing plot? If FALSE, a new plot is made.
- `show.plot`: Logical. Plot it?
- `rug`: Logical. If TRUE, a rug plot is plotted at the foot of the plot (applies to rank-1 models only). These values are jittered to expose ties.
- `y`: Logical. If TRUE, the responses will be plotted (applies only to rank-1 models and if `type = "fitted.values"`).
- `type`: Either "fitted.values" or "predictors", specifies whether the y-axis is on the response or eta-scales respectively.
- `xlab`: Caption for the x-axis. See `par`.
ylab  Caption for the y-axis. See par.
plex  Character expansion of the points. Here, for rank-1 models, points are the response y data. For rank-2 models, points are the optimums. See the cex argument in par.
pcol  Color of the points. See the col argument in par.
pch   Either an integer specifying a symbol or a single character to be used as the default in plotting points. See par. The pch argument can be of length M, the number of species.
llty  Line type. Rank-1 models only. See the lty argument of par.
lcol  Line color. Rank-1 models only. See the col argument of par.
lwd  Line width. Rank-1 models only. See the lwd argument of par.
label.arg Logical. Label the optimums and C? (applies only to rank-2 models only).
adj.arg Justification of text strings for labelling the optimums (applies only to rank-2 models only). See the adj argument of par.
ellipse Numerical, of length 0 or 1 (applies only to rank-2 models only). If Absolute is TRUE then ellipse should be assigned a value that is used for the elliptical contouring. If Absolute is FALSE then ellipse should be assigned a value between 0 and 1, for example, setting ellipse = 0.9 means an ellipse with contour = 90% of the maximum will be plotted about each optimum. If ellipse is a negative value, then the function checks that the model is an equal-tolerances model and var1.latvar = FALSE, and if so, plots circles with radius = -ellipse. For example, setting ellipse = -1 will result in circular contours that have unit radius (in latent variable units). If ellipse is null or FALSE then no ellipse is drawn around the optimums.

Absolute Logical. If TRUE, the contours corresponding to ellipse are on an absolute scale. If FALSE, the contours corresponding to ellipse are on a relative scale.
eltty  Line type of the ellipses. See the lty argument of par.
ecol  Line color of the ellipses. See the col argument of par.
elwd  Line width of the ellipses. See the lwd argument of par.
egrid Numerical. Line resolution of the ellipses. Choosing a larger value will result in smoother ellipses. Useful when ellipses are large.
chull.arg Logical. Add a convex hull around the site scores?
clty  Line type of the convex hull. See the lty argument of par.
ccol  Line color of the convex hull. See the col argument of par.
clwd  Line width of the convex hull. See the lwd argument of par.
cpch  Character to be plotted at the intersection points of the convex hull. Having white spaces means that site labels are not obscured there. See the pch argument of par.
C Logical. Add C (represented by arrows emanating from OriginC) to the plot?
OriginC Character or numeric. Where the arrows representing C emanate from. If character, it must be one of the choices given. By default the first is chosen. The value "origin" means c(0,0). The value "mean" means the sample mean of the latent variables (centroid). Alternatively, the user may specify a numerical vector of length 2.
C1ty
Line type of the arrows representing C. See the lty argument of \texttt{par}.

Ccol
Line color of the arrows representing C. See the col argument of \texttt{par}.

Clwd
Line width of the arrows representing C. See the lwd argument of \texttt{par}.

Ccex
Numeric. Character expansion of the labelling of C. See the cex argument of \texttt{par}.

Cadj.arg
Justification of text strings when labelling C. See the adj argument of \texttt{par}.

stretchC
Numerical. Stretching factor for C. Instead of using \texttt{C}, \texttt{stretchC * C} is used.

sites
Logical. Add the site scores (aka latent variable values, nu's) to the plot? (applies only to rank-2 models only).

spch
Plotting character of the site scores. The default value of NULL means the row labels of the data frame are used. They often are the site numbers. See the pch argument of \texttt{par}.

scol
Color of the site scores. See the col argument of \texttt{par}.

scex
Character expansion of the site scores. See the cex argument of \texttt{par}.

sfont
Font used for the site scores. See the font argument of \texttt{par}.

check.ok
Logical. Whether a check is performed to see that \texttt{norrr} \sim \texttt{Q} was used. It doesn't make sense to have a latent variable plot unless this is so.

... Arguments passed into the \texttt{plot} function when setting up the entire plot. Useful arguments here include \texttt{xlim} and \texttt{ylim}.

Details
This function only works for rank-1 and rank-2 QRR-VGLMs with argument \texttt{norrr} \sim 1.

For unequal-tolerances models, the latent variable axes can be rotated so that at least one of the tolerance matrices is diagonal; see \texttt{Coef.qrrvglm} for details.

Arguments beginning with “p” correspond to the points e.g., \texttt{pcex} and \texttt{pcol} correspond to the size and color of the points. Such “p” arguments should be vectors of length 1, or \texttt{n}, the number of sites. For the rank-2 model, arguments beginning with “p” correspond to the optimums.

Value
Returns a matrix of latent variables (site scores) regardless of whether a plot was produced or not.

Warning
Interpretation of a latent variable plot (CQO diagram) is potentially very misleading in terms of distances if (i) the tolerance matrices of the species are unequal and (ii) the contours of these tolerance matrices are not included in the ordination diagram.

Note
A species which does not have an optimum will not have an ellipse drawn even if requested, i.e., if its tolerance matrix is not positive-definite.

Plotting C gives a visual display of the weights (loadings) of each of the variables used in the linear combination defining each latent variable.
The arguments `elty`, `ecol` and `elwd`, may be replaced in the future by `lty`, `lcol` and `lwd`, respectively.

For rank-1 models, a similar function to this one is `perspqrvglm`. It plots the fitted values on a more fine grid rather than at the actual site scores here. The result is a collection of smooth bell-shaped curves. However, it has the weakness that the plot is more divorced from the data; the user thinks it is the truth without an appreciation of the statistical variability in the estimates.

In the example below, the data comes from an equal-tolerances model. The species’ tolerance matrices are all the identity matrix, and the optimums are at (0,0), (1,1) and (-2,0) for species 1, 2, 3 respectively.

Author(s)

Thomas W. Yee

References


See Also

`lvplot`, `perspqrvglm`, `Coeff.qrrvglm`, `par`, `cqq`.

Examples

```r
set.seed(123); nn <- 200
cdata <- data.frame(x2 = rnorm(nn), # Has mean 0 (needed when I.tol=TRUE)
x3 = rnorm(nn), # Has mean 0 (needed when I.tol=TRUE)
x4 = rnorm(nn)) # Has mean 0 (needed when I.tol=TRUE)
cdata <- transform(cdata, latvar1 = x2 + x3 - 2*x4,
                   latvar2 = -x2 + x3 + 0*x4)
# Nb. latvar2 is weakly correlated with latvar1

cdata <- transform(cdata,
                   lambda1 = exp(6 - 0.5 * (latvar1-0)^2 - 0.5 * (latvar2-0)^2),
                   lambda2 = exp(5 - 0.5 * (latvar1-1)^2 - 0.5 * (latvar2-1)^2),
                   lambda3 = exp(5 - 0.5 * (latvar1+2)^2 - 0.5 * (latvar2-0)^2))
cdata <- transform(cdata,
                   spp1 = rpois(nn, lambda1),
                   spp2 = rpois(nn, lambda2),
                   spp3 = rpois(nn, lambda3))
set.seed(111)
## Not run:
p2 <- cqq(apply(spp1, 1, function(x) pchisq(x, df=1)))
if (deviance(p2) > 506) stop("Suboptimal fit obtained")
```

## End(Not run)
lvplot.rvglm

Latent Variable Plot for RR-VGLMs

Description

Produces an ordination diagram (also known as a biplot or latent variable plot) for reduced-rank vector generalized linear models (RR-VGLMs). For rank-2 models only, the x- and y-axis are the first and second canonical axes respectively.

Usage

```r
lvplot.rvglm(object,
A = TRUE, C = TRUE, scores = FALSE, show.plot = TRUE,
groups = rep(1, n), gapC = sqrt(sum(par()$cxy^2)),
scaleA = 1,
xlab = "Latent Variable 1", ylab = "Latent Variable 2",
Alabels = if (length(object@misc$predictors.names))
object@misc$predictors.names else paste("LP", 1:M, sep = ""),
Aadj = par()$adj, Acex = par()$cex, Acol = par()$col,
Apch = NULL,
Clabels = rownames(Cmat), Cadj = par()$adj,
Ccex = par()$cex, Ccol = par()$col, Clty = par()$lty,
Clwd = par()$lwd,
chull.arg = FALSE, ccmex = par()$cex, cccol = par()$col,
clty = par()$lty, cclwd = par()$lwd,
spch = NULL, scex = par()$cex, scol = par()$col,
slabels = rownames(x2mat), ...)
```

Arguments

- `object`: Object of class "rvglm".
- `A`: Logical. Allow the plotting of A?

## Not run:

```r
lvplot(p2, sites = TRUE, spch = "*", scol = "darkgreen", scex = 1.5,
chull = TRUE, label = TRUE, Absolute = TRUE, ellipse = 140,
adj = -0.5, pcol = "blue", pex = 1.3, las = 1, Ccol = "orange",
C = TRUE, Cadj = c(-0.3, -0.3, 1), Clwd = 2, Ccex = 1.4,
main = paste("Contours at Abundance = 140 with",
"convex hull of the site scores"))
```

## End(Not run)

## Not run:

```r
var(latvar(p2))  # A diagonal matrix, i.e., uncorrelated latent vars
var(latvar(p2, varI.latvar = TRUE))  # Identity matrix
Tol(p2)[, 1:2]  # Identity matrix
Tol(p2, varI.latvar = TRUE)[, 1:2]  # A diagonal matrix
```

## End(Not run)
Logical. Allow the plotting of C? If TRUE then C is represented by arrows emanating from the origin.

scores Logical. Allow the plotting of the n scores? The scores are the values of the latent variables for each observation.

show.plot Logical. Plot it? If FALSE, no plot is produced and the matrix of scores (n latent variable values) is returned. If TRUE, the rank of object need not be 2.

groups A vector whose distinct values indicate which group the observation belongs to. By default, all the observations belong to a single group. Useful for the multinomial logit model (see multinomial).

gapC The gap between the end of the arrow and the text labelling of C, in latent variable units.

scaleA Numerical value that is multiplied by A, so that C is divided by this value.

xlab Caption for the x-axis. See par.

ylab Caption for the y-axis. See par.

A labels Character vector to label A. Must be of length M.

A adj Justification of text strings for labelling A. See the adj argument of par.

A cex Numeric. Character expansion of the labelling of A. See the cex argument of par.

A col Line color of the arrows representing C. See the col argument of par.

A pch Either an integer specifying a symbol or a single character to be used as the default in plotting points. See par. The pch argument can be of length M, the number of species.

C labels Character vector to label C. Must be of length p^2.

C adj Justification of text strings for labelling C. See the adj argument of par.

C cex Numeric. Character expansion of the labelling of C. See the cex argument of par.

C col Line color of the arrows representing C. See the col argument of par.

C lty Line type of the arrows representing C. See the lty argument of par.

C lwd Line width of the arrows representing C. See the lwd argument of par.

chull.arg Logical. Plot the convex hull of the scores? This is done for each group (see the group argument).

c cex Numeric. Character expansion of the labelling of the convex hull. See the cex argument of par.

c col Line color of the convex hull. See the col argument of par.

c lty Line type of the convex hull. See the lty argument of par.

c lwd Line width of the convex hull. See the lwd argument of par.

spch Either an integer specifying a symbol or a single character to be used as the default in plotting points. See par. The spch argument can be of length M, the number of species.

s cex Numeric. Character expansion of the labelling of the scores. See the cex argument of par.
scol Line color of the arrows representing \( C \). See the col argument of \texttt{par}.

\texttt{slabels} Character vector to label the scores. Must be of length \( n \).

\ldots Arguments passed into the \texttt{plot} function when setting up the entire plot. Useful arguments here include \texttt{xlim} and \texttt{ylim}.

**Details**

For RR-VGLMs, a \textit{biplot} and a \textit{latent variable} plot coincide. In general, many of the arguments starting with “A” refer to \( A \) (of length \( M \)), “C” to \( C \) (of length \( p^2 \)), “c” to the convex hull (of length \( \text{length(unique(groups))} \)), and “s” to scores (of length \( n \)).

As the result is a biplot, its interpretation is based on the inner product.

**Value**

The matrix of scores (\( n \) latent variable values) is returned regardless of whether a plot was produced or not.

**Note**

The functions \texttt{lvplot.rrvglm} and \texttt{biplot.rrvglm} are equivalent.

In the example below the predictor variables are centered, which is a good idea.

**Author(s)**

Thomas W. Yee

**References**


**See Also**

\texttt{lvplot}, \texttt{par}, \texttt{rrvglm}, \texttt{Coef.rrvglm}, \texttt{rrvglm.control}.

**Examples**

```r
nn <- nrow(pneumo)  # x1, x2 and x3 are some unrelated covariates
pneumo <- transform(pneumo, slet = scale(log(exposure.time)),
  x1 = rnorm(nn), x2 = rnorm(nn), x3 = rnorm(nn))
fit <- rrvglm(cbind(normal, mild, severe) ~ slet + x1 + x2 + x3,
  multinomial, data = pneumo, Rank = 2,
  Corner = FALSE, Uncorrel = TRUE)

# Not run:
lvplot(fit, chull = TRUE, scores = TRUE, clty = 2, ccol = “blue”,
  scol = “red”, Ccol = “darkgreen”, Clwd = 2, Ccex = 2,
  main = “Biplot of some fictitional data”)

# End(Not run)
```
**machinists**  

### Description

A small count data set involving 414 machinists from a three months study, of accidents around the end of WWI.

### Usage

```r
data(machinists)
```

### Format

A data frame with the following variables.

- **accidents**: The number of accidents
- **ofreq**: Observed frequency, i.e., the number of machinists with that many accidents

### Details

The data was collected over a period of three months. There were 414 machinists in total. Also, there were data collected over six months, but it is not given here.

### Source


### References


### See Also

- negbinomial
- poissonff

### Examples

```r
machinists
mean(with(machinists, rep(accidents, times = ofreq)))
var(with(machinists, rep(accidents, times = ofreq)))
```

```r
# Not run: barplot(with(machinists, ofreq),
# names.arg = as.character(with(machinists, accidents)),
# main = "Machinists accidents",
# col = "lightblue", las = 1,
# ylab = "Frequency", xlab = "accidents")
```

```r
## End(Not run)
```
Description

Density, cumulative distribution function, quantile function and random generation for the Makeham distribution.

Usage

dmakeham(x, scale = 1, shape, epsilon = 0, log = FALSE)
pmakeham(q, scale = 1, shape, epsilon = 0, lower.tail = TRUE, log.p = FALSE)
qmakeham(p, scale = 1, shape, epsilon = 0, lower.tail = TRUE, log.p = FALSE)
rmakeham(n, scale = 1, shape, epsilon = 0)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. Same as in runif.
log Logical. If log = TRUE then the logarithm of the density is returned.
lower.tail, log.p Same meaning as in pnorm or qnorm.

scale, shape positive scale and shape parameters.
epsilon another parameter. Must be non-negative. See below.

Details

See makeham for details. The default value of epsilon = 0 corresponds to the Gompertz distribution. The function pmakeham uses lambertW.

Value

dmakeham gives the density, pmakeham gives the cumulative distribution function, qmakeham gives the quantile function, and rmakeham generates random deviates.

Author(s)

T. W. Yee and Kai Huang

References

makeham

Makeham Distribution Family Function

Description

Maximum likelihood estimation of the 3-parameter Makeham distribution.

Usage

makeham(ls考察 = "loge", l考察 = "loge", 考察 = "loge", 考察 = NULL, 考察 = NULL, 考察 = NULL,
考察 = exp(-5:5), 考察 = exp(-5:5), 考察 = exp(-4:1),
考察EIM = 500, oim.mean = TRUE, zero = NULL, nowarning = FALSE)

Arguments

nowarning Logical. Suppress a warning? Ignored for VGAM 0.9-7 and higher.
考察, 考察, 考察 Parameter link functions applied to the shape parameter shape, scale parameter
考察, 考察, 考察 scale, and other parameter 考察. All parameters are treated as positive here
(cf. dm考察 allows 考察 = 0, etc.). See Links for more choices.
考察, 考察, 考察 Optional initial values. A NULL means a value is computed internally. A value
must be given for 考察 currently, and this is a sensitive parameter!

Examples

probs <- seq(0.01, 0.99, by = 0.01)
考察 <- exp(-1); 考察 <- exp(1); 考察 <- 考察 <- 考察(-1)
max(abs(考察(考察(考察 = 考察, 考察 = 考察, 考察 = 考察),
考察 = 考察, 考察 = 考察, 考察 = 考察) - 考察)) # Should be 0

## Not run: x <- seq(-0.1, 2.0, by = 0.01);
plot(x, 考察(x, 考察 = 考察, 考察 = 考察, 考察 = 考察), type = "l",
考察 = "Blue is density, orange is cumulative distribution function",
考察 = "Purple lines are the 10,20,...,90 percentiles",
考察 = "blue", las = 1, ylab = ")
abline(h = 0, col = "blue", lty = 2)
lines(x, 考察(x, 考察 = 考察, 考察 = 考察, 考察 = 考察), col = "orange")
probs <- seq(0.1, 0.9, by = 0.1)
考察 <- 考察(考察 = 考察, 考察 = 考察, 考察 = 考察)
lines(考察, 考察(x, 考察 = 考察, 考察 = 考察, 考察 = 考察),
考察 = "purple", lty = 3, type = "h")
考察(考察 = 考察, 考察 = 考察, 考察 = 考察) - 考察 # Should be all zero
abline(h = 考察, col = "purple", lty = 3)
## End(Not run)
gshape, gscale, gepsilon  
See CommonVGAMffArguments.

nsimeim, zero  See CommonVGAMffArguments. Argument probs.y is used only when imethod = 2.
oim.mean  To be currently ignored.

Details

The Makeham distribution, which adds another parameter to the Gompertz distribution, has cumulative distribution function

\[
F(x; \alpha, \beta, \varepsilon) = 1 - \exp\left\{-\varepsilon y + \frac{\alpha}{\beta} \left[1 - e^{\beta y}\right]\right\}
\]

which leads to a probability density function

\[
f(x; \alpha, \beta, \varepsilon) = \left[\varepsilon + \alpha e^{\beta x}\right] \exp\left\{-x \varepsilon + \frac{\alpha}{\beta} \left[1 - e^{\beta x}\right]\right\},
\]

for \(\alpha > 0, \beta > 0, \varepsilon \geq 0, x > 0\). Here, \(\beta\) is called the scale parameter scale, and \(\alpha\) is called a shape parameter. The moments for this distribution do not appear to be available in closed form.

Simulated Fisher scoring is used and multiple responses are handled.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

A lot of care is needed because this is a rather difficult distribution for parameter estimation, especially when the shape parameter is large relative to the scale parameter. If the self-starting initial values fail then try experimenting with the initial value arguments, especially iepsilon. Successful convergence depends on having very good initial values. More improvements could be made here. Also, monitor convergence by setting trace = TRUE.

A trick is to fit a gompertz distribution and use it for initial values; see below. However, this family function is currently numerically fraught.

Author(s)

T. W. Yee

See Also

dmakeham, gompertz, simulate.vlm.
Examples

```r
## Not run: set.seed(123)
mdat <- data.frame(x2 = runif(nn <- 1000))
mdat <- transform(mdata, etal = -1,
                  ceta1 = 1,
                  eeta1 = -2)
mdat <- transform(mdata, shape1 = exp(eta1),
                  scale1 = exp(ceta1),
                  epsil1 = exp(eeta1))
mdat <- transform(mdata,
                  y1 = rmakeham(nn, shape = shape1, scale = scale1, eps = epsil1))

# A trick is to fit a Gompertz distribution first
fit0 <- vglm(y1 - 1, gompertz, data = mdata, trace = TRUE)
fit1 <- vglm(y1 - 1, makeham, data = mdata,
            etastart = cbind(predict(fit0), log(0.1)), trace = TRUE)

coe(fit1, matrix = TRUE)
sum(fit1)
## End(Not run)
```

margeff

Marginal effects for the multinomial logit and cumulative models

Description

Marginal effects for the multinomial logit model and cumulative logit/probit/... models: the derivative of the fitted probabilities with respect to each explanatory variable.

Usage

`margeff(object, subset = NULL)`

Arguments

- `object` A `vglm` multinomial or cumulative object.
- `subset` Numerical or logical vector, denoting the required observation(s). Recycling is used if possible. The default means all observations.

Details

Computes the derivative of the fitted probabilities of a multinomial logit model or cumulative logit/probit/... model with respect to each explanatory variable.
Value

A $p$ by $M + 1$ by $n$ array, where $p$ is the number of explanatory variables and the (hopefully) nominal response has $M + 1$ levels, and there are $n$ observations.

If `is.numeric(subset)` and `length(subset) == 1` then a $p$ by $M + 1$ matrix is returned.

Warning

Care is needed in interpretation, e.g., the change is not universally accurate for a unit change in each explanatory variable because eventually the ‘new’ probabilities may become negative or greater than unity. Also, the ‘new’ probabilities will not sum to one.

This function is not applicable for models with data-dependent terms such as `bs` and `poly`. Also the function should not be applied to models with any terms that have generated more than one column of the LM model matrix, such as `bs` and `poly`. For such try using numerical methods such as finite-differences. The formula in object should comprise of simple terms of the form $\sim x_2 + x_3 + x_4$, etc.

Note

For `multinomial` this function should handle any value of `refLevel` and also any constraint matrices. However, it does not currently handle the `xij` or `form2` arguments, nor `vgam` objects.

For `multinomial` if `subset` is numeric then the function uses a for loop over the observations (slow). The default computations use vectorization; this uses more memory than a for loop but is faster.

Author(s)

T. W. Yee

See Also

`multinomial`, `cumulative`, `vglm`.

Examples

```r
# Not a good example for multinomial() because the response is ordinal!!
ii <- 3; hh <- 1/100
pneumo <- transform(pneumo, let = log(exposure.time))
fit <- vglm(cbind(normal, mild, severe) ~ let, multinomial, data = pneumo)
fit <- vglm(cbind(normal, mild, severe) ~ let,
            cumulative(reverse = TRUE, parallel = TRUE),
            data = pneumo)
fitted(fit)[ii, ]

mynewdata <- with(pneumo, data.frame(let = let[ii] + hh))
(newwp <- predict(fit, newdata = mynewdata, type = "response"))

# Compare the difference. Should be the same as hh --> 0.
round(digits = 3, (newwp - fitted(fit)[ii, ])/hh) # Finite-difference approxn
round(digits = 3, margeff(fit, subset = ii)["let",])
```
marital.nz

Description

Some marital data mainly from a large NZ company collected in the early 1990s.

Usage

data(marital.nz)

Format

A data frame with 6053 observations on the following 3 variables.

age  a numeric vector, age in years
ethnicity a factor with levels European Maori Other Polynesian. Only Europeans are included in the data set.
mstatus a factor with levels Divorced/Separated, Married/Partnered, Single, Widowed.

Details

This is a subset of a data set collected from a self-administered questionnaire administered in a large New Zealand workforce observational study conducted during 1992–3. The data were augmented by a second study consisting of retirees. The data can be considered a reasonable representation of the white male New Zealand population in the early 1990s.

Source

Clinical Trials Research Unit, University of Auckland, New Zealand.

References

See bmi.nz and chest.nz.

Examples

summary(marital.nz)
Description

Generic function for the *maximums* (maxima) of a model.

Usage

Max(object, ...)

Arguments

object An object for which the computation or extraction of a maximum (or maximums) is meaningful.

... Other arguments fed into the specific methods function of the model. Sometimes they are fed into the methods function for Coef.

Details

Different models can define a maximum in different ways. Many models have no such notion or definition.

Maximums occur in quadratic and additive ordination, e.g., CQO or CAO. For these models the maximum is the fitted value at the optimum. For quadratic ordination models there is a formula for the optimum but for additive ordination models the optimum must be searched for numerically. If it occurs on the boundary, then the optimum is undefined. For a valid optimum, the fitted value at the optimum is the maximum.

Value

The value returned depends specifically on the methods function invoked.

Author(s)

Thomas W. Yee

References


See Also

Max.qrrvglm, Tol, Opt.
Examples

```r
## Not run:
set.seed(111)  # This leads to the global solution
hsq <- scale(hs[1:6])  # Standardized environmental vars
pl <- cbind(Alopecce, Alopecune, Alopfabr, Arctlute, Arctperi,
            Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
            Trocerr, Zoraspin) ~
               WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + RefLux,
               quasipoissonff, Bestof = 2, data = hsq, Crowlpositive = FALSE)

Max(pl)

index <- 1:ncol(depvar(pl))
persp(pl, col = index, las = 1, llwd = 2)
abline(h = Max(pl), lty = 2, col = index)

## End(Not run)
```

Maxwell

The Maxwell Distribution

Description

Density, distribution function, quantile function and random generation for the Maxwell distribution.

Usage

```r
dmaxwell(x, rate, log = FALSE)
pmaxwell(q, rate, lower.tail = TRUE, log.p = FALSE)
qmaxwell(p, rate, lower.tail = TRUE, log.p = FALSE)
rmmaxwell(n, rate)
```

Arguments

- `x, q, p, n` Same as `Uniform`.
- `rate` the (rate) parameter.
- `log` Logical. If `log = TRUE` then the logarithm of the density is returned.
- `lower.tail, log.p` Same meaning as in `pnorm` or `qnorm`.

Details

See `maxwell`, the VGAM family function for estimating the (rate) parameter \( \alpha \) by maximum likelihood estimation, for the formula of the probability density function.
maxwell

maxwell

Value

dmaxwell gives the density, pmaxwell gives the distribution function, qmaxwell gives the quantile function, and rmaxwell generates random deviates.

Note

The Maxwell distribution is related to the Rayleigh distribution.

Author(s)

T. W. Yee and Kai Huang

References


See Also

maxwell, Rayleigh, rayleigh.

Examples

```r
# Not run: rate <- 3; x <- seq(-0.5, 3, length = 100)
plot(x, dmaxwell(x, rate = rate), type = "l", col = "blue", lty = 1, ylab = "",
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Purple lines are the 10, 20, ..., 90 percentiles")
abline(h = 0, col = "blue", lty = 2)
lines(x, pmaxwell(x, rate = rate), type = "l", col = "orange")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qmaxwell(probs, rate = rate)
lines(Q, dmaxwell(Q, rate), col = "purple", lty = 3, type = "h")
lines(Q, pmaxwell(Q, rate), col = "purple", lty = 3, type = "h")
abline(h = probs, col = "purple", lty = 3)
max(abs(pmaxwell(Q, rate) - probs)) # Should be zero
```

maxwell

Maxwell Distribution Family Function

Description

Estimating the parameter of the Maxwell distribution by maximum likelihood estimation.

Usage

maxwell(link = "loge", zero = NULL)
maxwell

Arguments

link, zero  Parameter link function applied to \( a \), which is called the parameter rate. See Links for more choices and information; a log link is the default because the parameter is positive. More information is at CommonVGAMffArguments.

Details

The Maxwell distribution, which is used in the area of thermodynamics, has a probability density function that can be written

\[
f(y; a) = \sqrt{\frac{2}{\pi a^{3/2}}} y^2 \exp(-0.5ay^2)
\]

for \( y > 0 \) and \( a > 0 \). The mean of \( Y \) is \( \sqrt{8/(a\pi)} \) (returned as the fitted values), and its variance is \((3\pi - 8)/(\pi a)\).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Note

Fisher-scoring and Newton-Raphson are the same here. A related distribution is the Rayleigh distribution. This VGAM family function handles multiple responses. This VGAM family function can be mimicked by poisson.points(ostatistic = 1.5, dimension = 2).

Author(s)

T. W. Yee

References


See Also

Maxwell, rayleigh, poisson.points.

Examples

mdata <- data.frame(y = rmaxwell(1000, rate = exp(2)))
fit <- vglm(y ~ 1, maxwell, data = mdata, trace = TRUE, crit = "coef")
coef(fit, matrix = TRUE)
Coef(fit)
Description


Usage

mccullagh89(ltheta = "rhobit", lnu = logoff(offset = 0.5),
       itheta = NULL, inu = NULL, zero = NULL)

Arguments

ltheta, lnu Link functions for the \( \theta \) and \( \nu \) parameters. See \textit{Links} for general information.
itheta, inu Numeric. Optional initial values for \( \theta \) and \( \nu \). The default is to internally compute them.
zero An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The default is none of them. If used, choose one value from the set \{1,2\}.

Details

The McCullagh (1989) distribution has density function

\[
f(y; \theta, \nu) = \frac{(1 - y^2)^{\nu - \frac{1}{2}}}{(1 - 2\theta y + \theta^2)^{\nu} \text{Beta}(\nu + \frac{1}{2}, \frac{1}{2})}
\]

where \(-1 < y < 1\) and \(-1 < \theta < 1\). This distribution is equation (1) in that paper. The parameter \( \nu \) satisfies \( \nu > -1/2 \), therefore the default is to use an log-offset link with offset equal to 0.5, i.e., \( \eta_2 = \log(\nu + 0.5) \). The mean is of \( Y \) is \( \nu \theta / (1 + \nu) \), and these are returned as the fitted values.

This distribution is related to the Leipnik distribution (see Johnson et al. (1995)), is related to ultraspherical functions, and under certain conditions, arises as exit distributions for Brownian motion. Fisher scoring is implemented here and it uses a diagonal matrix so the parameters are globally orthogonal in the Fisher information sense. McCullagh (1989) also states that, to some extent, \( \theta \) and \( \nu \) have the properties of a location parameter and a precision parameter, respectively.

Value

An object of class "\textit{vglmff}" (see \textit{vglmff-class}). The object is used by modelling functions such as \textit{vglm}, \textit{rrvglm} and \textit{vgam}.

Note

Convergence may be slow or fail unless the initial values are reasonably close. If a failure occurs, try assigning the argument \( \text{inu} \) and/or \( \text{itheta} \). Figure 1 of McCullagh (1989) gives a broad range of densities for different values of \( \theta \) and \( \nu \), and this could be consulted for obtaining reasonable initial values if all else fails.
**Description**

Melbourne daily maximum temperatures in degrees Celsius over the ten-year period 1981–1990.

**Usage**

```r
data(melbmaxtemp)
```

**Format**

A vector with 3650 observations.

**Details**

This is a time series data from Melbourne, Australia. It is commonly used to give a difficult quantile regression problem since the data is bimodal. That is, a hot day is likely to be followed by either an equally hot day or one much cooler. However, an independence assumption is typically made.

**References**

See Also

lms.bcn.

Examples

summary(melbmaxtemp)
## Not run: par(mfrow = c(1, 1), mar = c(5, 4, 0.2, 0.1) + 0.1, las = 1)
melb <- data.frame(tod = melbmaxtemp[-1],
                    yesterday = melbmaxtemp[-length(melbmaxtemp)]
plot(tod ~ yesterday, data = melb,
     xlab = "Yesterday's Max Temperature",
     ylab = "Today's Max Temperature", cex = 1.4, type = "n")
points(tod ~ yesterday, data = melb, pch = 0, cex = 0.50, col = "blue")
abline(a = 0, b = 1, lty = 3)
## End(Not run)

meplot

Mean Excess Plot

Description

Mean excess plot (also known as a mean residual life plot), a diagnostic plot for the generalized Pareto distribution (GPD).

Usage

meplot(object, ...)
meplot.default(y, main = "Mean Excess Plot",
               xlab = "Threshold", ylab = "Mean Excess", lty = c(2, 1:2),
               conf = 0.95, col = c("blue", "black", "blue"), type = "l", ...)
meplot.vlm(object, ...)

Arguments

y

A numerical vector. NAs etc. are not allowed.

main, xlab, ylab

Character. Overall title for the plot, and titles for the x- and y-axes.

lty

Line type. The second value is for the mean excess value, the first and third values are for the envelope surrounding the confidence interval.

conf

Confidence level. The default results in approximate 95 percent confidence intervals for each mean excess value.

col

Colour of the three lines.

type

Type of plot. The default means lines are joined between the mean excesses and also the upper and lower limits of the confidence intervals.

object

An object that inherits class "vlm", usually of class vglm-class or vgam-class.

...

Graphical argument passed into plot. See par for an exhaustive list. The arguments xlim and ylim are particularly useful.
Details

If \( Y \) has a GPD with scale parameter \( \sigma \) and shape parameter \( \xi < 1 \), and if \( y > 0 \), then

\[
E(Y - u|Y > u) = \frac{\sigma + \xi u}{1 - \xi}.
\]

It is a linear function in \( u \), the threshold. Note that \( Y - u \) is called the excess and values of \( Y \) greater than \( u \) are called exceedances. The empirical versions used by these functions is to use sample means to estimate the left hand side of the equation. Values of \( u \) in the plot are the values of \( y \) itself. If the plot is roughly a straight line then the GPD is a good fit; this plot can be used to select an appropriate threshold value. See gpd for more details. If the plot is flat then the data may be exponential, and if it is curved then it may be Weibull or gamma. There is often a lot of variance/fluuctuation at the RHS of the plot due to fewer observations.

The function meplot is generic, and meplot.default and meplot.vlm are some methods functions for mean excess plots.

Value

A list is returned invisibly with the following components.

- **threshold**: The x axis values.
- **meanExcess**: The y axis values. Each value is a sample mean minus a value \( u \).
- **plusminus**: The amount which is added or subtracted from the mean excess to give the confidence interval. The last value is a NA because it is based on one observation.

Note

The function is designed for speed and not accuracy, therefore huge data sets with extremely large values may cause failure (the function cumsum is used.) Ties may not be well handled.

Author(s)

T. W. Yee

References


See Also

gpd.
Examples

```r
## Not run: meplot(with(venice90, sealevel), las = 1) -> ii
names(ii)
abline(h = ii$meanExcess[1], col = "orange", lty = "dashed")

par(mfrow = c(2, 2))
for (ii in 1:4)
  meplot(rgpd(1000), col = c("orange", "blue", "orange"))
## End(Not run)
```

### micmen

**Michaelis-Menten Model**

**Description**

Fits a Michaelis-Menten nonlinear regression model.

**Usage**

```r
micmen(rpar = 0.001, divisor = 10, init1 = NULL, init2 = NULL,
imethod = 1, oim = TRUE, link1 = "identitylink", link2 = "identitylink",
firstDeriv = c("nsimEIM", "rpar"), probs.x = c(0.15, 0.85),
nsimEIM = 500, dispersion = 0, zero = NULL)
```

**Arguments**

- `rpar` Numeric. Initial positive ridge parameter. This is used to create positive-definite weight matrices.
- `divisor` Numerical. The divisor used to divide the ridge parameter at each iteration until it is very small but still positive. The value of `divisor` should be greater than one.
- `init1`, `init2` Numerical. Optional initial value for the first and second parameters, respectively. The default is to use a self-starting value.
- `link1`, `link2` Parameter link function applied to the first and second parameters, respectively. See `Links` for more choices.
- `firstDeriv` Character. Algorithm for computing the first derivatives and working weights. The first is the default.
- `imethod`, `probs.x` See `CommonVGAMffArguments` for more information.
- `nsimEIM` An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set `{1,2}`. A NULL means none. See `CommonVGAMffArguments` for more information.
Details

The Michaelis-Menten model is given by

\[ E(Y_i) = \frac{\theta_1 u_i}{\theta_2 + u_i} \]

where \( \theta_1 \) and \( \theta_2 \) are the two parameters.

The relationship between iteratively reweighted least squares and the Gauss-Newton algorithm is given in Wedderburn (1974). However, the algorithm used by this family function is different. Details are given at the Author’s web site.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

This function is not (nor could ever be) entirely reliable. Plotting the fitted function and monitoring convergence is recommended.

Note

The regressor values \( u_i \) are inputted as the RHS of the form argument. It should just be a simple term; no smart prediction is used. It should just a single vector, therefore omit the intercept term. The LHS of the formula form is ignored.

To predict the response at new values of \( u_i \) one must assign the @extra$Xm2 slot in the fitted object these values, e.g., see the example below.

Numerical problems may occur. If so, try setting some initial values for the parameters. In the future, several self-starting initial values will be implemented.

Author(s)

T. W. Yee

References


See Also

enzyme.
Examples

```r
fit <- vglm(velocity ~ 1, micmen, enzyme, trace = TRUE, crit = "coef",
            form2 = ~ conc - 1)
summary(fit)
```

```r
## Not run: plot(velocity ~ conc, enzyme, xlab = "concentration", las = 1,
##       col = "blue", main = "Michaelis-Menten equation for the enzyme data",
##       ylim = c(0, max(velocity)), xlim = c(0, max(conc)))
points(fitted(fit) ~ conc, enzyme, col = "red", pch = "+", cex = 1.5)
```

```r
# This predicts the response at a finer grid:
newenzyme <- data.frame(conc = seq(0, max(with(enzyme, conc)), len = 200))
fit@extra$Xm2 <- newenzyme$conc # This assignment is needed for prediction
lines(predict(fit, newenzyme, "response") ~ conc, newenzyme, col = "red")
## End(Not run)
```

mix2exp  

### Mixture of Two Exponential Distributions

**Description**

Estimates the three parameters of a mixture of two exponential distributions by maximum likelihood estimation.

**Usage**

```r
mix2exp(lphi = "logit", llambda = "loge", iphi = 0.5, il1 = NULL,
         il2 = NULL, qmu = c(0.8, 0.2), nsimEIM = 100, zero = 1)
```

**Arguments**

- `lphi`, `llambda` Link functions for the parameters \( \phi \) and \( \lambda \). The latter is the rate parameter and note that the mean of an ordinary exponential distribution is \( 1/\lambda \). See [Links](#) for more choices.
- `iphi`, `il1`, `il2` Initial value for \( \phi \), and optional initial value for \( \lambda_1 \) and \( \lambda_2 \). The last two have values that must be positive. The default is to compute initial values internally using the argument `qmu`.
- `qmu` Vector with two values giving the probabilities relating to the sample quantiles for obtaining initial values for \( \lambda_1 \) and \( \lambda_2 \). The two values are fed in as the `probs` argument into `quantile`.
- `nsimEIM`, `zero` See [CommonVGAMffArguments](#).
Details

The probability density function can be loosely written as

\[ f(y) = \phi \text{Exponential}(\lambda_1) + (1 - \phi) \text{Exponential}(\lambda_2) \]

where \( \phi \) is the probability an observation belongs to the first group, and \( y > 0 \). The parameter \( \phi \) satisfies \( 0 < \phi < 1 \). The mean of \( Y \) is \( \phi/\lambda_1 + (1 - \phi)/\lambda_2 \) and this is returned as the fitted values. By default, the three linear/additive predictors are \((\logit(\phi), \log(\lambda_1), \log(\lambda_2))^T\).

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

This VGAM family function requires care for a successful application. In particular, good initial values are required because of the presence of local solutions. Therefore running this function with several different combinations of arguments such as iphi, il1, il2, qmu is highly recommended. Graphical methods such as hist can be used as an aid.

This VGAM family function is experimental and should be used with care.

Note

Fitting this model successfully to data can be difficult due to local solutions, uniqueness problems and ill-conditioned data. It pays to fit the model several times with different initial values and check that the best fit looks reasonable. Plotting the results is recommended. This function works better as \( \lambda_1 \) and \( \lambda_2 \) become more different. The default control argument trace = TRUE is to encourage monitoring convergence.

Author(s)

T. W. Yee

See Also

rexp, exponential, mix2poisson.

Examples

```r
## Not run: lambda1 <- exp(1); lambda2 <- exp(3)
(phi <- logit(-1, inverse = TRUE))
mdata <- data.frame(y1 = rexp(nn <- 1000, lambda1))
mdata <- transform(mdata, y2 = rexp(nn, lambda2))
mdata <- transform(mdata, Y = ifelse(runif(nn) < phi, y1, y2))
fit <- vglm(Y ~ 1, mix2exp, data = mdata, trace = TRUE)
coef(fit, matrix = TRUE)

# Compare the results with the truth
round(rbind('Estimated' = Coef(fit),
```
mix2normal

Mixture of Two Univariate Normal Distributions

Description

Estimates the five parameters of a mixture of two univariate normal distributions by maximum likelihood estimation.

Usage

mix2normal(lphi = "logit", lmu = "identitylink", lsd = "loge",
iphi = 0.5, imu1 = NULL, imu2 = NULL, isd1 = NULL, isd2 = NULL,
qmu = c(0.2, 0.8), eq.sd = TRUE, nsimEIM = 100, zero = 1)

Arguments

lphi, lmu, lsd Link functions for the parameters $\phi$, $\mu$, and $\sigma$. See Links for more choices.
iphi Initial value for $\phi$, whose value must lie between 0 and 1.
imu1, imu2 Optional initial value for $\mu_1$ and $\mu_2$. The default is to compute initial values internally using the argument qmu.
isd1, isd2 Optional initial value for $\sigma_1$ and $\sigma_2$. The default is to compute initial values internally based on the argument qmu. Currently these are not great, therefore using these arguments where practical is a good idea.
qmu Vector with two values giving the probabilities relating to the sample quantiles for obtaining initial values for $\mu_1$ and $\mu_2$. The two values are fed in as the probs argument into quantile.
eq.sd Logical indicating whether the two standard deviations should be constrained to be equal. If TRUE then the appropriate constraint matrices will be used.
nsimEIM See CommonVGAMffArguments.
zero An integer specifying which linear/additive predictor is modelled as intercepts only. If given, the value or values must be from the set $\{1, 2, \ldots, 5\}$. The default is the first one only, meaning $\phi$ is a single parameter even when there are explanatory variables. Set zero = NULL to model all linear/additive predictors as functions of the explanatory variables. See CommonVGAMffArguments for more information.
Details

The probability density function can be loosely written as

\[ f(y) = \phi N(\mu_1, \sigma_1) + (1 - \phi) N(\mu_2, \sigma_2) \]

where \( \phi \) is the probability an observation belongs to the first group. The parameters \( \mu_1 \) and \( \mu_2 \) are the means, and \( \sigma_1 \) and \( \sigma_2 \) are the standard deviations. The parameter \( \phi \) satisfies \( 0 < \phi < 1 \). The mean of \( Y \) is \( \phi \mu_1 + (1 - \phi) \mu_2 \) and this is returned as the fitted values. By default, the five linear/additive predictors are \((\text{logit}(\phi), \mu_1, \text{log}(\sigma_1), \mu_2, \text{log}(\sigma_2))^T\). If eq. sd = TRUE then \( \sigma_1 = \sigma_2 \) is enforced.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

Numerical problems can occur and half-stepping is not uncommon. If failure to converge occurs, try inputting better initial values, e.g., by using iphi, qmu, imu1, imu2, isd1, isd2, etc.

This VGAM family function is experimental and should be used with care.

Note

Fitting this model successfully to data can be difficult due to numerical problems and ill-conditioned data. It pays to fit the model several times with different initial values and check that the best fit looks reasonable. Plotting the results is recommended. This function works better as \( \mu_1 \) and \( \mu_2 \) become more different.

Convergence can be slow, especially when the two component distributions are not well separated. The default control argument trace = TRUE is to encourage monitoring convergence. Having eq. sd = TRUE often makes the overall optimization problem easier.

Author(s)

T. W. Yee

References


See Also

uninormal, Normal, mix2poisson.
Examples

```r
# Not run: mu1 <- 99; mu2 <- 150; nn <- 1000
sd1 <- sd2 <- exp(3)
(phi <- logit(-1, inverse = TRUE))
mdata <- data.frame(y = ifelse(runif(nn) < phi, rnorm(nn, mu1, sd1),
                     rnorm(nn, mu2, sd2)))
fit <- vglm(y ~ 1, mix2normal(eq.sd = TRUE), data = mdata)

# Compare the results
cfit <- coef(fit)
round(rbind('Estimated' = c(logit(cfit[1], inverse = TRUE),
            cfit[2], exp(cfit[3]), cfit[4]),
            'Truth' = c(phi, mu1, sd1, mu2)), digits = 2)

# Plot the results
xx <- with(mdata, seq(min(y), max(y), len = 200))
plot(xx, (1-phi) * dnorm(xx, mu2, sd2), type = "l", xlab = "y",
     main = "Orange = estimate, blue = truth",
     col = "blue", ylab = "Density")
phi.est <- logit(coef(fit)[1], inverse = TRUE)
sd.est <- exp(coef(fit)[3])
lines(xx, phi.est*dnorm(xx, mu1, sd1), col = "blue")
lines(xx, phi.est*dnorm(xx, Coef(fit)[2], sd.est), col = "orange")
lines(xx, (1-phi.est) * dnorm(xx, Coef(fit)[4], sd.est), col = "orange")
abline(v = Coef(fit)[c(2,4)], lty = 2, col = "orange")
abline(v = c(mu1, mu2), lty = 2, col = "blue")
```

# End(Not run)

mix2poisson

---

Mixture of Two Poisson Distributions

Description

Estimates the three parameters of a mixture of two Poisson distributions by maximum likelihood estimation.

Usage

```r
mix2poisson(lphi = "logit", llambda = "loge",
            iphi = 0.5, il1 = NULL, il2 = NULL,
            qmu = c(0.2, 0.8), nsimEIM = 100, zero = 1)
```

Arguments

- `lphi`, `llambda`: Link functions for the parameter $\phi$ and $\lambda$. See Links for more choices.
- `iphi`: Initial value for $\phi$, whose value must lie between 0 and 1.
- `il1`, `il2`: Optional initial value for $\lambda_1$ and $\lambda_2$. These values must be positive. The default is to compute initial values internally using the argument `qmu`. 
qmu
Vector with two values giving the probabilities relating to the sample quantiles for obtaining initial values for \( \lambda_1 \) and \( \lambda_2 \). The two values are fed in as the probs argument into quantile.

nsimEIM, zero  See CommonVGAMffArguments.

Details
The probability function can be loosely written as

\[
P(Y = y) = \phi \text{Poisson}(\lambda_1) + (1 - \phi) \text{Poisson}(\lambda_2)
\]

where \( \phi \) is the probability an observation belongs to the first group, and \( y = 0, 1, 2, \ldots \). The mean of \( Y \) is \( \phi \lambda_1 + (1 - \phi) \lambda_2 \) and this is returned as the fitted values. By default, the three linear/additive predictors are \((\text{logit}(\phi), \log(\lambda_1), \log(\lambda_2))\)^T.

Value
An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning
This VGAM family function requires care for a successful application. In particular, good initial values are required because of the presence of local solutions. Therefore running this function with several different combinations of arguments such as iphi, i11, i12, qmu is highly recommended. Graphical methods such as hist can be used as an aid.

With grouped data (i.e., using the weights argument) one has to use a large value of nsimEIM; see the example below.

This VGAM family function is experimental and should be used with care.

Note
The response must be integer-valued since dpois is invoked.

Fitting this model successfully to data can be difficult due to local solutions and ill-conditioned data. It pays to fit the model several times with different initial values, and check that the best fit looks reasonable. Plotting the results is recommended. This function works better as \( \lambda_1 \) and \( \lambda_2 \) become more different. The default control argument trace = TRUE is to encourage monitoring convergence.

Author(s)
T. W. Yee

See Also
rpois, poissonff, mix2normal.
Examples

```r
## Not run: # Example 1: simulated data
nn <- 1000
mu1 <- exp(2.5) # Also known as lambda1
mu2 <- exp(3)
(phi <- logit(-0.5, inverse = TRUE))
ndata <- data.frame(y = rpois(nn, ifelse(runif(nn) < phi, mu1, mu2)))

mfit <- vglm(y ~ 1, mixpoisson, data = nndata)

# Compare the results with the truth
round(rbind("Estimated" = Coef(mfit), 'Truth' = c(phi, mu1, mu2)), digits = 2)

# Example 2: London Times data (Lange, 1997, p.31)
lndata1 <- data.frame(deaths = 0:9,
freq = c(162, 267, 271, 185, 111, 61, 27, 8, 3, 1))
lndata2 <- data.frame(y = with(lndata1, rep(deaths, freq)))

# Usually this does not work well unless nsimEIM is large
Mfit <- vglm(deaths ~ 1, weight = freq, data = ndata1,
mixpoisson(iph1 = 0.3, il1 = 1, il2 = 2.5, nsimEIM = 5000))

# This works better in general
Mfit <- vglm(y ~ 1, mixpoisson(iph1 = 0.3, il1 = 1, il2 = 2.5), data = ndata2)
coef(Mfit, matrix = TRUE)

## End(Not run)
```

### The MNSs Blood Group System

**Description**

Estimates the three independent parameters of the the MNSs blood group system.

**Usage**

```r
MNSs(link = "logit", ims = NULL, imS = NULL, inS = NULL)
```
Arguments

- **link**: Link function applied to the three parameters. See Links for more choices.
- **ims, ims, inS**: Optional initial value for mS, ms and nS respectively. A NULL means they are computed internally.

Details

There are three independent parameters: m_S, m_s, n_S, say, so that \( n_s = 1 - m_S - m_s - n_S \). We let the eta vector (transposed) be \( (g(m_S), g(m_s), g(n_S)) \) where g is the link function.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

The input can be a 6-column matrix of counts, where the columns are MS, Ms, MNS, MNs, NS, Ns (in order). Alternatively, the input can be a 6-column matrix of proportions (so each row adds to 1) and the weights argument is used to specify the total number of counts for each row.

Author(s)

T. W. Yee

References


See Also

AA, Aa, aa, AB, Ab, aB, ab, ABO, A1A2A3.

Examples

```r
# Order matters only:
y <- cbind(MS = 295, Ms = 107, MNS = 379, MNs = 322, NS = 102, Ns = 214)
fit <- vglm(y ~ 1, MNSs("logit", .25, .28, .08), trace = TRUE)
fit <- vglm(y ~ 1, MNSs(link = "logit"), trace = TRUE, crit = "coef")
Coef(fit)
rbind(y, sum(y)*fitted(fit))
sqrt(diag(vcov(fit)))
```
Construct the Model Frame of a VLM Object

Description

This function returns a data.frame with the variables. It is applied to an object which inherits from class "vlm" (e.g., a fitted model of class "vglm").

Usage

model.framevlm(object, setupsmart = TRUE, wrapupsmart = TRUE, ...)

Arguments

object a model object from the VGAM R package that inherits from a vector linear model (VLM), e.g., a model of class "vglm".

... further arguments such as data, na.action, subset. See model.frame for more information on these.

setupsmart, wrapupsmart
Logical. Arguments to determine whether to use smart prediction.

Details

Since object is an object which inherits from class "vlm" (e.g., a fitted model of class "vglm"), the method will either returned the saved model frame used when fitting the model (if any, selected by argument model = TRUE) or pass the call used when fitting on to the default method.

This code implements smart prediction (see smartpred).

Value

A data.frame containing the variables used in the object plus those specified in ....

References


See Also

model.frame, model.matrixvlm, predictvlm, smartpred.
Examples

# Illustrates smart prediction
pneumo <- transform(pneumo, let = log(exposure.time))
fit <- vglm(cbind(normal, mild, severe) ~ poly(c(scale(let)), 2),
           multinomial, data = pneumo, trace = TRUE, x = FALSE)
class(fit)

check1 <- head(model.frame(fit))
check1
cHECK2 <- model.frame(fit, data = head(pneumo))
cHECK2
all.equal(unlist(check1), unlist(check2)) # Should be TRUE

q0 <- head(predict(fit))
q1 <- head(predict(fit, newdata = pneumo))
q2 <- predict(fit, newdata = head(pneumo))
all.equal(q0, q1) # Should be TRUE
all.equal(q1, q2) # Should be TRUE

model.matrixvlm Construct the Design Matrix of a VLM Object

Description

Creates a design matrix. Two types can be returned: a large one (class "vlm") or one that inherits from this such as "vglm") or a small one (such as returned if it were of class "lm").

Usage

model.matrixvlm(object, type = c("vlm", "lm", "lm2", "bothlmlm2"),
linpred.index = NULL, ...)

Arguments

object an object of a class that inherits from the vector linear model (VLM).
type Type of design matrix returned. The first is the default. The value "vlm" is the VLM model matrix corresponding to the formula argument. The value "lm" is the LM model matrix corresponding to the formula argument. The value "lm2" is the second (LM) model matrix corresponding to the form2 argument. The value "bothlmlm2" means both LM and VLM model matrices.
linpred.index Single integer. The index for a linear/additive predictor, it must have a value from the set 1:M, and type = "lm" must be assigned. Then it returns a subset of the VLM matrix corresponding to the linpred.indexth linear/additive predictor; this is a LM-type matrix.

... further arguments passed to or from other methods. These include data (which is a data frame created with model.framevlm), contrasts.arg, and xlev. See model.matrix for more information.
Details

This function creates a design matrix from object. This can be a small LM object or a big VLM object (default). The latter is constructed from the former and the constraint matrices.

This code implements smart prediction (see smartpred).

Value

The design matrix for a regression model with the specified formula and data. If type = "both1lm1m2" then a list is returned with components "X" and "Xm2".

References


See Also

`model.matrix`, `model.frame.vlm`, `predict.vglm`, `smartpred`.

Examples

```r
# Illustrates smart prediction
pneumo <- transform(pneumo, let = log(exposure.time))
fit <- vglm(cbind(normal, mild, severe) ~ poly(c(scale(let)), 2),
            multinomial, data = pneumo, trace = TRUE, x = FALSE)
class(fit)
fit$x  # Not saved on the object
model.matrix(fit)
model.matrix(fit, linpred.index = 1, type = "lm")
model.matrix(fit, linpred.index = 2, type = "lm")

(Check1 <- head(model.matrix(fit, type = "lm")))
(Check2 <- model.matrix(fit, data = head(pneumo), type = "lm"))
all.equal(c(Check1), c(Check2))

q0 <- head(predict(fit))
q1 <- head(predict(fit, newdata = pneumo))
q2 <- predict(fit, newdata = head(pneumo))
all.equal(q0, q1)  # Should be TRUE
all.equal(q1, q2)  # Should be TRUE
```
**moffset**

**Matrix Offset**

**Description**

Modify a matrix by shifting successive elements.

**Usage**

\[
\text{moffset(mat, roffset = 0, coffset = 0, postfix = '',}
\]

\[\text{rprefix = "Row.", cprefix = "Col."})\]

**Arguments**

- **mat**
  Data frame or matrix. This ought to have at least three rows and three columns. The elements are shifted in the order of \(c(mat)\), i.e., going down successive columns, as the columns go from left to right. Wrapping of values is done.

- **roffset, coffset**
  Numeric or character. If numeric, the amount of shift (offset) for each row and column. The default is no change to \(mat\). If character, the offset is computed by matching with the row or column names. For example, for the \texttt{alcoff}, put \(roffset = "6"\) means that we make an effective day's dataset start from 6:00 am, and this wraps around to include midnight to 05.59 am on the next day.

- **postfix**
  Character. Modified rows and columns are renamed by pasting this argument to the end of each name. The default is no change.

- **rprefix, cprefix**
  Same as \texttt{rcim}.

**Details**

This function allows a matrix to be rearranged so that element \((roffset + 1, coffset + 1)\) becomes the \((1, 1)\) element. The elements are assumed to be ordered in the same way as the elements of \(c(mat)\).

This function is applicable to, e.g., \texttt{alcoff}, where it is useful to define the effective day as starting at some other hour than midnight, e.g., 6.00am. This is because partying on Friday night continues on into Saturday morning, therefore it is more interpretable to use the effective day when considering a daily effect.

This is a data preprocessing function for \texttt{rcim} and \texttt{plotrcim}. The differences between \texttt{Rcim} and \texttt{moffset} is that \texttt{Rcim} only reorders the level of the rows and columns so that the data is shifted but not moved. That is, a value in one row stays in that row, and ditto for column. But in \texttt{moffset} values in one column can be moved to a previous column. See the examples below.

**Value**

A matrix of the same dimensional as its input.
Note
The input mat should have row names and column names.

Author(s)
T. W. Yee, Alfian F. Hadi.

See Also
Rcim, rcim.plotrcim0, alcoff, crashi.

Examples

moffset(alcoff, 3, 2, "*")  # Some day's data is moved to previous day.
Rcim(alcoff, 3 + 1, 2 + 1)  # Data does not move as much.
alcoff  # Original data
moffset(alcoff, 3, 2, "*") - Rcim(alcoff, 3+1, 2+1)  # Note the differences

# An 'effective day' data set:
alcoff.e <- moffset(alcoff, roffset = "6", postfix = "*")
fit.o <- rcim(alcoff)  # default baselines are first row and col
fit.e <- rcim(alcoff.e)  # default baselines are first row and col

## Not run: par(mfrow = c(2, 2), mar = c(9, 4, 2, 1))
plot(fit.o, rsub = "Not very interpretable", csub = "Not very interpretable")
plot(fit.e, rsub = "More interpretable", csub = "More interpretable")

## End(Not run)

# Some checking
all.equal(moffset(alcoff), alcoff)  # Should be no change
moffset(alcoff, 1, 1, "*")
moffset(alcoff, 2, 3, "*")
moffset(alcoff, 1, 0, "*")
moffset(alcoff, 0, 1, "*")
moffset(alcoff, "6", "Mon", "*")  # This one is good

# Customise row and column baselines
fit2 <- rcim(Rcim(alcoff.e, rbaseline = "11", cbaseline = "Mon*"))

——-

multilogit  

Multi-logit Link Function

Description
Computes the multilogit transformation, including its inverse and the first two derivatives.
**Usage**

```r
multilogit(theta, reflevel = "last", M = NULL, whitespace = FALSE,
          bvalue = NULL, inverse = FALSE, deriv = 0,
          short = TRUE, tag = FALSE)
```

**Arguments**

- `theta` Numeric or character. See below for further details.
- `reflevel, M, whitespace` See `multinomial`.
- `bvalue` See `Links`.
- `inverse, deriv, short, tag` Details at `Links`.

**Details**

The `multilogit()` link function is a generalization of the `logit` link to `M` levels/classes. It forms the basis of the `multinomial` logit model. It is sometimes called the multi-logit link or the multinomial logit link. When its inverse function is computed it returns values which are positive and add to unity.

**Value**

For `multilogit` with `deriv = 0`, the multilogit of `theta`, i.e., `log(theta[, j]/theta[, M+1])` when `inverse = FALSE`, and if `inverse = TRUE` then `exp(theta[, j])/(1+rowSums(exp(theta)))`.

For `deriv = 1`, then the function returns `d theta / d eta` as a function of `theta` if `inverse = FALSE`, else if `inverse = TRUE` then it returns the reciprocal.

Here, all logarithms are natural logarithms, i.e., to base `e`.

**Note**

Numerical instability may occur when `theta` is close to 1 or 0 (for `multilogit`). One way of overcoming this is to use, e.g., `bvalue`. Currently `care.exp()` is used to avoid NAs being returned if the probability is too close to 1.

**Author(s)**

Thomas W. Yee

**References**


**See Also**

`Links, multinomial, logit, normal.vcm, CommonVGAMffArguments.`
Examples

```r
cpyemo <- transform(pnego, let = log(exposure.time))
fit <- vglm(cbind(normal, mild, severe) ~ let,
  multinomial, trace = TRUE, data = pneumo)  # For illustration only!
fitted(fit)
predict(fit)

multilogit(fitted(fit))
multilogit(fitted(fit)) - predict(fit)  # Should be all 0s

multilogit(predict(fit), inverse = TRUE)  # rowSums() add to unity
multilogit(predict(fit), inverse = TRUE, reflevel = 1)  # For illustration only
multilogit(predict(fit), inverse = TRUE) - fitted(fit)  # Should be all 0s

multilogit(fitted(fit), deriv = 1)
multilogit(fitted(fit), deriv = 2)
```

**Multinomial Logit Model**

### Description

Fits a multinomial logit model to a (preferably unordered) factor response.

### Usage

```r
multinomial(zero = NULL, parallel = FALSE, nointercept = NULL,
  refLevel = "last", whitespace = FALSE)
```

### Arguments

- **zero**: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. Any values must be from the set \{1,2,...,M\}. The default value means none are modelled as intercept-only terms. See CommonVGAMffArguments for more information.

- **parallel**: A logical, or formula specifying which terms have equal/unequal coefficients.

- **nointercept, whitespace**: See CommonVGAMffArguments for more details.

- **refLevel**: Either a single positive integer or a value of the factor. If an integer then it specifies which column of the response matrix is the reference or baseline level. The default is the last one (the \((M + 1)\)th one). If used, this argument will be often assigned the value 1. If inputted as a value of a factor then beware of missing values of certain levels of the factor (drop.unused.levels = TRUE or drop.unused.levels = FALSE). See the example below.
Details

In this help file the response $Y$ is assumed to be a factor with unordered values $1, 2, \ldots, M + 1$, so that $M$ is the number of linear/additive predictors $\eta_j$.

The default model can be written

$$
\eta_j = \log(P[Y = j]/P[Y = M + 1])
$$

where $\eta_j$ is the $j$th linear/additive predictor. Here, $j = 1, \ldots, M$, and $\eta_{M+1}$ is 0 by definition. That is, the last level of the factor, or last column of the response matrix, is taken as the reference level or baseline—this is for identifiability of the parameters. The reference or baseline level can be changed with the refLevel argument.

In almost all the literature, the constraint matrices associated with this family of models are known. For example, setting `parallel = TRUE` will make all constraint matrices (except for the intercept) equal to a vector of $M$ 1’s. If the constraint matrices are unknown and to be estimated, then this can be achieved by fitting the model as a reduced-rank vector generalized linear model (RR-VGLM; see `rrvglm`). In particular, a multinomial logit model with unknown constraint matrices is known as a stereotyped model (Anderson, 1984), and can be fitted with `rrvglm`.

Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Warning

No check is made to verify that the response is nominal. See `CommonVGAMffArguments` for more warnings.

Note

The response should be either a matrix of counts (with row sums that are all positive), or a factor. In both cases, the $y$ slot returned by `vglm/vgam/rrvglm` is the matrix of sample proportions.

The multinomial logit model is more appropriate for a nominal (unordered) factor response than for an ordinal (ordered) factor response. Models more suited for the latter include those based on cumulative probabilities, e.g., `cumulative`.

multinomial is prone to numerical difficulties if the groups are separable and/or the fitted probabilities are close to 0 or 1. The fitted values returned are estimates of the probabilities $P[Y = j]$ for $j = 1, \ldots, M + 1$. See `safeBinaryRegression` for the logistic regression case.

Here is an example of the usage of the `parallel` argument. If there are covariates $x_2$, $x_3$ and $x_4$, then `parallel = TRUE ~ x2 + x3 - 1` and `parallel = FALSE ~ x4` are equivalent. This would constrain the regression coefficients for $x_2$ and $x_3$ to be equal; those of the intercepts and $x_4$ would be different.

In Example 4 below, a conditional logit model is fitted to an artificial data set that explores how cost and travel time affect people’s decision about how to travel to work. Walking is the baseline group. The variable `Cost.car` is the difference between the cost of travel to work by car and walking, etc. The variable `Time.car` is the difference between the travel duration/time to work by car and walking, etc. For other details about the `xij` argument see `vglm.control` and `fill`.
The `multinom` function in the `nnet` package uses the first level of the factor as baseline, whereas the last level of the factor is used here. Consequently the estimated regression coefficients differ.

Author(s)

Thomas W. Yee

References


See Also

`margeff, cumulative, acat, cratio, sratio, dirichlet, dirmultinomial, rrvglm, fillQ, Multinomial, multilogit, iris` The author’s homepage has further documentation about categorical data analysis using VGAM.

Examples

```r
# Example 1: fit a multinomial logit model to Edgar Anderson's iris data
data(iris)
## Not run:
 fit <- vglm(Species ~ ., multinomial, iris)
coef(fit, matrix = TRUE)
## End(Not run)

# Example 2a: a simple example
ycounts <- t(rmultinom(10, size = 20, prob = c(0.1, 0.2, 0.8))) # Counts
fit <- vglm(ycounts ~ 1, multinomial)
head(fitted(fit))  # Proportions
fit@prior.weights # NOT recommended for extraction of prior weights
weights(fit, type = "prior", matrix = FALSE) # The better method
depvar(fit) # Sample proportions; same as fit@y
constraints(fit) # Constraint matrices

# Example 2b: Different reference level used as the baseline
fit2 <- vglm(ycounts ~ 1, multinomial(refLevel = 2))
coef(fit2, matrix = TRUE)
```
multinomial

```
coeff(fit, matrix = TRUE) # Easy to reconcile this output with fit2

# Example 3: The response is a factor.
nn <- 10
dframe3 <- data.frame(yfactor = gl(3, nn, labels = c("Control", "Trt1", "Trt2")),
                      x2 = rnorm(3 * nn))
myreflevel <- with(dframe3, yfactor[12])
fit3a <- vglm(yfactor ~ x2, multinomial(reflevel = myreflevel), dframe3)
fit3b <- vglm(yfactor ~ x2, multinomial(reflevel = 2), dframe3)
coeff(fit3a, matrix = TRUE) # "Treatment1" is the reference level
coeff(fit3b, matrix = TRUE) # "Treatment1" is the reference level
margeff(fit3b)

# Example 4: Fit a rank-1 stereotype model
fit4 <- rrvglm(Country ~ Width + Height + HP, multinomial, data = car.all)
coeff(fit4) # Contains the C matrix
constraints(fit4)$HP # The A matrix
coeff(fit4, matrix = TRUE) # The B matrix
Coeff(fit4)$C # The C matrix
concoef(fit4) # Better to get the C matrix this way
Coeff(fit4)$A # The A matrix
svd(coeff(fit4, matrix = TRUE)[-1, , ])$d # This has rank 1; = C %*% t(A)
# Classification (but watch out for nas in some of the variables):
apply(fitted(fit4), 1, which.max) # Classification
colnames(fitted(fit4))# apply(fitted(fit4), 1, which.max) # Classification
apply(predict(fit4, car.all, type = "response"), 1, which.max) # Ditto

# Example 5: The use of the xij argument (aka conditional logit model)
set.seed(111)
nn <- 100 # Number of people who travel to work
M <- 3 # There are M+1 models of transport to go to work
ycounts <- matrix(0, nn, M+1)
ycounts[cbind(1:nn, sample(x = M+1, size = nn, replace = TRUE))] <- 1
dimnames(ycounts) <- list(NULL, c("bus","train","car","walk"))
gotowork <- data.frame(cost.bus = rnorm(nn), time.bus = rnorm(nn),
                       cost.train = rnorm(nn), time.train = rnorm(nn),
                       cost.car = rnorm(nn), time.car = rnorm(nn),
                       cost.walk = rnorm(nn), time.walk = rnorm(nn))
gotowork <- round(gotowork, digits = 2) # For convenience
gotowork <- transform(gotowork, 
Cost.bus = cost.bus - cost.walk,
Cost.car = cost.car - cost.walk,
Cost.train = cost.train - cost.walk,
Cost = cost.train - cost.walk, # for labelling
Time.bus = time.bus - time.walk,
Time.car = time.car - time.walk,
Time.train = time.train - time.walk,
Time = time.train - time.walk) # for labelling
fit <- vglm(ycounts ~ Cost + Time,
            multinomial(parall = TRUE ~ Cost + Time - 1),
xij = list(Cost ~ Cost.bus + Cost.train + Cost.car,
                      Time ~ Time.bus + Time.train + Time.car),
```

form2 = - Cost + Cost.bus + Cost.train + Cost.car +
        Time + Time.bus + Time.train + Time.car,
data = gotowork, trace = TRUE)
head(model.matrix(fit, type = "lm")) # LM model matrix
head(model.matrix(fit, type = "vlm")) # Big VLM model matrix
coef(fit)
coef(fit, matrix = TRUE)
constraints(fit)
summary(fit)
max(abs(predict(fit) - predict(fit, new = gotowork))) # Should be 0

Nakagami Distribution

Description

Density, cumulative distribution function, quantile function and random generation for the Nakagami distribution.

Usage

dnaka(x, scale = 1, shape, log = FALSE)
pnaka(q, scale = 1, shape, lower.tail = TRUE, log.p = FALSE)
qnaka(p, scale = 1, shape, ...)
rnaka(n, scale = 1, shape, Smallno = 1.0e-6)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. Same as in runif.
scale, shape arguments for the parameters of the distribution. See nakagami for more details. For rnaka, arguments shape and scale must be of length 1.
Smallno Numeric, a small value used by the rejection method for determining the upper limit of the distribution. That is, pnaka(U) > 1-Smallno where U is the upper limit.
... Arguments that can be passed into uniroot.
log Logical. If log = TRUE then the logarithm of the density is returned.
lower.tail, log.p Same meaning as in pnorm or qnorm.

Details

See nakagami for more details.
nakagami

Value
dnaka gives the density, pnaka gives the cumulative distribution function, qnaka gives the quantile function, and rnaka generates random deviates.

Author(s)
T. W. Yee and Kai Huang

See Also
nakagami.

Examples

```r
# Not run: x <- seq(0, 3.2, len = 200)
plot(x, dgamma(x, shape = 1), type = "n", col = "black", ylab = "",
     ylim = c(0,1.5), main = "dnaka(x, shape = shape)"
) lines(x, dnaka(x, shape = 1), col = "orange")
lines(x, dnaka(x, shape = 2), col = "blue")
lines(x, dnaka(x, shape = 3), col = "green")
legend(2, 1.0, col = c("orange","blue","green"), lty = rep(1, len = 3),
       legend = paste("shape =", c(1, 2, 3)))

plot(x, pnorm(x), type = "n", col = "black", ylab = "",
     ylim = 0:1, main = "pnaka(x, shape = shape)"
) lines(x, pnaka(x, shape = 1), col = "orange")
lines(x, pnaka(x, shape = 2), col = "blue")
lines(x, pnaka(x, shape = 3), col = "green")
legend(2, 0.6, col = c("orange","blue","green"), lty = rep(1, len = 3),
       legend = paste("shape =", c(1, 2, 3)))
# End(Not run)

probs <- seq(0.1, 0.9, by = 0.1)
pnaka(qnaka(p = probs, shape = 2), shape = 2) - probs  # Should be all 0
```

nakagami  Nakagami Distribution Family Function

Description

Estimation of the two parameters of the Nakagami distribution by maximum likelihood estimation.

Usage

```r
nakagami(lscale = "loge", lshape = "loge", iscale = 1, ishape = NULL,
         nowarning = FALSE)
```
Arguments

- nowarning: Logical. Suppress a warning?
- lscale, lshape: Parameter link functions applied to the scale and shape parameters. Log links ensure they are positive. See Links for more choices and information.
- iscale, ishape: Optional initial values for the shape and scale parameters. For ishape, a NULL value means it is obtained in the initialize slot based on the value of iscale. For iscale, assigning a NULL means a value is obtained in the initialize slot, however, setting another numerical value is recommended if convergence fails or is too slow.

Details

The Nakagami distribution, which is useful for modelling wireless systems such as radio links, can be written

\[ f(y) = 2^{\text{shape}} / \Gamma(\text{shape}) \times y^{2 \times \text{shape} - 1} \times \exp(-\text{shape} \times y^2 / \text{scale}) \]

for \( y > 0, \text{shape} > 0, \text{scale} > 0 \). The mean of \( Y \) is \( \sqrt{\text{scale} / \text{shape}} \times \Gamma(\text{shape} + 0.5) / \Gamma(\text{shape}) \) and these are returned as the fitted values. By default, the linear/additive predictors are \( \eta_1 = \log(\text{scale}) \) and \( \eta_2 = \log(\text{shape}) \). Fisher scoring is implemented.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

The Nakagami distribution is also known as the Nakagami-\( m \) distribution, where \( m = \text{shape} \) here. Special cases: \( m = 0.5 \) is a one-sided Gaussian distribution and \( m = 1 \) is a Rayleigh distribution. The second moment is \( E(Y^2) = m \).

If \( Y \) has a Nakagami distribution with parameters \( \text{shape} \) and \( \text{scale} \) then \( Y^2 \) has a gamma distribution with shape parameter \( \text{shape} \) and scale parameter \( \text{scale} / \text{shape} \).

Author(s)

T. W. Yee

References


See Also

rnaka, gamma2, rayleigh.
Examples

nn <- 1000; shape <- exp(0); Scale <- exp(1)
data <- data.frame(y1 = sqrt(rgamma(nn, shape = shape, scale = Scale/shape)))
fit <- vglm(y1 ~ 1, nakagami, data = ndata, trace = TRUE, crit = "coeff")
data <- transform(ndata, y2 = rnaka(nn, scale = Scale, shape = shape))
fit <- vglm(y2 ~ 1, nakagami(iscale = 3), data = ndata, trace = TRUE)
head(fitted(fit))
with(ndata, mean(y2))
coef(fit, matrix = TRUE)
(Cfit <- Coef(fit))
## Not run: sy <- with(ndata, sort(y2))
hist(with(ndata, y2), prob = TRUE, main = "", xlab = "y", ylim = c(0, 0.6),
     col = "lightblue")
lines(dnaka(sy, scale = Cfit["scale"], shape = Cfit["shape"] ~ sy,
     data = ndata, col = "orange")
## End(Not run)

nbcanlink  

Description

Computes the negative binomial canonical link transformation, including its inverse and the first two derivatives.

Usage

nbcanlink(theta, size = NULL, wrt.eta = NULL, bvalue = NULL,
         inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)

Arguments

theta     Numeric or character. Typically the mean of a negative binomial (NB) distribution. See below for further details.
size, wrt.eta size contains the k matrix which must be of a conformable dimension as theta. Also, if deriv > 0 then wrt.eta is either 1 or 2 (1 for with respect to the first linear predictor, and 2 for with respect to the second linear predictor (a function of k)).
bvalue    Details at Links.
inverse, deriv, short, tag      Details at Links.

Details

The negative binomial (NB) canonical link is \( \log\left( \frac{\theta}{\theta + k} \right) \) where \( \theta \) is the mean of a NB distribution. The canonical link is used for theoretically relating the NB to GLM class.

This link function was specifically written for \texttt{negbinomial} and \texttt{negbinomial.size}, and should not be used elsewhere (these \texttt{VGAM} family functions have code that specifically handles \texttt{nbcanlink}().)
Value

For \( \text{deriv} = 0 \), the above equation when \( \text{inverse} = \text{FALSE} \), and if \( \text{inverse} = \text{TRUE} \) then \( \text{kmatrix} / \exp(-\text{theta}) \). For \( \text{deriv} = 1 \), then the function returns \( d \text{theta} / d \text{eta} \) as a function of \( \text{theta} \) if \( \text{inverse} = \text{FALSE} \), else if \( \text{inverse} = \text{TRUE} \), then it returns the reciprocal.

Warning

This function currently does not work very well with \texttt{negbinomial}! The NB-C model is sensitive to the initial values and may converge to a local solution. Pages 210 and 309 of Hilbe (2011) notes convergence difficulties (of Newton-Raphson type algorithms), and this applies here. This function should work okay with \texttt{negbinomial.size}. Currently trying something like \texttt{imethod = 3 or imu}, \texttt{stepsize = 0.5, maxit = 100, zero = -2} should help; see the example below.

Standard errors may be unreliable.

Note

While theoretically nice, this function is not recommended in general since its value is always negative (linear predictors ought to be unbounded in general). A \texttt{log} \ texttt{e} link for argument \texttt{lmu} is recommended instead.

Numerical instability may occur when \( \text{theta} \) is close to 0 or 1. Values of \( \text{theta} \) which are less than or equal to 0 can be replaced by \texttt{bvalue} before computing the link function value. See \texttt{Links}.

Author(s)

Thomas W. Yee

References


See Also

\texttt{negbinomial}, \texttt{negbinomial.size}.

Examples

\footnotesize
\begin{verbatim}
nbcanlink("mu", short = FALSE)

mymu <- 1:10  # Test some basic operations:
kmatrix <- matrix(runif(length(mymu)), length(mymu), 1)
etal <- nbcanlink(mymu, size = kmatrix)
ans2 <- nbcanlink(etal, size = kmatrix, inverse = TRUE)
max(abs(ans2 - mymu))  # Should be 0

## Not run: mymu <- c(seq(0.5, 10, length = 101))
kmatrix <- matrix(10, length(mymu), 1)
\end{verbatim}

\normalsize
nbolf

**Negative Binomial-Ordinal Link Function**

**Description**

Computes the negative binomial-ordinal transformation, including its inverse and the first two derivatives.

**Usage**

```r
nbolf(theta, cutpoint = NULL, k = NULL, 
      inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)
```

**Arguments**

- **theta**
  Numeric or character. See below for further details.

- **cutpoint, k**
  Here, k is the \( k \) parameter associated with the negative binomial distribution; see `negbinomial`. The cutpoints should be non-negative integers. If `nbolf()` is used as the link function in `cumulative` then one should choose `reverse = TRUE, parallel = TRUE`.

- **inverse, deriv, short, tag**
  Details at [Links](#).

---

```r
plot(nbcanlink(mymu, size = kmatrix) ~ mymu, las = 1, 
     type = "l", col = "blue", lwd = 1.5, xlab = expression({mu}))

# Estimate the parameters from some simulated data (see Warning section)
set.seed(123)
ndata <- data.frame(x2 = runif(nn <- 1000 ))
s1 <- exp(1); s2 <- exp(2)
ndata <- transform(ndata, etal = -1 - 2 * x2, # etal < 0
                    size1 = s1, 
                    size2 = s2)
ndata <- transform(ndata, 
                    mu1 = nbcanlink(etal, size = s1, inv = TRUE), 
                    mu2 = nbcanlink(etal, size = s2, inv = TRUE))
ndata <- transform(ndata, y1 = rnbinom(nn, mu = mu1, size = s1), 
                    y2 = rnbinom(nn, mu = mu2, size = s2))
head(ndata)
summary(ndata)

fit <- vglm(cbind(y1, y2) ~ x2, negbinomial("nbcanlink", imethod = 3), 
            stepsize = 0.5, data = ndata, # Deliberately slow the convergence rate
            maxit = 100, trace = TRUE) # Warning: may converge to a local soln
coef(fit, matrix = TRUE)
summary(fit)

## End(Not run)
```
Details

The negative binomial-ordinal link function (NBOLF) can be applied to a parameter lying in the unit interval. Its purpose is to link cumulative probabilities associated with an ordinal response coming from an underlying negative binomial distribution.

See Links for general information about VGAM link functions.

Value

See Yee (2012) for details.

Warning

Prediction may not work on vglm or vgam etc. objects if this link function is used.

Note

Numerical values of theta too close to 0 or 1 or out of range result in large positive or negative values, or maybe 0 depending on the arguments. Although measures have been taken to handle cases where theta is too close to 1 or 0, numerical instabilities may still arise.

In terms of the threshold approach with cumulative probabilities for an ordinal response this link function corresponds to the negative binomial distribution (see negbinomial) that has been recorded as an ordinal response using known cutpoints.

Author(s)

Thomas W. Yee

References


See Also

Links, negbinomial, polf, golf, nbolf2, cumulative, CommonVGAMffArguments.

Examples

nbolf("p", cutpoint = 2, k = 1, short = FALSE)
nbolf("p", cutpoint = 2, k = 1, tag = TRUE)

p <- seq(0.02, 0.98, by = 0.01)
y <- nbolf(p, cutpoint = 2, k = 1)
y. <- nbolf(p, cutpoint = 2, k = 1, deriv = 1)
max(abs(nbolf(y, cutpoint = 2, k = 1, inv = TRUE) - p))  # Should be 0

## Not run: par(mfrow = c(2, 1), las = 1)
plot(p, y, type = "l", col = "blue", main = "nbolf()")
abline(h = 0, v = 0.5, col = "red", lty = "dashed")
plot(p, y, type = "l", col = "blue", main = "(Reciprocal of) first NBOLF derivative")
## End(Not run)

# Another example
nn <- 1000
x2 <- sort(runif(nn))
x3 <- runif(nn)
mymu <- exp(3 + 1 * x2 - 2 * x3)
k <- 4
y1 <- rnorm(nn, mu = mymu, size = k)
cutpoints <- c(-Inf, 10, 20, Inf)
cuty <- cut(y1, breaks = cutpoints)
## Not run: plot(x2, x3, col = cuty, pch = as.character(cuty))
table(cuty) / sum(table(cuty))
fit <- vglm(cuty ~ x2 + x3, trace = TRUE,
cumulative(reverse = TRUE, multiple.responses = TRUE,
parallel = TRUE,
link = nbolf(cutpoint = cutpoints[2:3], k = k)))
head(depvar(fit))
head(fitted(fit))
head(predict(fit))
coef(fit)
coef(fit, matrix = TRUE)
constraints(fit)
fit@misc

---

### negbinomial

#### Negative Binomial Distribution Family Function

**Description**

Maximum likelihood estimation of the two parameters of a negative binomial distribution.

**Usage**

```r
negbinomial(lmu = "loge", lsize = "loge",
   imu = NULL, isize = NULL, probs.y = 0.75,
   nsimEIM = 250, cutoff.prob = 0.995,
   max.qnbinom = 1000, max.chunk.Mb = 20,
   deviance.arg = FALSE, imethod = 1, gsize = exp((-4):4),
   parallel = FALSE, ishrinkage = 0.95, zero = -2)

polya(lprob = "logit", lsize = "loge",
   iprob = NULL, isize = NULL, probs.y = 0.75, nsimEIM = 100,
   imethod = 1, ishrinkage = 0.95, zero = -2)

polyaR(lsize = "loge", lprob = "logit",
   isize = NULL, iprob = NULL, probs.y = 0.75, nsimEIM = 100,
   imethod = 1, ishrinkage = 0.95, zero = -1)
```
Arguments

\texttt{imu, isize, lprob}

Link functions applied to the \( \mu \), \( k \) and \( p \) parameters. See \texttt{Links} for more choices. Note that the \( \mu \), \( k \) and \( p \) parameters are the \texttt{mu}, size and prob arguments of \texttt{rnbinom} respectively. Common alternatives for \texttt{isize} are \texttt{negloge} and \texttt{reciprocal}, and \texttt{loglog} (if \( k > 1 \)).

\texttt{imu, isize, iprob}

Optional initial values for the mean and \( k \) and \( p \). For \( k \), if failure to converge occurs then try different values (and/or use \texttt{imethod}). For a \( S \)-column response, \texttt{isize} can be of length \( S \). A value \texttt{NULL} means an initial value for each response is computed internally using a grid search based on \texttt{gsize}. The last argument is ignored if used within \texttt{cqo}; see the \texttt{iKvector} argument of \texttt{qrvglm.control} instead.

\texttt{nnsimEIM}

This argument is used for computing the diagonal element of the \textit{expected information matrix} (EIM) corresponding to \( k \) based on the \textit{simulated Fisher scoring} (SFS) algorithm. See \texttt{CommonVGAMffArguments} for more information and the notes below. SFS is one of two algorithms for computing the EIM elements (so that both algorithms may be used on a given data set). SFS is faster than the exact method when \( Q_{\text{max}} \) is large.

\texttt{cutoff.prob}

Fed into the \texttt{p} argument of \texttt{qnbinom} in order to obtain an upper limit for the approximate support of the distribution, called \( Q_{\text{max}} \), say. Hence the approximate support is \( \emptyset : Q_{\text{max}} \). This argument should be a numeric and close to 1 but never exactly 1. Used to specify how many terms of the infinite series for computing the second diagonal element of the EIM are actually used. The closer this argument is to 1, the more accurate the standard errors of the regression coefficients will be.

\texttt{max.chunk.Mb, max.qnbinom}

\texttt{max.qnbinom} is used to describe the eligibility of individual observations to have their EIM computed by the \textit{exact method}. Here, we are concerned about computing the EIM wrt \( k \). The exact method algorithm operates separately on each response variable, and it constructs a large matrix provided that the number of columns is less than \texttt{max.qnbinom}. If so, then the computations are done in chunks, so that no more than about \texttt{max.chunk.Mb} megabytes of memory is used at a time (actually, it is proportional to this amount). Regarding eligibility of this algorithm, each observation must have the \texttt{cutoff.prob} quantile less than \texttt{max.qnbinom} as its approximate support. If you have abundant memory then you might try setting \texttt{max.chunk.Mb = Inf}, but then the computations might take a very long time. Setting \texttt{max.chunk.Mb = 0} or \texttt{max.qnbinom = 0} will force the EIM to be computed using the SFS algorithm only (this \textit{used to be} the default method for \textit{all} the observations). When the fitted values of the model are large and \( k \) is small, the computation of the EIM will be costly with respect to time and memory if the exact method is used. Hence the argument \texttt{max.qnbinom} limits the cost in terms of time.

\texttt{gsize}

Similar to \texttt{gsigma} in \texttt{CommonVGAMffArguments}.

\texttt{deviance.arg}

Logical. If \texttt{TRUE}, the deviance is computed \textit{after} convergence. It only works in the NB-2 model. It is also necessary to set \texttt{criterion = \"coefficients\"} or \texttt{half.step = \texttt{FALSE}} since one cannot use that criterion properly for the
negbinomial

minimization within the IRLS algorithm. It should be set TRUE when used with
CQO under the fast algorithm.

imethod
An integer with value 1 or 2 or 3 which specifies the initialization method for
the \( \mu \) parameter. If failure to converge occurs try another value and/or else specify
a value for ishrinkage and/or else specify a value for isize.

parallel
See CommonVGAMffArguments for more information. Setting parallel = TRUE
is useful in order to get something similar to quasipoissonff or what is known
as NB-1. If parallel = TRUE then the parallelism constraint does not apply to
any intercept term. You should set zero = NULL too if parallel = TRUE to
avoid a conflict.

probs.y
Passed into the probs argument of quantile when imethod = 3 to obtain an
initial value for the mean.

ishrinkage
How much shrinkage is used when initializing \( \mu \). The value must be between
0 and 1 inclusive, and a value of 0 means the individual response values are
used, and a value of 1 means the median or mean is used. This argument is
used in conjunction with imethod. If convergence failure occurs try setting this
argument to 1.

zero
Integer valued vector, usually assigned \(-2 \) or \(2\) if used at all. Specifies which of
the two linear/additive predictors are modelled as an intercept only. By default,
the \( k \) parameter (after lsize is applied) is modelled as a single unknown number
that is estimated. It can be modelled as a function of the explanatory variables
by setting zero = NULL: this has been called a NB-H model by Hilbe (2011).
A negative value means that the value is recycled, so setting \(-2\) means all \( k \) are
intercept-only. See CommonVGAMffArguments for more information.

Details

The negative binomial distribution can be motivated in several ways, e.g., as a Poisson distribution
with a mean that is gamma distributed. There are several common parametrizations of the negative
binomial distribution. The one used by negbinomial() uses the mean \( \mu \) and an index parameter \( k \),
both of which are positive. Specifically, the density of a random variable \( Y \) is

\[
f(y; \mu, k) = \binom{y + k - 1}{y} \left( \frac{\mu}{\mu + k} \right)^y \left( \frac{k}{k + \mu} \right)^k
\]

where \( y = 0, 1, 2, \ldots \), and \( \mu > 0 \) and \( k > 0 \). Note that the dispersion parameter is \( 1/k \), so that
as \( k \) approaches infinity the negative binomial distribution approaches a Poisson distribution. The
response has variance \( \text{Var}(Y) = \mu + \mu^2/k \). When fitted, the fitted.values slot of the object
contains the estimated value of the \( \mu \) parameter, i.e., of the mean \( E(Y) \). It is common for some
to use \( \alpha = 1/k \) as the ancillary or heterogeneity parameter; so common alternatives for lsize are
negloge and reciprocal.

For polya the density is

\[
f(y; p, k) = \binom{y + k - 1}{y} (1 - p)^y p^k
\]

where \( y = 0, 1, 2, \ldots \), and \( k > 0 \) and \( 0 < p < 1 \).
Family function polyaR() is the same as polya() except the order of the two parameters are switched. The reason is that polyaR() tries to match with \texttt{rnbinom} closely in terms of the argument order, etc. Should the probability parameter be of primary interest, probably, users will prefer using polya() rather than polyaR(). Possibly polyaR() will be decommissioned one day.

The negative binomial distribution can be coerced into the classical GLM framework with one of the parameters being of interest and the other treated as a nuisance/scale parameter (this is implemented in the \texttt{MASS} library). The \texttt{VGAM} family function \texttt{negbinomial()} treats both parameters on the same footing, and estimates them both by full maximum likelihood estimation.

The parameters \(\mu\) and \(k\) are independent (diagonal EIM), and the confidence region for \(k\) is extremely skewed so that its standard error is often of no practical use. The parameter \(1/k\) has been used as a measure of aggregation.

These \texttt{VGAM} family functions handle multiple responses, so that a response matrix can be inputted. The number of columns is the number of species, say, and setting \texttt{zero = -2} means that all species have a \(k\) equalling a (different) intercept only.

### Value

An object of class "\texttt{vglmff}" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm, rrvglm} and \texttt{vgam}.

### Warning

Poisson regression corresponds to \(k\) equalling infinity. If the data is Poisson or close to Poisson, numerical problems will occur. Possibly choosing a log-log link may help in such cases, otherwise try \texttt{poissonff} or \texttt{quasipoissonff}. It is possible to fit a NBD that has a similar variance function as a quasi-Poisson; see the NB-1 example below.

These functions are fragile; the maximum likelihood estimate of the index parameter is fraught (see Lawless, 1987). In general, the \texttt{quasipoissonff} is more robust. Other alternatives to \texttt{negbinomial} are to fit a NB-1 or RR-NB (aka NB-P) model; see Yee (2014). Also available are the NB-C, NB-H and NB-G. Assigning values to the isize argument may lead to a local solution, and smaller values are preferred over large values when using this argument.

If one wants to force SFS to be used on all observations, then set \(\max\.qnbinom = 0\) or \(\max\.chunk\.Mb = 0\).

If one wants to force the exact method to be used for all observations, then set \(\max\.qnbinom = \text{Inf}\).

If the computer has much memory, then trying \texttt{max\.chunk\.Mb = Inf} may provide a small speed increase. If SFS is used at all, then the \texttt{@weights} slot of the fitted object will be a matrix; otherwise that slot will be a \(0 \times 0\) matrix.

Yet to do: write a family function which uses the methods of moments estimator for \(k\).

### Note

These 3 functions implement 2 common parameterizations of the negative binomial (NB). Some people called the NB with integer \(k\) the Pascal distribution, whereas if \(k\) is real then this is the Polya distribution. I don’t. The one matching the details of \texttt{rnbinom} in terms of \(p\) and \(k\) is polya().

For polya() the code may fail when \(p\) is close to 0 or 1. It is not yet compatible with \texttt{cqo} or \texttt{cao}.

Suppose the response is called \texttt{ymat}. For \texttt{negbinomial()} the diagonal element of the expected information matrix (EIM) for parameter \(k\) involves an infinite series; consequently SFS (see \texttt{nsimEIM}) is used as the backup algorithm only. SFS should be better if \(\max(\texttt{ymat})\) is large, e.g., \(\max(\texttt{ymat}) > 1000\),
or if there are any outliers in ymat. The default algorithm involves a finite series approximation to the support \(0: \text{Inf}\); the arguments \texttt{max.memory}, \texttt{min.size} and \texttt{cutoff.prob} are pertinent.

Regardless of the algorithm used, convergence problems may occur, especially when the response has large outliers or is large in magnitude. If convergence failure occurs, try using arguments (in recommended decreasing order) \texttt{max.qnbinom}, \texttt{nsimEIM}, \texttt{cutoff.prob}, \texttt{ishrinkage}, \texttt{imethod}, \texttt{isize}, \texttt{zero}, \texttt{max.chunk}.Mb.

The function \texttt{negbinomial} can be used by the fast algorithm in \texttt{cqo}, however, setting \texttt{eq.tolerances = TRUE} and \texttt{I.tolerances = FALSE} is recommended.

In the first example below (Bliss and Fisher, 1953), from each of 6 McIntosh apple trees in an orchard that had been sprayed, 25 leaves were randomly selected. On each of the leaves, the number of adult female European red mites were counted.

There are two special uses of \texttt{negbinomial} for handling count data. Firstly, when used by \texttt{rrvglm} this results in a continuum of models in between and inclusive of quasi-Poisson and negative binomial regression. This is known as a reduced-rank negative binomial model (\texttt{RR-NB}). It fits a negative binomial log-linear regression with variance function \(\text{Var}(Y) = \mu + \delta_1 \mu^2\) where \(\delta_1\) and \(\delta_2\) are parameters to be estimated by MLE. Confidence intervals are available for \(\delta_2\), therefore it can be decided upon whether the data are quasi-Poisson or negative binomial, if any.

Secondly, the use of \texttt{negbinomial} with \texttt{parallel = TRUE} inside \texttt{vglm} can result in a model similar to \texttt{quasipoissonff}. This is named the \texttt{NB-1} model. The dispersion parameter is estimated by MLE whereas \texttt{glm} uses the method of moments. In particular, it fits a negative binomial log-linear regression with variance function \(\text{Var}(Y) = \phi_0 \mu\) where \(\phi_0\) is a parameter to be estimated by MLE. Confidence intervals are available for \(\phi_0\).

Author(s)

Thomas W. Yee

References


See Also

\texttt{quasipoissonff}, \texttt{poissonff}, \texttt{zinegbinomial}, \texttt{negbinomial.size} (e.g., \texttt{NB-G}), \texttt{nbcanlink} (\texttt{NB-C}), \texttt{posnegbinomial}, \texttt{inv.binomial}, \texttt{rnbinom}, \texttt{nbolf}, \texttt{rrvglm}, \texttt{cao}, \texttt{cqo}, \texttt{CommonVGAMffArguments}, \texttt{simulate.vlm}, \texttt{qnbinom}.
Examples

# Example 1: apple tree data (Bliss and Fisher, 1953)
appletree <- data.frame(y = 0:7, w = c(70, 38, 17, 10, 9, 3, 2, 1))
fit <- vglm(y ~ 1, negbinomial(deviance = TRUE), data = appletree,
weights = w, crit = "coef")  # Obtain the deviance
fit <- vglm(y ~ 1, negbinomial(deviance = TRUE), data = appletree,
weights = w, half.step = FALSE)  # Alternative method
summary(fit)
coef(fit, matrix = TRUE)
Coef(fit)  # For intercept-only models
deviance(fit)  # NB2 only; needs 'crit = "coef"' & 'deviance = TRUE' above

# Example 2: simulated data with multiple responses
ndata <- data.frame(x2 = runif(nn <- 300))
ndata <- transform(ndata, y1 = rnbinom(nn, mu = exp(3+x2), size = exp(1)),
y2 = rnbinom(nn, mu = exp(2-x2), size = exp(0)))
fit1 <- vglm(cbind(y1, y2) ~ x2, negbinomial, data = ndata, trace = TRUE)
coef(fit1, matrix = TRUE)

# Example 3: large counts implies SFS is used
ndata <- transform(ndata, y3 = rnbinom(nn, mu = exp(10+x2), size = exp(1)))
with(ndata, range(y3))  # Large counts
fit2 <- vglm(y3 ~ x2, negbinomial, data = ndata, trace = TRUE)
coef(fit2, matrix = TRUE)
head(fit2@weights)  # Non-empty; SFS was used

# Example 4: a NB-1 to estimate a negative binomial with Var(Y) = phi0 * mu
nn <- 500  # Number of observations
phi0 <- 10  # Specify this; should be greater than unity
delta0 <- 1 / (phi0 - 1)
mydata <- data.frame(x2 = runif(nn), x3 = runif(nn))
mydata <- transform(mydata, mu = exp(2 + 3 * x2 + 0 * x3))
mydata <- transform(mydata, y3 = rnbinom(nn, mu = mu, size = delta0 * mu))
## Not run:
plot(y3 ~ x2, data = mydata, pch = "*", col = "blue",
    main = paste("Var(Y) = ", phi0, "* mu", sep = ""), las = 1)
## End(Not run)
nb1 <- vglm(y3 ~ x2 + x3, negbinomial(parallel = TRUE, zero = NULL),
            data = mydata, trace = TRUE)
# Extracting out some quantities:
cnb1 <- coef(nb1, matrix = TRUE)
mydiff <- (cnb1["(Intercept)"], "loge(size)"]
          - cnb1["(Intercept)"], "loge(mu)"])
delta0.hat <- exp(mydiff)
(phi.hat <- 1 + 1 / delta0.hat)  # MLE of phi
summary(nb1)
# Obtain a 95 percent confidence interval for phi0:
myvec <- rbind(-1, 1, 0, 0)
(se.mydiff <- sqrt(t(myvec) %*% vcov(nb1) %*% myvec))
ci.mydiff <- mydiff + c(-1.96, 1.96) * se.mydiff
ci.delta0 <- ci.exp.mydiff <- exp(ci.mydiff)
(ci.phi0 <- 1 + 1 / rev(ci.delta0))  # The 95 percent conf. interval for phi0
Confint.nb1(nb1)  # Quick way to get it
summary(glm(y3 ~ x2 + x3, quasipoisson, mydata))$dispers  # cf. moment estimator

Description

Maximum likelihood estimation of the mean parameter of a negative binomial distribution with known size parameter.

Usage

negbinomial.size(size = Inf, lmu = "loge", imu = NULL, 
probs.y = 0.75, imethod = 1, 
is shrinkage = 0.95, zero = NULL)

Arguments

size      Numeric, positive. Same as argument size of rnbinom. If the response is a matrix then this is recycled to a matrix of the same dimension, by row (matrix with byrow = TRUE).

lmu, imu  Same as negbinomial.

probs.y   Same as negbinomial.

imethod, zero Same as negbinomial.

ishrinkage Same as negbinomial.

Details

This VGAM family function estimates only the mean parameter of the negative binomial distribution. See negbinomial for general information. Setting size = 1 gives what I call the NB-G (geometric model; see Hilbe (2011)). The default, size = Inf, corresponds to the Poisson distribution.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Note

If lmu = "nbcanlink" in negbinomial.size() then the size argument here should be assigned.

Author(s)

Thomas W. Yee
References


See Also

negbinomial, nbcanlink (NB-C model), quasipoissonff, poissonff, rnbinom, simulate.vlm.

Examples

# Simulated data with various multiple responses
size1 <- exp(1); size2 <- exp(2); size3 <- exp(0); size4 <- Inf
ndata <- data.frame(x2 = runif(nn <- 1000))
ndata <- transform(ndata, eta1 = -1 - 2 * x2, # eta1 must be negative
                   size1 = size1)
ndata <- transform(ndata, 
                   mu1 = nbcanlink(eta1, size = size1, inv = TRUE))
ndata <- transform(ndata, 
                   y1 = rnbinom(nn, mu = mu1, size = size1), # NB-C
                   y2 = rnbinom(nn, mu = exp(2 - x2), size = size2),
                   y3 = rnbinom(nn, mu = exp(3 + x2), size = size3), # NB-G
                   y4 = rpois (nn, la = exp(1 + x2)))

# Also known as NB-C with size known (Hilbe, 2011)
fit1 <- vglm(y1 ~ x2, negbinomial.size(size = size1, lmu = "nbcanlink"),
            data = ndata, trace = TRUE, crit = "coef")
coef(fit1, matrix = TRUE)
head(fit1@misc$size) # size saved here

fit2 <- vglm(cbind(y2, y3, y4) ~ x2,
            negbinomial.size(size = c(size2, size3, size4)),
            data = ndata, trace = TRUE)
coef(fit2, matrix = TRUE)
head(fit2@misc$size) # size saved here

---

**normal.vcm**

Univariate Normal Distribution as a Varying-Coefficient Model

Description

Maximum likelihood estimation of all the coefficients of a LM where each of the usual regression coefficients is modelled with other explanatory variables via parameter link functions. Thus this is a basic varying-coefficient model.
Usage

```r
normal.vcm(link.list = list("(Default)" = "identitylink"),
          earg.list = list("(Default)" = list()),
          lsd = "loge", lvar = "loge",
          esd = list(), evar = list(),
          var.arg = FALSE, imethod = 1,
          icoefficients = NULL, isd = NULL, zero = "M")
```

Arguments

- `link.list`, `earg.list`
  Link functions and extra arguments applied to the coefficients of the LM, excluding the standard deviation/variance. See `CommonVGAMffArguments` for more information. The default is for an identity link to be applied to each of the regression coefficients.

- `lsd`, `esd`, `lvar`, `evar`
  Link function and extra argument applied to the standard deviation/variance. See `CommonVGAMffArguments` for more information. Same as `uninormal`.

- `icoefficients`
  Optional initial values for the coefficients. Recycled to length \( M - 1 \) (does not include the standard deviation/variance). Try using this argument if there is a link function that is not programmed explicitly to handle range restrictions in the initialize slot.

- `var.arg`, `imethod`, `isd`
  Same as `uninormal`.

- `zero`
  See `CommonVGAMffArguments` for more information. The default applies to the last one, viz. the standard deviation/variance.

Details

This function allows all the usual LM regression coefficients to be modelled as functions of other explanatory variables via parameter link functions. For example, we may want some of them to be positive. Or we may want a subset of them to be positive and add to unity. So a class of such models have been named varying-coefficient models (VCMs).

The usual linear model is specified through argument `form2`. As with all other VGAM family functions, the linear/additive predictors are specified through argument `formula`.

The `multilogit` link allows a subset of the coefficients to be positive and add to unity. Either none or more than one call to `multilogit` is allowed. The last variable will be used as the baseline/reference group, and therefore excluded from the estimation.

By default, the log of the standard deviation is the last linear/additive predictor. It is recommended that this parameter be estimated as intercept-only, for numerical stability.

Technically, the Fisher information matrix is of unit-rank for all but the last parameter (the standard deviation/variance). Hence an approximation is used that pools over all the observations.

This VGAM family function cannot handle multiple responses. Also, this function will probably not have the full capabilities of the class of varying-coefficient models as described by Hastie and Tibshirani (1993). However, it should be able to manage some simple models, especially involving the following links: `identity`, `loge`, `logoff`, `loglog`, `logit`, `probit`, `cauchit`, `cloglog`, `rhobit`, `fisherz`.
Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Warning

This VGAM family function is fragile. One should monitor convergence, and possibly enter initial values especially when there are non-identity-link functions. If the initial value of the standard deviation/variance is too small then numerical problems may occur. One trick is to fit an intercept-only only model and feed its predict() output into argument etastart of a more complicated model. The use of the zero argument is recommended in order to keep models as simple as possible.

Note

The standard deviation/variance parameter is best modelled as intercept-only.

Yet to do: allow an argument such as parallel that enables many of the coefficients to be equal. Fix a bug: Coef() does not work for intercept-only models.

Author(s)

T. W. Yee

References


See Also

uninormal, lm.

Examples

ndata <- data.frame(x2 = runif(nn <- 2000))
# Note that coeff1 + coeff2 + coeff5 == 1. So try a "multilogit" link.
myoffset <- 10
ndata <- transform(ndata,
  coeff1 = 0.25, # "multilogit" link
  coeff2 = 0.25, # "multilogit" link
  coeff3 = exp(-0.5), # "loge" link
  coeff4 = logoff(+0.5, offset = myoffset, inverse = TRUE), # "logoff" link
  coeff5 = 0.50, # "multilogit" link
  coeff6 = 1.00, # "identitylink" link
  v2 = runif(nn),
  v3 = runif(nn),
  v4 = runif(nn),
  v5 = rnorm(nn),
  v6 = rnorm(nn))
ndata <- transform(ndata,
  Coeff1 = 0.25 - 0 * x2,
normal.vcm

Coef2 = 0.25 - 0 * x2,
Coef3 = logit(-0.5 - 1 * x2, inverse = TRUE),
Coef4 = loglog(0.5 - 1 * x2, inverse = TRUE),
Coef5 = 0.50 - 0 * x2,
Coef6 = 1.00 + 1 * x2)

ndata <- transform(ndata,
  y1 = coeffQ * 1 +
    coeff2 * v2 +
    coeff3 * v3 +
    coeff4 * v4 +
    coeff5 * v5 +
    coeff6 * v6 + rnorm(nn, sd = exp(0)),
  y2 = Coeff1 * 1 +
    Coeff2 * v2 +
    Coeff3 * v3 +
    Coeff4 * v4 +
    Coeff5 * v5 +
    Coeff6 * v6 + rnorm(nn, sd = exp(0)))

# An intercept-only model
fit1 <- vglm(y1 ~ 1,
  form2 = ~ 1 + v2 + v3 + v4 + v5 + v6,
  normal.vcm(link.list = list("(Intercept)" = "multilogit",
    "v2" = "multilogit",
    "v3" = "logit",
    "v4" = "logoff",
    "(Default)" = "identitylink",
    "v5" = "multilogit"),
  earg.list = list("(Intercept)" = list(),
    "v2" = list(),
    "v3" = list()
    "v4" = list(offset = myoffset),
    "v5" = list() ),
  data = ndata, trace = TRUE)
coef(fit1, matrix = TRUE)
summary(fit1)
# This works only for intercept-only models:
multilogit(rbind(coef(fit1, matrix = TRUE)[1, c(1, 2)]), inverse = TRUE)

# A model with covariate x2 for the regression coefficients
fit2 <- vglm(y2 ~ 1 + x2,
  form2 = ~ 1 + v2 + v3 + v4 + v5 + v6,
  normal.vcm(link.list = list("(Intercept)" = "multilogit",
    "v2" = "multilogit",
    "v3" = "logit",
    "v4" = "logoff",
    "(Default)" = "identitylink",
    "v5" = "multilogit"),
  earg.list = list("(Intercept)" = list(),
    "v2" = list(),
    "v3" = list(),
    "v4" = list(offset = myoffset),
    "v5" = list()))
nparam.vlm

Number of Parameters

Description

Returns the number of parameters in a fitted model object.

Usage

\[
\text{nparam(object, \ldots)} \\
\text{nparam.vlm(object, dpar = TRUE, \ldots)} \\
\text{nparam.vgam(object, dpar = TRUE, linear.only = FALSE, \ldots)} \\
\text{nparam.rrvglm(object, dpar = TRUE, \ldots)} \\
\text{nparam.qrrvglm(object, dpar = TRUE, \ldots)} \\
\text{nparam.rrvgam(object, dpar = TRUE, \ldots)}
\]

Arguments

- **object**: Some VGAM object, for example, having class `vglmff-class`.
- **\ldots**: Other possible arguments fed into the function.
- **dpar**: Logical, include any (estimated) dispersion parameters as a parameter?
- **linear.only**: Logical, include only the number of linear (parametric) parameters?

Details

The code was copied from the AIC() methods functions.

Value

Returns a numeric value with the corresponding number of parameters. For `vgam` objects, this may be real rather than integer, because the nonlinear degrees of freedom is real-valued.

Warning

This code has not been double-checked.

Author(s)

T. W. Yee.
olympics 2008 and 2012 Summer Olympic Final Medal Count Data

Description
Final medal count, by country, for the Summer 2008 and 2012 Olympic Games.

Usage

data(olym08)
data(olym12)

Format
A data frame with 87 or 85 observations on the following 6 variables.

- rank  a numeric vector, overall ranking of the countries.
- country a factor.
- gold  a numeric vector, number of gold medals.
- silver a numeric vector, number of silver medals.
- bronze a numeric vector, number of bronze medals.
- totalmedal a numeric vector, total number of medals.

Details
The events were held during (i) August 8–24, 2008, in Beijing; and (ii) 27 July–12 August, 2012, in London.
References

The official English website was/is http://en.beijing2008.cn and http://www.london2012.com. Help from Viet Hoang Quoc is gratefully acknowledged.

See Also

grc.

Examples

summary(olymp08)
summary(olymp12)
## maybe str(olymp08); plot(olymp08) ...
## Not run: par(mfrow = c(1, 2))
myylim <- c(0, 55)
with(head(olymp08, n = 8),
  barplot(rbind(gold, silver, bronze),
      col = c("gold", "grey", "brown"), # No "silver" or "bronze"!
      names.arg = country, cex.names = 0.5, ylim = myylim,
      beside = TRUE, main = "2008 Summer Olympic Final Medal Count",
      ylab = "Medal count", las = 1,
      sub = "Top 8 countries; 'gold'=gold, 'grey'=silver, 'brown'=bronze")
  with(head(olymp12, n = 8),
    barplot(rbind(gold, silver, bronze),
            col = c("gold", "grey", "brown"), # No "silver" or "bronze"!
            names.arg = country, cex.names = 0.5, ylim = myylim,
            beside = TRUE, main = "2012 Summer Olympic Final Medal Count",
            ylab = "Medal count", las = 1,
            sub = "Top 8 countries; 'gold'=gold, 'grey'=silver, 'brown'=bronze")
## End(Not run)

Optimums

Description

Generic function for the optimums (or optima) of a model.

Usage

Opt(object, ...)

Arguments

object An object for which the computation or extraction of an optimum (or optimums) is meaningful.

... Other arguments fed into the specific methods function of the model. Sometimes they are fed into the methods function for Coef.
Details

Different models can define an optimum in different ways. Many models have no such notion or definition.

Optima occur in quadratic and additive ordination, e.g., CQO or CAO. For these models the optimum is the value of the latent variable where the maximum occurs, i.e., where the fitted value achieves its highest value. For quadratic ordination models there is a formula for the optimum but for additive ordination models the optimum must be searched for numerically. If it occurs on the boundary, then the optimum is undefined. At an optimum, the fitted value of the response is called the maximum.

Value

The value returned depends specifically on the methods function invoked.

Note

In ordination, the optimum of a species is sometimes called the species score.

Author(s)

Thomas W. Yee

References


See Also

Opt.qrrvglm, Max, Tol.

Examples

```r
set.seed(111)  # This leads to the global solution
hspted[,1:6] <- scale(hspted[,1:6])  # Standardized environmental vars
# vvv p1 = cpo(cbind(Alopacce, Alopune, Alopabr, Arctlute, Arctperi,
# vvv Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpulr,
# vvv Trocterr, Zoraspin) ~
# vvv WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + Reflux,
# vvv Bestof = 2,
# vvv fam = quasipossionff, data = hspted, Crow
positive = FALSE)
# vvv Opt(p1)

## Not run: index <- 1:ncol(depvar(p1))
persp(p1, col = index, las = 1, lwd = 2, main = "Vertical lines at the optimums")
abline(v = Opt(p1), lty = 2, col = index)

## End(Not run)
```
Ordinal Poisson Family Function

Description

Fits a Poisson regression where the response is ordinal (the Poisson counts are grouped between known cutpoints).

Usage

ordpoisson(cutpoints, countdata = FALSE, NOS = NULL, Levels = NULL, init.mu = NULL, parallel = FALSE, zero = NULL, link = "logel")

Arguments

cutpoints Numeric. The cutpoints, \( K_l \). These must be non-negative integers. Inf values may be included. See below for further details.

countdata Logical. Is the response (LHS of formula) in count-data format? If not then the response is a matrix or vector with values 1, 2, \ldots, L, say, where L is the number of levels. Such input can be generated with cut with argument labels = FALSE. If countdata = TRUE then the response is expected to be in the same format as depvar(fit) where fit is a fitted model with ordpoisson as the VGAM family function. That is, the response is matrix of counts with L columns (if NOS = 1).

NOS Integer. The number of species, or more generally, the number of response random variates. This argument must be specified when countdata = TRUE. Usually NOS = 1.

Levels Integer vector, recycled to length NOS if necessary. The number of levels for each response random variate. This argument should agree with cutpoints. This argument must be specified when countdata = TRUE.

init.mu Numeric. Initial values for the means of the Poisson regressions. Recycled to length NOS if necessary. Use this argument if the default initial values fail (the default is to compute an initial value internally).

parallel, zero, link
See poisonnff.

Details

This VGAM family function uses maximum likelihood estimation (Fisher scoring) to fit a Poisson regression to each column of a matrix response. The data, however, is ordinal, and is obtained from known integer cutpoints. Here, \( l = 1, \ldots, L \) where \( L (L \geq 2) \) is the number of levels. In more detail, let \( Y^* = l \) if \( K_{l-1} < Y \leq K_l \) where the \( K_l \) are the cutpoints. We have \( K_0 = -\infty \) and \( K_L = \infty \). The response for this family function corresponds to \( Y^* \) but we are really interested in the Poisson regression of \( Y \).
If \( NOS=1 \) then the argument `cutpoints` is a vector \((K_1, K_2, \ldots, K_L)\) where the last value (\( \text{Inf} \)) is optional. If \( NOS>1 \) then the vector should have \( NOS-1 \) \( \text{Inf} \) values separating the cutpoints. For example, if there are \( NOS=3 \) responses, then something like `ordpoisson(cut = c(\emptyset, 5, 10, \text{Inf}, 20, 30, \text{Inf}, 0, 10, 40, \text{Inf})` is valid.

**Value**

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

**Warning**

The input requires care as little to no checking is done. If `fit` is the fitted object, have a look at `fit@extra` and `depvar(fit)` to check.

**Note**

Sometimes there are no observations between two cutpoints. If so, the arguments `Levels` and `NOS` need to be specified too. See below for an example.

**Author(s)**

Thomas W. Yee

**References**


**See Also**

`poissonff`, `polf`, `ordered`.

**Examples**

```r
set.seed(123)  # Example 1
x2 <- runif(n <- 1000); x3 <- runif(n)
mymu <- exp(3 - 1 * x2 + 2 * x3)
y1 <- rpois(n, lambda = mymu)
cutpts <- c(-Inf, 20, 30, Inf)
fcutpts <- cutpts[is.finite(cutpts)]  # finite cutpoints
ystar <- cut(y1, breaks = cutpts, labels = FALSE)
## Not run:
plot(x2, x3, col = ystar, pch = as.character(ystar))

## End(Not run)
table(ystar) / sum(table(ystar))
fit <- vglm(ystar ~ x2 + x3, fam = ordpoisson(cutpoi = fcutpts))
head(depvar(fit))  # This can be input if countdata = TRUE
head(fitted(fit))
head(predict(fit))
```
oxtemp

Oxford Temperature Data

Description

Annual maximum temperatures collected at Oxford, UK.

Usage

data(oxtemp)

Format

A data frame with 80 observations on the following 2 variables.

maxtemp Annual maximum temperatures (in degrees Fahrenheit).

year The values 1901 to 1980.

Details

The data were collected from 1901 to 1980.

Source

Unknown.

Examples

## Not run: fit <- vglm(maxtemp ~ 1, egev, data = oxtemp, trace = TRUE)
The Paralogistic Distribution

Description

Density, distribution function, quantile function and random generation for the paralogistic distribution with shape parameter \( a \) and scale parameter \( \text{scale} \).

Usage

\[
\begin{align*}
\text{dparalogistic}(x, \text{scale} = 1, \text{shape} = a, \log = \text{FALSE}) \\
\text{pparalogistic}(q, \text{scale} = 1, \text{shape} = a, \text{lower.tail} = \text{TRUE}, \log.p = \text{FALSE}) \\
\text{qparalogistic}(p, \text{scale} = 1, \text{shape} = a, \text{lower.tail} = \text{TRUE}, \log.p = \text{FALSE}) \\
\text{rparalogistic}(n, \text{scale} = 1, \text{shape} = a)
\end{align*}
\]

Arguments

- \( x, q \) vector of quantiles.
- \( p \) vector of probabilities.
- \( n \) number of observations. If \( \text{length}(n) > 1 \), the length is taken to be the number required.
- \( \text{shape} = a \) shape parameter.
- \( \text{scale} \) scale parameter.
- \( \log \) Logical. If \( \log = \text{TRUE} \) then the logarithm of the density is returned.
- \( \text{lower.tail}, \log.p \) Same meaning as in \text{pnorm} or \text{qnorm}.

Details

See \text{paralogistic}, which is the \text{VGAM} family function for estimating the parameters by maximum likelihood estimation.

Value

- \text{dparalogistic} gives the density,
- \text{pparalogistic} gives the distribution function,
- \text{qparalogistic} gives the quantile function,
- \text{rparalogistic} generates random deviates.

Note

The paralogistic distribution is a special case of the 4-parameter generalized beta II distribution.

Author(s)

T. W. Yee and Kai Huang
paralogistic

References


See Also

paralogistic, genbetaII.

Examples

```r
pdata <- data.frame(y = rparalogistic(n = 3000, scale = exp(1), exp(2)))
fit <- vglm(y ~ 1, paralogistic(lss = FALSE, ishape1.a = 4.1),
           data = pdata, trace = TRUE)
coef(fit, matrix = TRUE)
coef(fit)
```

paralogistic  Paralogistic Distribution Family Function

Description

Maximum likelihood estimation of the 2-parameter paralogistic distribution.

Usage

```r
paralogistic(lscale = "loge", lshape1.a = "loge", iscale = NULL,
             ishape1.a = NULL, imethod = 1, lss = TRUE, gscale = exp(-5:5),
             gshape1.a = exp(-5:5), probs.y = c(0.25, 0.5, 0.75),
             zero = ifelse(lss, -2, -1))
```

Arguments

- `lss`  
  See CommonVGAMffArguments for important information.
- `lshape1.a, lscale`  
  Parameter link functions applied to the (positive) parameters \(a\) and \(scale\). See Links for more choices.
- `iscale, ishape1.a, imethod, zero`  
  See CommonVGAMffArguments for information. For `imethod = 2` a good initial value for `ishape1.a` is needed to obtain good estimates for the other parameter.
- `gscale, gshape1.a`  
  See CommonVGAMffArguments for information.
- `probs.y`  
  See CommonVGAMffArguments for information.
Details

The 2-parameter paralogistic distribution is the 4-parameter generalized beta II distribution with shape parameter \( p = 1 \) and \( a = q \). It is the 3-parameter Singh-Maddala distribution with \( a = q \). More details can be found in Kleiber and Kotz (2003).

The 2-parameter paralogistic has density

\[
f(y) = a^2 y^{a-1}/[b^a \{1 + (y/b)^a\}^{1+a}]
\]

for \( a > 0, b > 0, y \geq 0 \). Here, \( b \) is the scale parameter scale, and \( a \) is the shape parameter. The mean is

\[
E(Y) = b \Gamma(1 + 1/a) \Gamma(a - 1/a)/\Gamma(a)
\]

provided \( a > 1 \); these are returned as the fitted values. This family function handles multiple responses.

Value

An object of class "gllmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

See the notes in genbetaII.

Author(s)

T. W. Yee

References


See Also

Paralogistic, sinmad, genbetaII, betaII, dagum, fisk, inv.lomax, lomax, inv.paralogistic.

Examples

```r
pdata <- data.frame(y = rparalogistic(n = 3000, exp(1), scale = exp(1)))
fit <- vglm(y ~ 1, paralogistic(lss = FALSE), data = pdata, trace = TRUE)
fit <- vglm(y ~ 1, paralogistic(ishape1.a = 2.3, iscale = 5),
            data = pdata, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)
```
The Pareto Distribution

Description

Density, distribution function, quantile function and random generation for the Pareto(I) distribution with parameters scale and shape.

Usage

dpareto(x, scale = 1, shape, log = FALSE)
ppareto(q, scale = 1, shape, lower.tail = TRUE, log.p = FALSE)
qpareto(p, scale = 1, shape, lower.tail = TRUE, log.p = FALSE)
rpareto(n, scale = 1, shape)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. Same as in runif.
scale, shape the \( \alpha \) and \( k \) parameters.
log Logical. If log = TRUE then the logarithm of the density is returned.
lower.tail, log.p Same meaning as in pnorm or qnorm.

Details

See paretoff, the VGAM family function for estimating the parameter \( k \) by maximum likelihood estimation, for the formula of the probability density function and the range restrictions imposed on the parameters.

Value

dpareto gives the density, ppareto gives the distribution function, qpareto gives the quantile function, and rpareto generates random deviates.

Author(s)

T. W. Yee and Kai Huang

References

See Also

paretoff, ParetoIV.

Examples

alpha <- 3; k <- exp(1); x <- seq(2.8, 8, len = 300)
## Not run:
plot(x, dpareto(x, scale = alpha, shape = k), type = "l",
     main = "Pareto density split into 10 equal areas")
abline(h = 0, col = "blue", lty = 2)
qvec <- qpareto(seq(0.1, 0.9, by = 0.1), scale = alpha, shape = k)
lines(qvec, dpareto(qvec, scale = alpha, shape = k),
      col = "purple", lty = 3, type = "h")

## End(Not run)
pvec <- seq(0.1, 0.9, by = 0.1)
qvec <- qpareto(pvec, scale = alpha, shape = k)
ppareto(pvec, scale = alpha, shape = k)
qpareto(ppareto(qvec, scale = alpha, shape = k),
      scale = alpha, shape = k) - qvec  # Should be 0

Description

Estimates one of the parameters of the Pareto(I) distribution by maximum likelihood estimation. Also includes the upper truncated Pareto(I) distribution.

Usage

paretoff(scale = NULL, lshape = "loge")
truncpareto(lower, upper, lshape = "loge", ishape = NULL, imethod = 1)

Arguments

lshape Parameter link function applied to the parameter \( k \). See Links for more choices. A log link is the default because \( k \) is positive.

scale Numeric. The parameter \( \alpha \) below. If the user inputs a number then it is assumed known with this value. The default means it is estimated by maximum likelihood estimation, which means \( \min(y) \) is used, where \( y \) is the response vector.

lower, upper Numeric. Lower and upper limits for the truncated Pareto distribution. Each must be positive and of length 1. They are called \( \alpha \) and \( U \) below.

ishape Numeric. Optional initial value for the shape parameter. A NULL means a value is obtained internally. If failure to converge occurs try specifying a value, e.g., 1 or 2.

imethod See CommonVGAMffArguments for information. If failure to converge occurs then try specifying a value for ishape.
Details

A random variable $Y$ has a Pareto distribution if

$$P[Y > y] = C/y^k$$

for some positive $k$ and $C$. This model is important in many applications due to the power law probability tail, especially for large values of $y$.

The Pareto distribution, which is used a lot in economics, has a probability density function that can be written

$$f(y; \alpha, k) = k\alpha^k y^{-k-1}$$

for $0 < \alpha < y$ and $0 < k$. The $\alpha$ is called the scale parameter, and it is either assumed known or else $\min(y)$ is used. The parameter $k$ is called the shape parameter. The mean of $Y$ is $\alpha k / (k - 1)$ provided $k > 1$. Its variance is $\alpha^2 k / ((k - 1)^2 (k - 2))$ provided $k > 2$.

The upper truncated Pareto distribution has a probability density function that can be written

$$f(y) = k\alpha^k [y^{k+1}(1 - (\alpha/U)^k)]$$

for $0 < \alpha < y < U < \infty$ and $k > 0$. Possibly, better names for $k$ are the index and tail parameters. Here, $\alpha$ and $U$ are known. The mean of $Y$ is $k\alpha^k (U^{1-k} - \alpha^{1-k})/((1-k)(1-(\alpha/U)^k))$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

The usual or unbounded Pareto distribution has two parameters (called $\alpha$ and $k$ here) but the family function paretoff estimates only $k$ using iteratively reweighted least squares. The MLE of the $\alpha$ parameter lies on the boundary and is $\min(y)$ where $y$ is the response. Consequently, using the default argument values, the standard errors are incorrect when one does a summary on the fitted object. If the user inputs a value for alpha then it is assumed known with this value and then summary on the fitted object should be correct. Numerical problems may occur for small $k$, e.g., $k < 1$.

Note

Outside of economics, the Pareto distribution is known as the Bradford distribution.

For paretoff, if the estimate of $k$ is less than or equal to unity then the fitted values will be NAs. Also, paretoff fits the Pareto(I) distribution. See paretoIV for the more general Pareto(IV/III/II) distributions, but there is a slight change in notation: $s = k$ and $b = \alpha$.

In some applications the Pareto law is truncated by a natural upper bound on the probability tail. The upper truncated Pareto distribution has three parameters (called $\alpha$, $U$ and $k$ here) but the family function truncpareto() estimates only $k$. With known lower and upper limits, the ML estimator of $k$ has the usual properties of MLEs. Aban (2006) discusses other inferential details.

Author(s)

T. W. Yee
References


See Also

Pareto, Truncpareto, paretoIV, gpd, benini1.

Examples

```r
alpha <- 2; kay <- exp(3)
pdata <- data.frame(y = rpareto(n = 1000, scale = alpha, shape = kay))
fit <- vglm(y ~ 1, paretof, data = pdata, trace = TRUE)
fit@extra # The estimate of alpha is here
head(fitted(fit))
with(pdata, mean(y))
coef(fit, matrix = TRUE)
summary(fit) # Standard errors are incorrect!!

# Here, alpha is assumed known
fit2 <- vglm(y ~ 1, paretof(scale = alpha), data = pdata, trace = TRUE)
fit2@extra # alpha stored here
head(fitted(fit2))
coef(fit2, matrix = TRUE)
summary(fit2) # Standard errors are okay

# Upper truncated Pareto distribution
lower <- 2; upper <- 8; kay <- exp(2)
pdata3 <- data.frame(y = rtruncpareto(n = 100, lower = lower, upper = upper, shape = kay))
fit3 <- vglm(y ~ 1, truncpareto(lower, upper), data = pdata3, trace = TRUE)
coef(fit3, matrix = TRUE)
c(fit3@misc$lower, fit3@misc$upper)
```

The Pareto(IV/III/II) Distributions

Description

Density, distribution function, quantile function and random generation for the Pareto(IV/III/II) distributions.
Usage

dparetoIV(x, location = 0, scale = 1, inequality = 1, shape = 1, log = FALSE)
pparetoIV(q, location = 0, scale = 1, inequality = 1, shape = 1,
    lower.tail = TRUE, log.p = FALSE)
qparetoIV(p, location = 0, scale = 1, inequality = 1, shape = 1,
    lower.tail = TRUE, log.p = FALSE)
rparetoIV(n, location = 0, scale = 1, inequality = 1, shape = 1)
dparetoIII(x, location = 0, scale = 1, inequality = 1, log = FALSE)
pparetoIII(q, location = 0, scale = 1, inequality = 1,
    lower.tail = TRUE, log.p = FALSE)
qparetoIII(p, location = 0, scale = 1, inequality = 1,
    lower.tail = TRUE, log.p = FALSE)
rparetoIII(n, location = 0, scale = 1, inequality = 1)
dparetoII(x, location = 0, scale = 1, shape = 1, log = FALSE)
pparetoII(q, location = 0, scale = 1, shape = 1,
    lower.tail = TRUE, log.p = FALSE)
qparetoII(p, location = 0, scale = 1, shape = 1,
    lower.tail = TRUE, log.p = FALSE)
rparetoII(n, location = 0, scale = 1, shape = 1)
dparetoI(x, scale = 1, shape = 1, log = FALSE)
pparetoI(q, scale = 1, shape = 1,
    lower.tail = TRUE, log.p = FALSE)
qparetoI(p, scale = 1, shape = 1,
    lower.tail = TRUE, log.p = FALSE)
rparetoI(n, scale = 1, shape = 1)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. Same as in runif.
location the location parameter.
scale, shape, inequality the (positive) scale, inequality and shape parameters.
log Logical. If log = TRUE then the logarithm of the density is returned.
lower.tail, log.p Same meaning as in pnorm or qnorm.

Details

For the formulas and other details see paretoIV.

Value

Functions beginning with the letter d give the density, functions beginning with the letter p give the
distribution function, functions beginning with the letter q give the quantile function, and functions
beginning with the letter r generates random deviates.
Note

The functions \( dpqr \text{paretoIV} \) are the same as \( dpqr \text{pareto} \) except for a slight change in notation: \( s = k \) and \( b = \alpha \); see \text{Pareto}.

Author(s)

T. W. Yee and Kai Huang

References


See Also

\text{paretoIV}, \text{Pareto}.

Examples

```r
## Not run:
x <- seq(-0.2, 4, by = 0.01)
loc <- 0; Scale <- 1; ineq <- 1; shape <- 1.0
plot(x, dparetoIV(x, loc, Scale, ineq, shape), type = "l", col = "blue",
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Purple are 5,10,...,95 percentiles", ylim = 0:1, las = 1, ylab = "")
abline(h = 0, col = "blue", lty = 2)
Q <- qparetoIV(seq(0.05, 0.95, by = 0.05), loc, Scale, ineq, shape)
lines(Q, dparetoIV(Q, loc, Scale, ineq, shape), col = "purple", lty = 3, type = "h")
lines(x, pparetoIV(x, loc, Scale, ineq, shape), col = "orange")
abline(h = 0, lty = 2)
## End(Not run)
```

Description

Estimates three of the parameters of the Pareto(IV) distribution by maximum likelihood estimation. Some special cases of this distribution are also handled.
Usage

```r
paretoIV(location = 0, lscale = "loge", linequality = "loge", lshape = "loge",
         iscale = 1, iinequality = 1, ishape = NULL, imethod = 1)
paretoIII(location = 0, lscale = "loge", linequality = "loge",
          iscale = NULL, iinequality = NULL)
paretoII(location = 0, lscale = "loge", lshape = "loge",
          iscale = NULL, ishape = NULL)
```

Arguments

- `location` - Location parameter, called \( a \) below. It is assumed known.
- `lscale`, `linequality`, `lshape` - Parameter link functions for the scale parameter (called \( b \) below), inequality parameter (called \( g \) below), and shape parameter (called \( s \) below). See Links for more choices. A log link is the default for all because all these parameters are positive.
- `iscale`, `iinequality`, `ishape` - Initial values for the parameters. A NULL value means that it is obtained internally. If convergence failure occurs, use these arguments to input some alternative initial values.
- `imethod` - Method of initialization for the shape parameter. Currently only values 1 and 2 are available. Try the other value if convergence failure occurs.

Details

The Pareto(IV) distribution, which is used in actuarial science, economics, finance and telecommunications, has a cumulative distribution function that can be written

\[
F(y) = 1 - \left[1 + \left(\frac{y - a}{b}\right)^{1/g}\right]^{-s}
\]

for \( y > a \), \( b > 0 \), \( g > 0 \) and \( s > 0 \). The \( a \) is called the location parameter, \( b \) the scale parameter, \( g \) the inequality parameter, and \( s \) the shape parameter.

The location parameter is assumed known otherwise the Pareto(IV) distribution will not be a regular family. This assumption is not too restrictive in modelling because in typical applications this parameter is known, e.g., in insurance and reinsurance it is pre-defined by a contract and can be represented as a deductible or a retention level.

The inequality parameter is so-called because of its interpretation in the economics context. If we choose a unit shape parameter value and a zero location parameter value then the inequality parameter is the Gini index of inequality, provided \( g \leq 1 \).

The fitted values are currently the median, e.g., `qp` is used for `paretoIV()`.

There are a number of special cases of the Pareto(IV) distribution. These include the Pareto(I), Pareto(II), Pareto(III), and Burr family of distributions. Denoting \( PIV(a, b, g, s) \) as the Pareto(IV) distribution, the Burr distribution \( Burr(b, g, s) \) is \( PIV(a = 0, b, 1/g, s) \), the Pareto(III) distribution \( PIII(a, b, g) \) is \( PIV(a, b, g, s = 1) \), the Pareto(II) distribution \( PII(a, b, s) \) is \( PIV(a, b, g = 1, s) \), and the Pareto(I) distribution \( PI(b, s) \) is \( PIV(b, b, g = 1, s) \). Thus the Burr distribution can be fitted using the `negloge` link function and using the default `location=0` argument. The Pareto(I) distribution can be fitted using `paretoff` but there is a slight change in notation: \( s = k \) and \( b = \alpha \).
Value

An object of class "vglmff" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, and \texttt{vgam}.

Warning

The Pareto(IV) distribution is very general, for example, special cases include the Pareto(I), Pareto(II), Pareto(III), and Burr family of distributions. [Johnson et al. (1994) says on p.19 that fitting Type IV by ML is very difficult and rarely attempted]. Consequently, reasonably good initial values are recommended, and convergence to a local solution may occur. For this reason setting \texttt{trace=TRUE} is a good idea for monitoring the convergence. Large samples are ideally required to get reasonable results.

Note

The extra slot of the fitted object has a component called "location" which stores the location parameter value(s).

Author(s)

T. W. Yee

References


See Also

\texttt{paretoIV, paretoff, gpd}.

Examples

```r
pdata <- data.frame(y = rparetoIV(2000, scale = exp(1),
                        ineq = exp(-0.3), shape = exp(1)))
## Not run: par(mfrow = c(2, 1))
with(pdata, hist(y)); with(pdata, hist(log(y)))
## End(Not run)
fit <- vglm(y ~ 1, paretoIV, data = pdata, trace = TRUE)
head(fitted(fit))
summary(pdata)
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)
```
The Perks Distribution

Description
Density, cumulative distribution function, quantile function and random generation for the Perks distribution.

Usage
\[
\begin{align*}
\text{dperks}(x, \text{scale} = 1, \text{shape}, \text{log} = \text{FALSE}) \\
\text{pperks}(q, \text{scale} = 1, \text{shape}, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE}) \\
\text{qperks}(p, \text{scale} = 1, \text{shape}, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE}) \\
\text{rperks}(n, \text{scale} = 1, \text{shape})
\end{align*}
\]

Arguments
- \(x, q\) vector of quantiles.
- \(p\) vector of probabilities.
- \(n\) number of observations. Same as in \text{runif}.
- \(\text{log}\) Logical. If \(\text{log} = \text{TRUE}\) then the logarithm of the density is returned.
- \(\text{lower.tail, log.p}\) Same meaning as in \text{pnorm} or \text{qnorm}.
- \(\text{shape, scale}\) positive shape and scale parameters.

Details
See \text{perks} for details.

Value
\(\text{dperks}\) gives the density, \(\text{pperks}\) gives the cumulative distribution function, \(\text{qperks}\) gives the quantile function, and \(\text{rperks}\) generates random deviates.

Author(s)
T. W. Yee and Kai Huang

See Also
\text{perks}.
perks

Examples

```r
probs <- seq(0.01, 0.99, by = 0.01)
Shape <- exp(-1.0); Scale <- exp(1);
max(abs(pperks(qpperks(p = probs, Shape, Scale),
           Shape, Scale) - probs))  # Should be 0

## Not run: x <- seq(-0.1, 0.7, by = 0.01);
plot(x, dperks(x, Shape, Scale), type = "l", col = "blue", las = 1,
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Purple lines are the 10,20,...,90 percentiles",
     ylim = range(x, probs))
abline(h = 0, col = "blue", lty = 2)
lines(x, pperks(x, Shape, Scale), col = "orange")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qperks(probs, Shape, Scale)
lines(Q, dperks(Q, Shape, Scale), col = "purple", lty = 3, type = "h")
pperks(Q, Shape, Scale) - probs  # Should be all zero
abline(h = probs, col = "purple", lty = 3)
## End(Not run)
```

---

**Perks Distribution Family Function**

Description

Maximum likelihood estimation of the 2-parameter Perks distribution.

Usage

```r
perks(1scale = "loge", 1shape = "loge",
     iscale = NULL, ishape = NULL,
     gscale = exp(-5:5), gshape = exp(-5:5),
     nsimEIM = 500, oim.mean = FALSE, zero = NULL, nowarning = FALSE)
```

Arguments

- `nowarning` Logical. Suppress a warning? Ignored for VGAM 0.9-7 and higher.
- `1scale`, `1shape` Parameter link functions applied to the shape parameter shape, scale parameter scale. All parameters are treated as positive here See Links for more choices.
- `iscale`, `ishape` Optional initial values. A NULL means a value is computed internally.
- `gscale`, `gshape` See CommonVGAMffArguments.
- `nsimEIM`, `zero` See CommonVGAMffArguments.
- `oim.mean` To be currently ignored.
The Perks distribution has cumulative distribution function

\[ F(x; \alpha, \beta) = 1 - \left( \frac{1 + \alpha}{1 + \alpha e^{\beta y}} \right)^{1/\beta} \]

which leads to a probability density function

\[ f(x; \alpha, \beta) = \left[ 1 + \alpha \right]^{1/\beta} \frac{\alpha e^{\beta y}}{(1 + \alpha e^{\beta y})^{1+1/\beta}} \]

for \( \alpha > 0, \beta > 0, x > 0 \). Here, \( \beta \) is called the scale parameter scale, and \( \alpha \) is called a shape parameter. The moments for this distribution do not appear to be available in closed form. Simulated Fisher scoring is used and multiple responses are handled.

Value

An object of class "glmff" (see glmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

A lot of care is needed because this is a rather difficult distribution for parameter estimation. If the self-starting initial values fail then try experimenting with the initial value arguments, especially iscale. Successful convergence depends on having very good initial values. Also, monitor convergence by setting trace = TRUE.

Author(s)

T. W. Yee

References


See Also

dperks, simulate.vlm.

Examples

```r
## Not run: set.seed(123)
pdata <- data.frame(x2 = runif(nn <- 1000)) # x2 unused
pdata <- transform(pdata, eta1 = -1,
    ceta1 = 1)
pdata <- transform(pdata, shape1 = exp(eta1),
    scale1 = exp(ceta1))
pdata <- transform(pdata, y1 = rperks(nn, shape = shape1, scale = scale1))
```
perspqrrvglm

Perspective plot for QRR-VGLMs

Description

Produces a perspective plot for a CQO model (QRR-VGLM). It is only applicable for rank-1 or rank-2 models with argument norrR = ~ 1.

Usage

```r
perspqrrvglm(x, varI.latvar = FALSE, refResponse = NULL, show.plot = TRUE,
            xlim = NULL, ylim = NULL, zlim = NULL,
            gridlength = if (Rank == 1) 301 else c(51,51),
            which.species = NULL,
            xlab = if (Rank == 1) "Latent Variable" else "Latent Variable 1",
            ylab = if (Rank == 1) "Expected Value" else "Latent Variable 2",
            zlab = "Expected value", labelSpecies = FALSE,
            stretch = 1.05, main = "", ticktype = "detailed",
            col = if (Rank == 1) par()$col else "white",
            lty = par()$lty, llwd = par()$lwd,
            add1 = FALSE, ...)
```

Arguments

- `x` Object of class "qrrvglm", i.e., a constrained quadratic ordination (CQO) object.
- `varI.latvar` Logical that is fed into `coef.qrrvglm`.
- `refResponse` Integer or character that is fed into `coef.qrrvglm`.
- `show.plot` Logical. Plot it?
- `xlim`, `ylim` Limits of the x- and y-axis. Both are numeric of length 2. See `par`.
- `zlim` Limits of the z-axis. Numeric of length 2. Ignored if rank is 1. See `par`.
- `gridlength` Numeric. The fitted values are evaluated on a grid, and this argument regulates the fineness of the grid. If Rank = 2 then the argument is recycled to length 2, and the two numbers are the number of grid points on the x- and y-axes respectively.
- `which.species` Numeric or character vector. Indicates which species are to be plotted. The default is to plot all of them. If numeric, it should contain values in the set \{1,2,\ldots ,S\} where \(S\) is the number of species.
`xlab`, `ylab`  Character caption for the x-axis and y-axis. By default, a suitable caption is found. See the `xlab` argument in `plot` or `title`.

`zlab`  Character caption for the z-axis. Used only if `Rank = 2`. By default, a suitable caption is found. See the `xlab` argument in `plot` or `title`.

`labelSpecies`  Logical. Whether the species should be labelled with their names. Used for `Rank = 1` only. The position of the label is just above the species’ maximum.

`stretch`  Numeric. A value slightly more than 1, this argument adjusts the height of the y-axis. Used for `Rank = 1` only.

`main`  Character, giving the title of the plot. See the `main` argument in `plot` or `title`.

`ticktype`  Tick type. Used only if `Rank = 2`. See `persp` for more information.

`col`  Color. See `persp` for more information.

`lty`  Line type. Rank-1 models only. See the `lty` argument of `par`.

`lwd`  Line width. Rank-1 models only. See the `lwd` argument of `par`.

`add`  Logical. Add to an existing plot? Used only for rank-1 models.

`...`  Arguments passed into `persp`. Useful arguments here include `theta` and `phi`, which control the position of the eye.

**Details**

For a rank-1 model, a perspective plot is similar to `lvplot.qrrglm` but plots the curves along a fine grid and there is no rugplot to show the site scores.

For a rank-2 model, a perspective plot has the first latent variable as the x-axis, the second latent variable as the y-axis, and the expected value (fitted value) as the z-axis. The result of a CQO is that each species has a response surface with elliptical contours. This function will, at each grid point, work out the maximum fitted value over all the species. The resulting response surface is plotted. Thus rare species will be obscured and abundant species will dominate the plot. To view rare species, use the `whichSpecies` argument to select a subset of the species.

A perspective plot will be performed if `noRRR = ~ 1`, and `Rank = 1` or `2`. Also, all the tolerance matrices of those species to be plotted must be positive-definite.

**Value**

For a rank-2 model, a list with the following components.

- `fitted`  A $(G_1 \times G_2)$ by $M$ matrix of fitted values on the grid. Here, $G_1$ and $G_2$ are the two values of `gridlength`.

- `latvar1grid`, `latvar2grid`  The grid points for the x-axis and y-axis.

- `max.fitted`  A $G_1$ by $G_2$ matrix of maximum of the fitted values over all species. These are the values that are plotted on the z-axis.

For a rank-1 model, the components `latvar2grid` and `max.fitted` are NULL.
Note

Yee (2004) does not refer to perspective plots. Instead, contour plots via `lvplot.qrrvglm` are used.

For rank-1 models, a similar function to this one is `lvplot.qrrvglm`. It plots the fitted values at the actual site score values rather than on a fine grid here. The result has the advantage that the user sees the curves as a direct result from a model fitted to data whereas here, it is easy to think that the smooth bell-shaped curves are the truth because the data is more of a distance away.

Author(s)

Thomas W. Yee

References


See Also

`persp`, `cqo`, `Coef.qrrvglm`, `lvplot.qrrvglm`, `par`, `title`.

Examples

```r
## Not run:
hspider[, 1:6] <- scale(hspider[, 1:6])  # Good idea when I.tolerances = TRUE
set.seed(111)
r1 <- cqo(cbind(Alopecce, Alopcune, Alopfabr, Arctlute, Arctper, Aulaalbi, Pardmont, Pardnigr, Pardpull, Trocterr) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + RefLLux,
         poissonff, data = hspider, trace = FALSE, I.tolerances = TRUE)
set.seed(111)  # r2 below is an ill-conditioned model
r2 <- cqo(cbind(Alopecce, Alopcune, Alopfabr, Arctlute, Arctper,
                 Aulaalbi, Pardmont, Pardnigr, Pardpull, Trocterr) ~
         WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + RefLLux,
         isd.lv = c(2.4, 1.0), Muxfactor = 3.0, trace = FALSE,
         poissonff, data = hspider, Rank = 2, eq.tolerances = TRUE)

sort(deviance(r1, history = TRUE))  # A history of all the fits
sort(deviance(r2, history = TRUE))  # A history of all the fits
if (deviance(r2) > 857) stop("suboptimal fit obtained")

persp(r1, xlim = c(-6, 5), col = 1:4, label = TRUE)
# Involves all species
persp(r2, xlim = c(-6, 5), ylim = c(-4, 5), theta = 10, phi = 20, zlim = c(0, 220))
# Omit the two dominant species to see what is behind them
persp(r2, xlim = c(-6, 5), ylim = c(-4, 5), theta = 10, phi = 20, zlim = c(0, 220),
      which = (1:10)[-c(8, 10)]);  # Use zlim to retain the original z-scale
```

## End(Not run)
pgamma.deriv

Derivatives of the Incomplete Gamma Integral

Description

The first two derivatives of the incomplete gamma integral.

Usage

pgamma.deriv(q, shape, tmax = 100)

Arguments

q, shape As in pgamma but these must be vectors of positive values only and finite.
tmax Maximum number of iterations allowed in the computation (per q value).

Details

Write \( x = q \) and \( \text{shape} = a \). The first and second derivatives with respect to \( q \) and \( a \) are returned. This function is similar in spirit to pgamma; define

\[
P(a, x) = \frac{1}{\Gamma(a)} \int_0^x t^{a-1} e^{-t} dt
\]

so that \( P(a, x) \) is pgamma(x, a). Currently a 6-column matrix is returned (in the future this may change and an argument may be supplied so that only what is required by the user is computed.)

The computations use a series expansion for \( a \leq x \leq 1 \) or \( x < a \), else otherwise a continued fraction expansion. Machine overflow can occur for large values of \( x \) when \( x \) is much greater than \( a \).

Value

The first 5 columns, running from left to right, are the derivatives with respect to: \( x, x^2, a, a^2, xa \). The 6th column is \( P(a, x) \) (but it is not as accurate as calling pgamma directly).

Note

If convergence does not occur then try increasing the value of tmax.

Yet to do: add more arguments to give greater flexibility in the accuracy desired and to compute only quantities that are required by the user.

Author(s)

T. W. Yee wrote the wrapper function to the Fortran subroutine written by R. J. Moore. The subroutine was modified to run using double precision. The original code came from http://lib.stat.cmu.edu/apstat/187.
References


See Also

`pgamma derivation unscaled`, `pgamma`.

Examples

```r
x <- seq(2, 10, length = 501)
head(ans <- pgamma.deriv(x, 2))
## Not run: par(mfrow = c(2, 3))
for (jay in 1:6)
  plot(x, ans[, jay], type = "l", col = "blue", cex.lab = 1.5,
       cex.axis = 1.5, las = 1, log = "x",
       main = colnames(ans)[jay], xlab = "q", ylab = "")
## End(Not run)
```

---

**pgamma.deriv.unscaled**  
*Derivatives of the Incomplete Gamma Integral (Unscaled Version)*

**Description**

The first two derivatives of the incomplete gamma integral with scaling.

**Usage**

```r
pgamma.deriv.unscaled(q, shape)
```

**Arguments**

- `q`, `shape`  
  As in `pgamma` and `pgamma.deriv` but these must be vectors of positive values only and finite.

**Details**

Define

\[ G(x, a) = \int_0^x t^{a-1} e^{-t} dt \]

so that \( G(x, a) \) is `pgamma(x, a)` * gamma(a). Write \( x = q \) and \( shape = a \). The 0th and first and second derivatives with respect to \( a \) of \( G \) are returned. This function is similar in spirit to `pgamma.deriv` but here there is no gamma function to scale things. Currently a 3-column matrix is returned (in the future this may change and an argument may be supplied so that only what is required by the user is computed.) This function is based on Wingo (1989).
The 3 columns, running from left to right, are the 0, 1, and 2th derivatives with respect to $a$.

These function seems inaccurate for $q = 1$ and $q = 2$; see the plot below.

T. W. Yee.

See also `truncweibull`.

`pgamma.deriv`, `pgamma`.

```r
x <- 3; aa <- seq(0.3, 0.4, by = 0.01)
ans.u <- pgamma.deriv.unscaled(x, aa)
head(ans.u)

## Not run: par(mfrow = c(1, 3))
for (jay in 1:3) {
  plot(aa, ans.u[, jay], type = "l", col = "blue", cex.lab = 1.5,
     cex.axis = 1.5, las = 1, main = colnames(ans.u)[jay],
     log = "", xlab = "shape", ylab = "")
  abline(h = 0, v = 1:2, lty = "dashed", col = "gray")  # Inaccurate at 1 and 2
}

## End(Not run)
```

Description

Plots a probability density function associated with a LMS quantile regression.

Usage

```r
plotdeplot.lmscreg(answer, y.arg, add.arg = FALSE,
    xlab = "", ylab = "density", xlim = NULL, ylim = NULL,
    lty.arg = par()$lty, col.arg = par()$col,
    lwd.arg = par()$lwd, ...)
```
**Arguments**

- **answer**: Output from functions of the form `deplot.??` where `??` is the name of the VGAM LMS family function, e.g., `lms.yjn`. See below for details.
- **y.arg**: Numerical vector. The values of the response variable at which to evaluate the density. This should be a grid that is fine enough to ensure the plotted curves are smooth.
- **add.arg**: Logical. Add the density to an existing plot?
- **xlab, ylab**: Caption for the x- and y-axes. See `par`.
- **xlim, ylim**: Limits of the x- and y-axes. See `par`.
- **lty.arg**: Line type. See the `lty` argument of `par`.
- **col.arg**: Line color. See the `col` argument of `par`.
- **lwd.arg**: Line width. See the `lwd` argument of `par`.
- **...**: Arguments passed into the `plot` function when setting up the entire plot. Useful arguments here include `main` and `las`.

**Details**

The above graphical parameters offer some flexibility when plotting the quantiles.

**Value**

The list `answer`, which has components

- **newdata**: The argument `newdata` above from the argument list of `deplot.lmscreg`, or a one-row data frame constructed out of the `x0` argument.
- **y**: The argument `y.arg` above.
- **density**: Vector of the density function values evaluated at `y.arg`.

**Note**

While the graphical arguments of this function are useful to the user, this function should not be called directly.

**Author(s)**

Thomas W. Yee

**References**


**See Also**

- `deplot.lmscreg`. 
Examples

```r
fit <- vgam(BMI ~ s(age, df = c(4,2)), lms.bcn(zero = 1), bmi.nz)
## Not run:  y = seq(15, 43, by = 0.25)
deplot(fit, x0 = 20, y = y, xlab = "BMI", col = "green", llwd = 2,
   main = "BMI distribution at ages 20 (green), 40 (blue), 60 (orange)")
deplot(fit, x0 = 40, y = y, add = TRUE, col = "blue", llwd = 2)
deplot(fit, x0 = 60, y = y, add = TRUE, col = "orange", llwd = 2) -> aa

names(aa@post$deplot)
aa@post$deplot$newdata
head(aa@post$deplot$y)
head(aa@post$deplot$density)
## End(Not run)
```

Description

The residuals of a QRR-VGLM are plotted for model diagnostic purposes.

Usage

```r
plotqrrvglm(object, rtype = c("response", "pearson", "deviance", "working"),
   ask = FALSE,
   main = paste(Rtype, "residuals vs latent variable(s)")
   xlab = "Latent Variable",
   i.tolerances = object@control$eq.tolerances, ...)
```

Arguments

- `object`: An object of class "qrrvglm".
- `rtype`: Character string giving residual type. By default, the first one is chosen.
- `ask`: Logical. If TRUE, the user is asked to hit the return key for the next plot.
- `main`: Character string giving the title of the plot.
- `xlab`: Character string giving the x-axis caption.
- `i.tolerances`: Logical. This argument is fed into Coef(object, i.tolerances = i.tolerances).
  ... Other plotting arguments (see `par`).

Details

Plotting the residuals can be potentially very useful for checking that the model fit is adequate.

Value

The original object.
Note

An ordination plot of a QRR-VGLM can be obtained by `lvplot.qrrvglm`.

Author(s)

Thomas W. Yee

References


See Also

`lvplot.qrrvglm`, `cqo`.

Examples

```r
# Not run:
# QRR-VGLM on the hunting spiders data
# This is computationally expensive
set.seed(111)  # This leads to the global solution
# hspider[, 1:6] <- scale(hspider[, 1:6])  # Standardize the environmental variables
pl <- cqo(cbind(Alopecce, Alocume, Alopfabr, Arctlute, Arctperi,
               Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
               Trocterr, Zoraspin) ~
              WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + Reflux,
              quasipoissonff, data = hspider, Crowlpositive = FALSE)
par(mfrow = c(3, 4))
plot(pl, rtype = "response", col = "blue", pch = 4, las = 1, main = "")

# End(Not run)
```

---

**plotqtplot.lmscreg**

*Quantile Plot for LMS Quantile Regression*

Description

Plots the quantiles associated with a LMS quantile regression.

Usage

```r
plotqtplot.lmscreg(fitted.values, object, newdata = NULL,
                   percentiles = object@misc$percentiles, lp = NULL,
                   add.arg = FALSE, y = if (length(newdata)) FALSE else TRUE,
                   spline.fit = FALSE, label = TRUE, size.label = 0.06,
                   xlab = NULL, ylab = "",
                   pch = par()$pch, pcex = par()$cex, pcol.arg = par()$col,
```
Arguments

fitted.values  Matrix of fitted values.

object         A VGAM quantile regression model, i.e., an object produced by modelling functions such as vglm and vgam with a family function beginning with "lms." e.g., lms.yjn.

newdata        Data frame at which predictions are made. By default, the original data are used.

percentiles    Numerical vector with values between 0 and 100 that specify the percentiles (quantiles). The default is to use the percentiles when fitting the model. For example, the value 50 corresponds to the median.

lp             Length of percentiles.

add.arg        Logical. Add the quantiles to an existing plot?

y              Logical. Add the response as points to the plot?

spline.fit     Logical. Add a spline curve to the plot?

label          Logical. Add the percentiles (as text) to the plot?

size.label     Numeric. How much room to leave at the RHS for the label. It is in percent (of the range of the primary variable).

xlab           Caption for the x-axis. See par.

ylab           Caption for the x-axis. See par.

pch            Plotting character. See par.

pcex           Character expansion of the points. See par.

col.arg        Color of the points. See the col argument of par.

xlim           Limits of the x-axis. See par.

ylim           Limits of the y-axis. See par.

lty.arg        Line type. Line type. See the lty argument of par.

col.arg        Color of the lines. See the col argument of par.

lwd.arg        Line width. See the lwd argument of par.

col.arg        Color of the text (if label is TRUE). See the col argument of par.

tadj           Text justification. See the adj argument of par.

...            Arguments passed into the plot function when setting up the entire plot. Useful arguments here include main and las.

Details

The above graphical parameters offer some flexibility when plotting the quantiles.

Value

The matrix of fitted values.
Note

While the graphical arguments of this function are useful to the user, this function should not be called directly.

Author(s)

Thomas W. Yee

References


See Also

qtplot.lmscreg.

Examples

## Not run:
fit <- vgam(BMI ~ s(age, df = c(4,2)), lms.bcn(zero = 1), data = bmi.nz)
qtplot(fit)
qtplot(fit, perc = c(25,50,75,95), lcol = "blue", tcol = "blue", llwd = 2)
## End(Not run)

plotrcim0

Main effects plot for a Row-Column Interaction Model (RCIM)

Description

Produces a main effects plot for Row-Column Interaction Models (RCIMs).

Usage

plotrcim0(object, centered = TRUE, which.plots = c(1, 2),
           hline0 = TRUE, hlty = "dashed", hcol = par()$col, hlwd = par()$lwd,
           rfirst = 1, cfirst = 1,
           rtype = "h", ctype = "h",
           rcex.lab = 1, rcex.axis = 1, rtick = FALSE,
           ccex.lab = 1, ccex.axis = 1, ctick = FALSE,
           rmain = "Row effects", rsub = "",
           rxlabel = "", rylab = "Row effects",
           cmain = "Column effects", csub = "",
           cxlab= "", cylab = "Column effects",
           rcol = par()$col, ccol = par()$col,
           no.warning = FALSE, ...)
Arguments

object
An rcim object. This should be of rank-0, i.e., main effects only and no interactions.

which.plots
Numeric, describing which plots are to be plotted. The row effects plot is 1 and the column effects plot is 2. Set the value 0, say, for no plots at all.

centered
Logical. If TRUE then the row and column effects are centered (but not scaled) by scale. If FALSE then the raw effects are used (of which the first are zero by definition).

hline0, hlty, hcol, hlwd
hline0 is logical. If TRUE then a horizontal line is plotted at 0 and the other arguments describe this line. Probably having hline0 = TRUE only makes sense when centered = TRUE.

rfirst, cfirst
rfirst is the level of row that is placed first in the row effects plot, etc.

rmain, cmain
Character. rmain is the main label in the row effects plot, etc.

rtype, ctype, rsub, csub
See the type and sub arguments of plot.

rxlab, rylab, cxlab, cylab
Character. For the row effects plot, rxlab is xlab and rylab is ylab; see par. Ditto for cxlab and cylab for the column effects plot.

rcex.lab, ccex.lab
Numeric. rcex.lab is cex for the row effects plot label, etc.

rcex.axis, ccex.axis
Numeric. rcex.axis is the cex argument for the row effects axis label, etc.

rtick, ctick
Logical. If rtick = TRUE then add ticks to the row effects plot, etc.

rcol, ccol
rcol give a colour for the row effects plot, etc.

no.warning
Logical. If TRUE then no warning is issued if the model is not rank-0.

... Arguments fed into both plot calls.

Details

This function plots the row and column effects of a rank-0 RCIM. As the result is a main effects plot of a regression analysis, its interpretation when centered = FALSE is relative to the baseline (reference level) of a row and column, and should also be considered in light of the link function used. Many arguments that start with "r" refer to the row effects plot, and "c" for the column effects plot.

Value

The original object with the post slot assigned additional information from the plot.

Note

This function should be only used to plot the object of rank-0 RCIM. If the rank is positive then it will issue a warning.

Using an argument ylim will mean the row and column effects are plotted on a common scale; see plot.window.
Author(s)
T. W. Yee, A. F. Hadi.

See Also
moffset Rcim, rcim.

Examples

alcoff.e <- moffset(alcoff, "6", "Mon", postfix = "*") # Effective day
fit0 <- rcim(alcoff.e, family = poissonff)
## Not run: par(oma = c(0, 0, 4, 0), mfrow = 1:2) # For all plots below too
ii <- plot(fit0, rcol = "blue", ccol = "orange",
          lwd = 4, ylim = c(-2, 2), # A common ylim
cylab = "Effective daily effects", rylab = "Hourly effects",
          rxlab = "Hour", cxlab = "Effective day")
ni@post # Endowed with additional information

## End(Not run)

# Negative binomial example
fit1 <- rcim(alcoff.e, negbinomial, trace = TRUE)
## Not run: plot(fit1, ylim = c(-2, 2))

# Univariate normal example
fit2 <- rcim(alcoff.e, uninormal, trace = TRUE)
## Not run: plot(fit2, ylim = c(-200, 400))

# Median-polish example
## Not run:
fit3 <- rcim(alcoff.e, alaplace(tau = 0.5), maxit = 1000, trace = FALSE)
plot(fit3, ylim = c(-200, 250))
## End(Not run)

# Zero-inflated Poisson example on "crashp" (no 0s in alcoff)
cbind(rowSums(crashp)) # Easy to see the data
cbind(colSums(crashp)) # Easy to see the data
fit4 <- rcim(Rcim(crashp, rbaseline = "5", cbaseline = "Sun"),
             zipoissonff, trace = TRUE)
## Not run: plot(fit4, ylim = c(-3, 3))
Usage

```r
plotvgam(x, newdata = NULL, y = NULL, residuals = NULL,
          rugplot = TRUE, se = FALSE, scale = 0, raw = TRUE,
          offset.arg = 0, deriv.arg = 0, overlay = FALSE,
          type.residuals = c("deviance", "working", "pearson", "response"),
          plot.arg = TRUE, which.term = NULL, which.cf = NULL,
          control = plotvgam.control(...), varxij = 1, ...)
```

Arguments

- `x` A fitted VGAM object, e.g., produced by `vgam`, `vglm`, or `rrvglm`.
- `newdata` Data frame. May be used to reconstruct the original data set.
- `y` Unused.
- `residuals` Logical. If `TRUE` then residuals are plotted. See `type.residuals`.
- `rugplot` Logical. If `TRUE` then a rug plot is plotted at the foot of each plot. These values are jittered to expose ties.
- `se` Logical. If `TRUE` then approximate ±2 pointwise standard error bands are included in the plot.
- `scale` Numerical. By default, each plot will have its own y-axis scale. However, by specifying a value, each plot's y-axis scale will be at least scale wide.
- `raw` Logical. If `TRUE` then the smooth functions are those obtained directly by the algorithm, and are plotted without having to premultiply with the constraint matrices. If `FALSE` then the smooth functions have been premultiply by the constraint matrices. The `raw` argument is directly fed into `predict.vgam()`.
- `offset.arg` Numerical vector of length `r`. These are added to the component functions. Useful for separating out the functions when `overlay` is `TRUE`. If `overlay` is `TRUE` and there is one covariate then using the intercept values as the offsets can be a good idea.
- `deriv.arg` Numerical. The order of the derivative. Should be assigned an small integer such as 0, 1, 2. Only applying to `s()` terms, it plots the derivative.
- `overlay` Logical. If `TRUE` then component functions of the same covariate are overlaid on each other. The functions are centered, so `offset.arg` can be useful when `overlay` is `TRUE`.
- `type.residuals` if `residuals` is `TRUE` then the first possible value of this vector, is used to specify the type of residual.
- `plot.arg` Logical. If `FALSE` then no plot is produced.
- `which.term` Character or integer vector containing all terms to be plotted, e.g., `which.term = c("s(age)", "s(height)")` or `which.term = c(2, 5, 9)`. By default, all are plotted.
- `which.cf` An integer-valued vector specifying which linear/additive predictors are to be plotted. The values must be from the set `{1,2,...,r}`. By default, all are plotted.
- `control` Other control parameters. See `plotvgam.control`.
- `...` Other arguments that can be fed into `plotvgam.control`. This includes line colors, line widths, line types, etc.
varxij
Positive integer. Used if \( x_{ij} \) of `vglm.control` was used, this chooses which inner argument the component is plotted against. This argument is related to `raw = TRUE` and terms such as `NS(dum1, dum2)` and constraint matrices that have more than one column. The default would plot the smooth against \( \text{dum1} \) but setting `varxij = 2` could mean plotting the smooth against \( \text{dum2} \). See the `VGAM` website for further information.

Details
In this help file \( M \) is the number of linear/additive predictors, and \( r \) is the number of columns of the constraint matrix of interest.

Many of plotvgam()'s options can be found in `plotvgam.control`, e.g., line types, line widths, colors.

Value
The original object, but with the `preplot` slot of the object assigned information regarding the plot.

Note
While plot(fit) will work if `class(fit)` is "vgam", it is necessary to use `plotvgam(fit)` explicitly otherwise.

plotvgam() is quite buggy at the moment.

Author(s)
Thomas W. Yee

See Also
`vgam`, `plotvgam.control`, `predict.vgam`, `plotvglm`, `vglm`.

Examples
```r
coalminers <- transform(coalminers, Age = (age - 42) / 5)
fit <- vgam(cbind(nBnW, nBW, BnW, BW) ~ s(Age),
            binom2.or(zero = NULL), data = coalminers)
# Not run: par(mfrow = c(1, 3))
plot(fit, se = TRUE, ylim = c(-3, 2), las = 1)
plot(fit, se = TRUE, which.cf = 1:2, lcol = "blue", scol = "orange",
     ylim = c(-3, 2))
plot(fit, se = TRUE, which.cf = 1:2, lcol = "blue", scol = "orange",
     overlay = TRUE)
# End(Not run)
```
plotvgam.control

Control Function for plotvgam()

Description

Provides default values for many arguments available for plotvgam().

Usage

plotvgam.control(which.cf = NULL,
                 xlim = NULL, ylim = NULL, lty = par()$lty,
                 slty = "dashed", pcex = par()$cex,
                 pch = par()$pch, pcol = par()$col,
                 lcol = par()$col, rcol = par()$col,
                 scol = par()$col, llwd = par()$lwd, slwd = par()$lwd,
                 add.arg = FALSE, one.at.a.time = FALSE,
                 .include.dots = TRUE, noxmean = FALSE, ...)

Arguments

which.cf Integer vector specifying which component functions are to be plotted (for each covariate). Must have values from the set \{1,2,\ldots,M\}.
xlim Range for the x-axis.
ylim Range for the y-axis.
lty Line type for the fitted functions (lines). Fed into \texttt{par(lty)}.
slty Line type for the standard error bands. Fed into \texttt{par(lty)}.
pcex Character expansion for the points (residuals). Fed into \texttt{par(cex)}.
pch Character used for the points (residuals). Same as \texttt{par(pch)}.
pcol Color of the points. Fed into \texttt{par(col)}.
lcol Color of the fitted functions (lines). Fed into \texttt{par(col)}.
rcol Color of the rug plot. Fed into \texttt{par(col)}.
scol Color of the standard error bands. Fed into \texttt{par(col)}.
llwd Line width of the fitted functions (lines). Fed into \texttt{par(lwd)}.
slwd Line width of the standard error bands. Fed into \texttt{par(lwd)}.
add.arg Logical. If \texttt{TRUE} then the plot will be added to an existing plot, otherwise a new plot will be made.
one.at.a.time Logical. If \texttt{TRUE} then the plots are done one at a time, with the user having to hit the return key between the plots.
.include.dots Not to be used by the user.
noxmean Logical. If \texttt{TRUE} then the point at the mean of \(x\), which is added when standard errors are specified and it thinks the function is linear, is not added. One might use this argument if \texttt{ylab} is specified.
...
Other arguments that may be fed into \texttt{par()}. In the above, \(M\) is the number of linear/additive predictors.
Details

The most obvious features of `plotvgam` can be controlled by the above arguments.

Value

A list with values matching the arguments.

Author(s)

Thomas W. Yee

References


See Also

`plotvgam`.

Examples

```r
plotvgam.control(lcol = c("red", "blue"), scol = "darkgreen", se = TRUE)
```

---

### plotvglm

**Plots for VGLMs**

**Description**

Currently not working, this function can be used to feed the object to the VGAM plotting function. In the future some diagnostic plots will be plotted.

**Usage**

```r
plotvglm(x, type = c("vlglm", "vgam"),
  newdata = NULL, y = NULL, residuals = NULL,
  rugplot = TRUE, se = FALSE, scale = 0, raw = TRUE,
  offset.arg = 0, deriv.arg = 0, overlay = FALSE,
  type.residuals = c("deviance", "working", "pearson", "response"),
  plot.arg = TRUE, which.term = NULL, which.cf = NULL,
  control = plotvgam.control(...), varxij = 1, ...)
```
Arguments

- **x**
  - Same as `plotvgam`.

- **type**
  - Default is the first choice. Currently the first choice gives an error (not written yet). If "vgam" then all the arguments are fed into `plotvgam`.

- **newdata, y, residuals, rugplot**
  - Same as `plotvgam`.

- **se, scale, raw, offset.arg**
  - Same as `plotvgam`.

- **deriv.arg, overlay, type.residuals**
  - Same as `plotvgam`.

- **plot.arg, which.term, which.cf, control**
  - Same as `plotvgam`.

- **...**, **varxij**
  - Same as `plotvgam`.

Details

Currently this function has not been written. When this is done some diagnostic plots based on residuals and hatvalues will be done. In the meanwhile, this function can be used to call the plotting function for `vgam` objects.

Value

Same as `plotvgam`.

See Also

`plotvgam, plotvgam.control, vglm`.

Examples

```
coalminers <- transform(coalminers, Age = (age - 42) / 5)
fit <- vglm(cbind(nBnW, nBW, BnW, BW) ~ sm.bs(Age),
            binom2.or(zero = NULL), data = coalminers)
## Not run: par(mfrow = c(1, 3))
plot(fit, type = "vgam", se = TRUE, ylim = c(-3, 2), las = 1)
plot(fit, type = "vgam", se = TRUE, which.cf = 1:2,
     lcol = "blue", scol = "orange", ylim = c(-3, 2))
plot(fit, type = "vgam", se = TRUE, which.cf = 1:2,
     lcol = "blue", scol = "orange", overlay = TRUE)
## End(Not run)
```
Description

The `pneumo` data frame has 8 rows and 4 columns. Exposure time is explanatory, and there are 3 ordinal response variables.

Usage

data(pneumo)

Format

This data frame contains the following columns:

- `exposure.time` a numeric vector, in years
- `normal` a numeric vector, counts
- `mild` a numeric vector, counts
- `severe` a numeric vector, counts

Details

These were collected from coalface workers. In the original data set, the two most severe categories were combined.

Source


References


See Also

cumulative.

Examples

# Fit the proportional odds model, p.179, in McCullagh and Nelder (1989)
pneumo <- transform(pneumo, let = log(exposure.time))
vglm(cbind(normal, mild, severe) ~ let, propodds, data = pneumo)
### poisson.points

**Poisson-points-on-a-plane/volume Distances Distribution**

**Description**

Estimating the density parameter of the distances from a fixed point to the u-th nearest point, in a plane or volume.

**Usage**

```r
poisson.points(ostatistic, dimension = 2, link = "loge",
               idensity = NULL, imethod = 1)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ostatistic</td>
<td>Order statistic. A single positive value, usually an integer. For example, the value 5 means the response are the distances of the fifth nearest value to that point (usually over many planes or volumes). Non-integers are allowed because the value 1.5 coincides with maxwell when <code>dimension = 2</code>. Note: if <code>ostatistic = 1</code> and <code>dimension = 2</code> then this VGAM family function coincides with rayleigh.</td>
</tr>
<tr>
<td>dimension</td>
<td>The value 2 or 3; 2 meaning a plane and 3 meaning a volume.</td>
</tr>
<tr>
<td>link</td>
<td>Parameter link function applied to the (positive) density parameter, called ( \lambda ) below. See Links for more choices.</td>
</tr>
<tr>
<td>idensity</td>
<td>Optional initial value for the parameter. A NULL value means a value is obtained internally. Use this argument if convergence failure occurs.</td>
</tr>
<tr>
<td>imethod</td>
<td>An integer with value 1 or 2 which specifies the initialization method for ( \lambda ). If failure to converge occurs try another value and/or else specify a value for idensity.</td>
</tr>
</tbody>
</table>

**Details**

Suppose the number of points in any region of area \( A \) of the plane is a Poisson random variable with mean \( \lambda A \) (i.e., \( \lambda \) is the density of the points). Given a fixed point \( P \), define \( D_1, D_2, \ldots \) to be the distance to the nearest point to \( P \), second nearest to \( P \), etc. This VGAM family function estimates \( \lambda \) since the probability density function for \( D_u \) is easily derived, \( u = 1, 2, \ldots \). Here, \( u \) corresponds to the argument ostatistic.

Similarly, suppose the number of points in any volume \( V \) is a Poisson random variable with mean \( \lambda V \) where, once again, \( \lambda \) is the density of the points. This VGAM family function estimates \( \lambda \) by specifying the argument ostatistic and using `dimension = 3`.

The mean of \( D_u \) is returned as the fitted values. Newton-Raphson is the same as Fisher-scoring.

**Value**

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`. 
Warning

Convergence may be slow if the initial values are far from the solution. This often corresponds to
the situation when the response values are all close to zero, i.e., there is a high density of points.

Formulae such as the means have not been fully checked.

Author(s)

T. W. Yee

See Also

poissonff, maxwell, rayleigh.

Examples

```r
pdata <- data.frame(y = rgamma(10, shape = exp(-1)))  # Not proper data!
ostat <- 2
fit <- vglm(y ~ 1, poisson.points(o.stat, 2), data = pdata,
             trace = TRUE, crit = "coef")
fit <- vglm(y ~ 1, poisson.points(o.stat, 3), data = pdata,
             trace = TRUE, crit = "coef")  # Slow convergence?
fit <- vglm(y ~ 1, poisson.points(o.stat, 3, iden = 1), data = pdata,
             trace = TRUE, crit = "coef")
head(fitted(fit))
with(pdata, mean(y))
coef(fit, matrix = TRUE)
Coef(fit)
```

Description

Family function for a generalized linear model fitted to Poisson responses. The dispersion pa-
rameters may be known or unknown.

Usage

```r
poissonff(link = "log", dispersion = 1, onedpar = FALSE, imu = NULL,
          imethod = 1, parallel = FALSE, zero = NULL, bred = FALSE,
          earg.link = FALSE)
```
Arguments

link  
Link function applied to the mean or means. See Links for more choices and information.

dispersion  
Dispersion parameter. By default, maximum likelihood is used to estimate the model because it is known. However, the user can specify dispersion = 0 to have it estimated, or else specify a known positive value (or values if the response is a matrix—one value per column).

onedpar  
One dispersion parameter? If the response is a matrix, then a separate dispersion parameter will be computed for each response (column), by default. Setting onedpar=TRUE will pool them so that there is only one dispersion parameter to be estimated.

parallel  
A logical or formula. Used only if the response is a matrix.

imu, imethod  
See CommonVGAMffArguments for more information.

zero  
An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \{1,2,...,M\}, where \(M\) is the number of columns of the matrix response. See CommonVGAMffArguments for more information.

bred, earg.link  
Details at CommonVGAMffArguments. Setting bred = TRUE should work for multiple responses and all VGAM link functions; it has been tested for loge, identity but further testing is required.

Details

\(M\) defined above is the number of linear/additive predictors.

If the dispersion parameter is unknown, then the resulting estimate is not fully a maximum likelihood estimate.

A dispersion parameter that is less/greater than unity corresponds to under-/over-dispersion relative to the Poisson model. Over-dispersion is more common in practice.

When fitting a Quadratic RR-VGLM (see cqo), the response is a matrix of \(M\), say, columns (e.g., one column per species). Then there will be \(M\) dispersion parameters (one per column of the response matrix) if dispersion = 0 and onedpar = FALSE.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, vgam, rrvglm, cqo, and cao.

Warning

With a multivariate response, assigning a known dispersion parameter for each response is not handled well yet. Currently, only a single known dispersion parameter is handled well.
**Note**

This function will handle a matrix response automatically.

The call `poissonff(dispersion=0, ...)` is equivalent to `quasipoissonff(...).` The latter was written so that R users of `quasipoisson()` would only need to add a “ff” to the end of the family function name.

Regardless of whether the dispersion parameter is to be estimated or not, its value can be seen from the output from the `summary()` of the object.

**Author(s)**

Thomas W. Yee

**References**


**See Also**

`links`, `quasipoissonff`, `genpoisson`, `zipoisson`, `skellam`, `mix2poisson`, `cens.poisson`, `ordpoisson`, `amlpoisson`, `inv.binomial`, `simulate.vlm`, `loge`, `polf`, `rrvglm`, `cquo`, `cao`, `binomialff`, `quasibinomialff`, `poisson`, `poisson.points`, `ruge`, `V1`.

**Examples**

```r
poissonff()

set.seed(123)
pdata <- data.frame(x2 = rnorm(nn <- 100))
pdata <- transform(pdata, y1 = rpois(nn, exp(1 + x2)),
                   y2 = rpois(nn, exp(1 + x2)))
(fit1 <- vglm(cbind(y1, y2) ~ x2, poissonff, data = pdata))
(fit2 <- vglm(y1 ~ x2, poissonff(bred = TRUE), data = pdata))
coef(fit1, matrix = TRUE)
coef(fit2, matrix = TRUE)

nn <- 200
cdata <- data.frame(x2 = rnorm(nn), x3 = rnorm(nn), x4 = rnorm(nn))
cdata <- transform(cdata, lv1 = 0 + x3 - 2*x4)
cdata <- transform(cdata, lambda1 = exp(3 - 0.5 * (lv1-0)^2),
                    lambda2 = exp(2 - 0.5 * (lv1-1)^2),
                    lambda3 = exp(2 - 0.5 * ((lv1+4)/2)^2))
cdata <- transform(cdata, y1 = rpois(nn, lambda1),
                   y2 = rpois(nn, lambda2),
                   y3 = rpois(nn, lambda3))
## Not run: lvplot(p1, y = TRUE, lcol = 2:4, pch = 2:4, pcol = 2:4, rug = FALSE)
```
PoissonPoints  

Poisson Points Distribution

Description
Density for the PoissonPoints distribution.

Usage
dpoisNpoints(x, lambda, ostatistic, dimension = 2, log = FALSE)

Arguments
- x: vector of quantiles.
- lambda: the mean density of points.
- ostatistic: positive values, usually integers.
- dimension: Either 2 and/or 3.
- log: Logical; if TRUE, the logarithm is returned.

Details
See poissonNpoints, the VGAM family function for estimating the parameters, for the formula of the probability density function and other details.

Value
dpoisNpoints gives the density.

See Also
poissonNpoints, dpois, Maxwell.

Examples
## Not run: lambda <- 1; xvec <- seq(0, 2, length = 400)
plot(xvec, dpoisNpoints(xvec, lambda, ostat = 1, dimension = 2),
     type = "l", las = 1, col = "blue",
     sub = "First order statistic",
     main = paste("PDF of PoissonPoints distribution with lambda = ", lambda, " and on the plane", sep = ""))
## End(Not run)
Description
Computes the Poisson-ordinal transformation, including its inverse and the first two derivatives.

Usage
\[
polf(\theta, \text{cutpoint} = \text{NULL}, \\
\quad \text{inverse} = \text{FALSE}, \text{deriv} = 0, \text{short} = \text{TRUE}, \text{tag} = \text{FALSE})
\]

Arguments
- \(\theta\): Numeric or character. See below for further details.
- \(\text{cutpoint}\): The cutpoints should be non-negative integers. If \(\text{polf}()\) is used as the link function in \text{cumulative} then one should choose \text{reverse} = \text{TRUE}, \text{parallel} = \text{TRUE}.
- \text{inverse}, \text{deriv}, \text{short}, \text{tag}
  Details at \text{Links}.

Details
The Poisson-ordinal link function (POLF) can be applied to a parameter lying in the unit interval. Its purpose is to link cumulative probabilities associated with an ordinal response coming from an underlying Poisson distribution. If the cutpoint is zero then a complementary log-log link is used. See \text{Links} for general information about \text{VGAM} link functions.

Value
See \citet{Yee2012} for details.

Warning
Prediction may not work on \text{vglm} or \text{vgam} etc. objects if this link function is used.

Note
Numerical values of \(\theta\) too close to 0 or 1 or out of range result in large positive or negative values, or maybe 0 depending on the arguments. Although measures have been taken to handle cases where \(\theta\) is too close to 1 or 0, numerical instabilities may still arise.

In terms of the threshold approach with cumulative probabilities for an ordinal response this link function corresponds to the Poisson distribution (see \text{poissonff}) that has been recorded as an ordinal response using known cutpoints.

Author(s)
Thomas W. Yee
References


See Also

Links, ordpoisson, poissonff, nbolf, golf, cumulative.

Examples

```r
polf("p", cutpoint = 2, short = FALSE)
polf("p", cutpoint = 2, tag = TRUE)

p <- seq(0.01, 0.99, by = 0.01)
y <- polf(p, cutpoint = 2)
y. <- polf(p, cutpoint = 2, deriv = 1)
max(abs(polf(y, cutpoint = 2, inv = TRUE) - p))  # Should be 0

## Not run: par(mfrow = c(2, 1), las = 1)
plot(p, y, type = "l", col = "blue", main = "polf()")
abline(h = 0, v = 0.5, col = "orange", lty = "dashed")

plot(p, y., type = "l", col = "blue",
     main = "(Reciprocal of) first POLF derivative")
## End(Not run)

# Rutherford and Geiger data
ruge <- data.frame(yy = rep(0:14,
                      times = c(57, 203, 383, 525, 532, 408, 273, 139, 45, 27, 10, 4, 0, 1, 1)))
with(ruge, length(yy))  # 2608 1/8-minute intervals
cutpoint <- 5
ruge <- transform(ruge, yy01 = ifelse(yy <= cutpoint, 0, 1))
fit <- vglm(yy01 ~ 1, binomialff(link = polf(cutpoint = cutpoint)), ruge)
coef(fit, matrix = TRUE)
exp(coef(fit))

# Another example
pdata <- data.frame(x2 = sort(runif(nn <- 1000)))
pdata <- transform(pdata, x3 = runif(nn))
pdata <- transform(pdata, mymu = exp(3 + 1 * x2 - 2 * x3))
pdata <- transform(pdata, y1 = rpois(nn, lambda = mymu))
cutpoints <- c(-Inf, 10, 20, Inf)
pdata <- transform(pdata, cuty = Cut(y1, breaks = cutpoints))
## Not run: with(pdata, plot(x2, x3, col = cuty, pch = as.character(cuty)))
with(pdata, table(cuty) / sum(table(cuty)))
fit <- vglm(cuty ~ x2 + x3, data = pdata, trace = TRUE,
            cumulative(reverse = TRUE, parallel = TRUE,
            link = polf(cutpoint = cutpoints[2:3]),
            multiple.responses = TRUE))
```
Polono

The Poisson Lognormal Distribution

Description

Density, distribution function and random generation for the Poisson lognormal distribution.

Usage

dpolono(x, meanlog = 0, sdlog = 1, bigx = 170, ...)  
ppolono(q, meanlog = 0, sdlog = 1,  
isOne = 1 - sqrt(.Machine$double.eps), ...)  
rpolono(n, meanlog = 0, sdlog = 1)

Arguments

x, q vector of quantiles.
n number of observations. If length(n) > 1 then the length is taken to be the number required.
meanlog, sdlog the mean and standard deviation of the normal distribution (on the log scale). They match the arguments in Lognormal.
bigx Numeric. This argument is for handling large values of x and/or when integrate fails. A first order Taylor series approximation [Equation (7) of Bulmer (1974)] is used at values of x that are greater or equal to this argument. For bigx = 10, he showed that the approximation has a relative error less than 0.001 for values of meanlog and sdlog “likely to be encountered in practice”. The argument can be assigned Inf in which case the approximation is not used.
isOne Used to test whether the cumulative probabilities have effectively reached unity.
... Arguments passed into integrate.

Details

The Poisson lognormal distribution is similar to the negative binomial in that it can be motivated by a Poisson distribution whose mean parameter comes from a right skewed distribution (gamma for the negative binomial and lognormal for the Poisson lognormal distribution).
Value
dpolono gives the density, ppolono gives the distribution function, and rpolono generates random deviates.

Note
By default, dpolono involves numerical integration that is performed using integrate. Consequently, computations are very slow and numerical problems may occur (if so then the use of . . . may be needed). Alternatively, for extreme values of \( x, \) meanlog, sdlog, etc., the use of bigx = Inf avoids the call to integrate, however the answer may be a little inaccurate.

For the maximum likelihood estimation of the 2 parameters a VGAM family function called polono(), say, has not been written yet.

Author(s)
T. W. Yee. Some anonymous soul kindly wrote ppolono() and improved the original dpolono().

References

See Also
lognormal, poissonff, negbinomial.

Examples
meanlog <- 0.5; sdlog <- 0.5; yy <- 0:19
sum(proby <- dpolono(yy, m = meanlog, sd = sdlog)) # Should be 1
max(abs(cumsum(proby) - ppolono(yy, m = meanlog, sd = sdlog))) # Should be 0

## Not run: opar = par(no.readonly = TRUE)
par(mfrow = c(2, 2))
plot(yy, proby, type = "h", col = "blue", ylab = "P[Y=y]", log = "",
     main = paste("Poisson lognormal(m = ", meanlog,


# More extreme values; use the approximation and plot on a log scale
(meanlog <- 0.1; sdlog <- 0.5; yy <- 0:19)
sum(proby <- dpolono(yy, m = meanlog, sd = sdlog, bigx = 100)) # Should be 1

## Random number generation
table(y <- rpolono(n = 1000, m = meanlog, sd = sdlog))
hist(y, breaks = ((-1):max(y))+0.5, prob = TRUE, border = "blue")
par(opar)
## End(Not run)
Description

Fits a GLM-/GAM-like model to multiple Bernoulli responses where each row in the capture history matrix response has at least one success (capture). Capture history behavioural effects are accommodated.

Usage

```r
posbernoulli.b(link = "logit", drop.b = FALSE - 1,
    type.fitted = c("likelihood.cond", "mean.uncond"), I2 = FALSE,
    ipcapture = NULL, iprecapture = NULL,
    p.small = 1e-4, no.warning = FALSE)
```

Arguments

- `link, drop.b, ipcapture, iprecapture`
  
  See `CommonVGAMffArguments` for information about these arguments. By default the parallelism assumption does not apply to the intercept. With an intercept-only model setting `drop.b = TRUE ~ 1` results in the \( M_0/M_h \) model.

- `I2`
  Logical. This argument is used for terms that are not parallel. If `TRUE` then the constraint matrix `diag(2)` (the general default constraint matrix in `VGAM`) is used, else `cbind(0:1, 1)`. The latter means the first element/column corresponds to the behavioural effect. Consequently it and its standard error etc. can be accessed directly without subtracting two quantities.

- `type.fitted`
  Details at `posbernoulli.tb`.

- `p.small, no.warning`
  See `posbernoulli.t`.

Details

This model (commonly known as \( M_b/M_{bh} \) in the capture–recapture literature) operates on a capture history matrix response of 0s and 1s \((n \times \tau)\). See `posbernoulli.t` for details, e.g., common assumptions with other models. Once an animal is captured for the first time, it is marked/tagged so that its future capture history can be recorded. The effect of the recapture probability is modelled through a second linear/additive predictor. It is well-known that some species of animals are affected by capture, e.g., trap-shy or trap-happy. This `VGAM` family function does allow the capture history to be modelled via such behavioural effects. So does `posbernoulli.tb` but `posbernoulli.t` cannot.

The number of linear/additive predictors is \( M = 2 \), and the default links are \((\text{logit } p_c, \text{logit } p_r)^T\) where \( p_c \) is the probability of capture and \( p_r \) is the probability of recapture. The fitted value returned is of the same dimension as the response matrix, and depends on the capture history: prior to being first captured, it is \( p_{\text{capture}} \). Afterwards, it is \( p_{\text{precapture}} \).
By default, the constraint matrices for the intercept term and the other covariates are set up so that $p_r$ differs from $p_c$ by a simple binary effect, on a logit scale. However, this difference (the behavioural effect) is more directly estimated by having $I2 = \text{FALSE}$. Then it allows an estimate of the trap-happy/trap-shy effect; these are positive/negative values respectively. If $I2 = \text{FALSE}$ then the (nonstandard) constraint matrix used is $\text{cbind}(0:1, 1)$, meaning the first element can be interpreted as the behavioural effect.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

The dependent variable is not scaled to row proportions. This is the same as posbernoulli.t and posbernoulli.tb but different from posbinomial and binomialff.

Author(s)

Thomas W. Yee.

References

See posbernoulli.t.

See Also

posbernoulli.t and posbernoulli.tb (including estimating $N$), deermice, dposbern, rposbern, posbinomial, aux.posbernoulli.t.prinia.

Examples

# deermice data

# Fit a M_b model
M.b <- vglm(cbind(y1, y2, y3, y4, y5, y6) ~ 1,
            posbernoulli.b, data = deermice, trace = TRUE)
coef(M.b)["(Intercept):1"] # Behavioural effect on the logit scale
coef(M.b, matrix = TRUE)
constraints(M.b, matrix = TRUE)
summary(M.b, presid = FALSE)

# Fit a M_bh model
M.bh <- vglm(cbind(y1, y2, y3, y4, y5, y6) ~ sex + weight,
             posbernoulli.b, data = deermice, trace = TRUE)
coef(M.bh)["(Intercept):1"] # Behavioural effect on the logit scale
constraints(M.bh) # (2,1) element of "(Intercept)" is for the behavioural effect
summary(M.bh, presid = FALSE) # Significant positive (trap-happy) behavioural effect
# Approx. 95 percent confidence for the behavioural effect:
SE.M.bh <- coef(summary(M.bh))["(Intercept):1", "Std. Error"]
Positive Bernoulli Family Function with Time Effects

```r
coeff(M.bh)["(Intercept):1"] + c(-1, 1) * 1.96 * SE.M.bh

# Fit a M.h model
M.h <- vglm(cbind(y1, y2, y3, y4, y5, y6) ~ sex + weight,
            posbernoulli(b(0) = TRUE - sex + weight),
            data = deer.mice, trace = TRUE)
coeff(M.h, matrix = TRUE)
constraints(M.h, matrix = TRUE)
summary(M.h, presid = FALSE)

# Fit a M.0 model
M.0 <- vglm(cbind(y1 + y2 + y3 + y4 + y5 + y6,
                   6 - y1 - y2 - y3 - y4 - y5 - y6) ~ 1,
            poisson.binomial, data = deer.mice, trace = TRUE)
coeff(M.0, matrix = TRUE)
summary(M.0, presid = FALSE)

# Simulated data set
set.seed(123); nTimePts <- 5; N <- 1000 # N is the popn size
pdata <- rposbern(n = N, nTimePts = nTimePts, pvars = 2, is.popn = TRUE)
nrow(pdata) # Less than N (because some animals were never captured)
# The truth: xcoeffs are c(-2, 1, 2) and cap.eff = +1
M.bh.2 <- vglm(cbind(y1, y2, y3, y4, y5) ~ x2,
               posbernoulli.b, data = pdata, trace = TRUE)
coeff(M.bh.2)
coeff(M.bh.2, matrix = TRUE)
constraints(M.bh.2, matrix = TRUE)
summary(M.bh.2, presid = FALSE)
head(depvar(M.bh.2)) # Capture history response matrix
head(M.bh.2$extra$cap.hist1) # Info on its capture history
head(M.bh.2$extra$cap1) # When it was first captured
head(fitted(M.bh.2)) # Depends on capture history
(trap.eff <- coeff(M.bh.2)["(Intercept):1"] # Should be +1
head(model.matrix(M.bh.2, type = "vlm"), 21)
head(pdata)
summary(pdata)
dim(depvar(M.bh.2))
vcov(M.bh.2)
M.bh.2$extra$N.hat # Estimate of the population size; should be about N
M.bh.2$extra$SE.N.hat # SE of the estimate of the population size
# An approximate 95 percent confidence interval:
round(M.bh.2$extra$N.hat + c(-1, 1) * 1.96 * M.bh.2$extra$SE.N.hat, 1)
```

---

**positivebernoulli.t**
Description

Fits a GLM/GAM-like model to multiple Bernoulli responses where each row in the capture history matrix response has at least one success (capture). Sampling occasion effects are accommodated.

Usage

```R
posbernoulli.t(link = "logit", parallel.t = FALSE ~ 1, iprob = NULL,
               p.small = 1e-4, no.warning = FALSE)
```

Arguments

- `link`, `iprob`, `parallel.t`
  See `CommonVGAMffArguments` for information. By default, the parallelism assumption does not apply to the intercept. Setting `parallel.t = FALSE ~ -1`, or equivalently `parallel.t = FALSE ~ 0`, results in the $M_0/M_h$ model.

- `p.small`, `no.warning`
  A small probability value used to give a warning for the Horvitz–Thompson estimator. Any estimated probability value less than `p.small` will result in a warning, however, setting `no.warning = TRUE` will suppress this warning if it occurs. This is because the Horvitz-Thompson estimator is the sum of the reciprocal of such probabilities, therefore any probability that is too close to 0 will result in an unstable estimate.

Details

These models (commonly known as $M_t$ or $M_{th}$ (no prefix $h$ means it is an intercept-only model) in the capture–recapture literature) operate on a capture history matrix response of 0s and 1s ($n \times \tau$). Each column is a sampling occasion where animals are potentially captured (e.g., a field trip), and each row is an individual animal. Capture is a 1, else a 0. No removal of animals from the population is made (closed population), e.g., no immigration or emigration. Each row of the response matrix has at least one capture. Once an animal is captured for the first time, it is marked/tagged so that its future capture history can be recorded. Then it is released immediately back into the population to remix. It is released immediately after each recapture too. It is assumed that the animals are independent and that, for a given animal, each sampling occasion is independent. And animals do not lose their marks/tags, and all marks/tags are correctly recorded.

The number of linear/additive predictors is equal to the number of sampling occasions, i.e., $M = \tau$, say. The default link functions are $(\logit p_1, \ldots, \logit p_\tau)^T$ where each $p_j$ denotes the probability of capture at time point $j$. The fitted value returned is a matrix of probabilities of the same dimension as the response matrix.

A conditional likelihood is maximized here using Fisher scoring. Each sampling occasion has a separate probability that is modelled here. The probabilities can be constrained to be equal by setting `parallel.t = FALSE ~ 0`; then the results are effectively the same as `posbinomial` except the binomial constants are not included in the log-likelihood. If `parallel.t = TRUE ~ 0` then each column should have at least one 1 and at least one 0.

It is well-known that some species of animals are affected by capture, e.g., trap-shy or trap-happy. This `VGAM` family function does not allow any behavioral effect to be modelled (`posbernoulli.b` and `posbernoulli.tb` do) because the denominator of the likelihood function must be free of behavioral effects.
Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Upon fitting the extra slot has a (list) component called N.hat which is a point estimate of the population size \( N \) (it is the Horvitz-Thompson (1952) estimator). And there is a component called SE.N.hat containing its standard error.

Note

The weights argument of vglm need not be assigned, and the default is just a matrix of ones.

Fewer numerical problems are likely to occur for parallel.t = TRUE. Data-wise, each sampling occasion may need at least one success (capture) and one failure. Less stringent conditions in the data are needed when parallel.t = TRUE. Ditto when parallelism is applied to the intercept too.

The response matrix is returned unchanged; i.e., not converted into proportions like posbinomial. If the response matrix has column names then these are used in the labelling, else prob1, prob2, etc. are used.

Using AIC() or BIC() to compare posbernoulli.t, posbernoulli.b, posbernoulli.tb models with a posbinomial model requires posbinomial(omit.constant = TRUE) because one needs to remove the normalizing constant from the log-likelihood function. See posbinomial for an example.

Author(s)

Thomas W. Yee.

References


See Also

posbernoulli.b, posbernoulli.tb, Select, deermice, Huggins89tablel, Huggins89.t1, dposbern, rposbern, posbinomial, AICvlm, BICvlm, prinia.

Examples

```r
M.t <- vglm(cbind(y1, y2, y3, y4, y5, y6) ~ 1, posbernoulli.t,
            data = deermice, trace = TRUE)
coef(M.t, matrix = TRUE)
```
Constraints

```
M.h.1 <- vglm(Select(deermice, "y") ~ sex + weight, trace = TRUE,
    posbernoulli.tb(parallel.t = FALSE ~ -1), data = deermice)
coef(M.h.1, matrix = TRUE)
constraints(M.h.1)
summary(M.h.1, presid = FALSE)
head(depvar(M.h.1))  # Response capture history matrix
dim(depvar(M.h.1))
```

```
M.th.2 <- vglm(cbind(y1, y2, y3, y4, y5, y6) ~ sex + weight, trace = TRUE,
    posbernoulli.tb(parallel.t = FALSE), data = deermice)
lrtest(M.h.1, M.th.2)  # Test the parallelism assumption wrt sex and weight
coef(M.th.2)
constraints(M.th.2)
summary(M.th.2, presid = FALSE)
head(model.matrix(M.th.2), type = "vlm"), 21)
```

```
M.th.2@extra$N.hat  # Estimate of the population size; should be about N
M.th.2@extra$SE.N.hat # SE of the estimate of the population size
# An approximate 95 percent confidence interval:
round(M.th.2@extra$N.hat + c(-1, 1) * 1.96 * M.th.2@extra$SE.N.hat, 1)
```

```
# Fit a M.h model, effectively the parallel M_t model, using posbinomial()
deermice <- transform(deermice, ysum = y1 + y2 + y3 + y4 + y5 + y6,
    tau = 6)
M.h.3 <- vglm(cbind(ysum, tau ~ ysum) ~ sex + weight,
    posbinomial(omit.constant = TRUE), data = deermice, trace = TRUE)
max(abs(coef(M.h.1) - coef(M.h.3)))  # Should be zero
logLik(M.h.3) - logLik(M.h.1)  # Difference is due to the binomial constants
```

---

**Description**

Fits a GLM/GAM-like model to multiple Bernoulli responses where each row in the capture history matrix response has at least one success (capture). Sampling occasion effects and behavioural effects are accommodated.

**Usage**

```
posbernoulli.tb(link = "logit", parallel.t = FALSE ~ 1,
    parallel.b = FALSE ~ 0, drop.b = FALSE ~ 1,
    type.fitted = c("likelihood.cond", "mean.uncond"),
    imethod = 1, iprob = NULL,
    p.small = 1e-4, no.warning = FALSE,
    ridge.constant = 0.01, ridge.power = -4)
```
Arguments

link, imethod, iprob
See CommonVGAMffArguments for information.

parallel.t, parallel.b, drop.b
A logical, or formula with a logical as the response. See CommonVGAMffArguments for information. The parallel.-type arguments specify whether the constraint matrices have a parallelism assumption for the temporal and behavioural effects. Argument parallel.t means parallel with respect to time, and matches the same argument name in posbernoulli.t.

Suppose the model is intercept-only. Setting parallel.t = FALSE ~ 0 results in the $M_0$ model. Setting drop.b = FALSE ~ 0 results in the $M_I$ model because it drops columns off the constraint matrices corresponding to any behavioural effect. Setting parallel.t = FALSE ~ 0 and setting parallel.b = FALSE ~ 0 results in the $M_0$ model. Setting parallel.t = FALSE ~ 0, parallel.b = FALSE ~ 0 and drop.b = FALSE ~ 0 results in the $M_0$ model. Note the default for parallel.t and parallel.b may be unsuitable for data sets which have a large $\tau$ because of the large number of parameters; it might be too flexible. If it is desired to have the behaviour affect some of the other covariates then set drop.b = TRUE ~ 0.

The default model has a different intercept for each sampling occasion, a time-parallelism assumption for all other covariates, and a dummy variable representing a single behavioural effect (also in the intercept).

The most flexible model is to set parallel.b = TRUE ~ 0, parallel.t = TRUE ~ 0 and drop.b = TRUE ~ 0. This means that all possible temporal and behavioural effects are estimated, for the intercepts and other covariates. Such a model is not recommended; it will contain a lot of parameters.

type.fitted
Character, one of the choices for the type of fitted value returned. The default is the first one. Partial matching is okay. For "likelihood.cond": the probability defined by the conditional likelihood. For "mean.uncond": the unconditional mean, which should agree with colMeans applied to the response matrix for intercept-only models.

ridge.constant, ridge.power
Determines the ridge parameters at each IRLS iteration. They are the constant and power (exponent) for the ridge adjustment for the working weight matrices (the capture probability block matrix, hence the first $\tau$ diagonal values). At iteration $a$ of the IRLS algorithm a positive value is added to the first $\tau$ diagonal elements of the working weight matrices to make them positive-definite. This adjustment is the mean of the diagonal elements of $wz$ multiplied by $K \times a^p$ where $K$ is ridge.constant and $p$ is ridge.power. This is always positive but decays to zero as iterations proceed (provided $p$ is negative etc.).

p.small, no.warning
See posbernoulli.t.

Details

This model (commonly known as $M_{tb}/M_{tbh}$ in the capture–recapture literature) operates on a response matrix of 0s and 1s ($n \times \tau$). See posbernoulli.t for information that is in common. It allows time and behavioural effects to be modelled.
Evidently, the expected information matrix (EIM) seems not of full rank (especially in early iterations), so ridge.constant and ridge.power are used to try fix up the problem. The default link functions are \((\text{logit } p_1, \ldots, \text{logit } p_{\tau_1}, \text{logit } p_{\tau_2}, \ldots, \text{logit } p_{\tau_T})^T\) where the subscript \(c\) denotes capture, the subscript \(r\) denotes recapture, and it is not possible to recapture the animal at sampling occasion 1. Thus \(M = 2\tau - 1\). The parameters are currently prefixed by \(pc\) capture and \(prec\) precapture for the capture and recapture probabilities. This VGAM family function may be further modified in the future.

Value

An object of class "vglmff" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, and \texttt{vgam}.

Note

It is a good idea to apply the parallelism assumption to each sampling occasion except possibly with respect to the intercepts. Also, a simple behavioural effect such as being modelled using the intercept is recommended; if the behavioural effect is not parallel and/or allowed to apply to other covariates then there will probably be too many parameters, and hence, numerical problems. See \texttt{M_tbh.1} below.

It is a good idea to monitor convergence. Simpler models such as the \(M_0/M_h\) models are best fitted with \texttt{posbernoulli.t} or \texttt{posbernoulli.b} or \texttt{posbinomial}.

Author(s)

Thomas W. Yee.

References

See \texttt{posbernoulli.t}.

See Also

\texttt{posbernoulli.b} (including \texttt{N.hat}), \texttt{posbernoulli.t}, \texttt{posbinomial}, \texttt{Select}, \texttt{fill}, \texttt{Huggins89table1}, \texttt{Huggins89.t1}, \texttt{deermice}, \texttt{prinia}.

Examples

```r
# Not run:
# Example 1: simulated data
nTimePts <- 5  # (aka tau == # of sampling occasions)
nnn <- 1000    # Number of animals
pdata <- rposbern(n = nnn, nTimePts = nTimePts, pvars = 2)
dim(pdata); head(pdata)

M_tbh.1 <- vglm(cbind(y1, y2, y3, y4, y5) ~ x2,
    posbernoulli.tb, data = pdata, trace = TRUE)
coef(M_tbh.1) # First element is the behavioural effect
ccoef(M_tbh.1, matrix = TRUE)
constraints(M_tbh.1, matrix = TRUE)
summary(M_tbh.1, presid = FALSE) # Standard errors are approximate
```
posbernUC

Positive Bernoulli Sequence Model

Description

Density, and random generation for multiple Bernoulli responses where each row in the response matrix has at least one success.

Usage

```r
rposbern(n, nTimePts = 5, pvars = length(xcoeff), xcoeff = c(-2, 1, 2),
  Xmatrix = NULL, cap.eff = 1, is.popn = FALSE,
  link = "logit", earg.link = FALSE)
dposbern(x, prob, prob0 = prob, log = FALSE)
```
Arguments

- **x**: response vector or matrix. Should only have 0 and 1 values, at least two columns, and each row should have at least one 1.
- **nTimePts**: Number of sampling occasions. Called \( \tau \) in `posbernoulli.b` and `posbernoulli.t`.
- **n**: number of observations. Usually a single positive integer, else the length of the vector is used. See argument `is.popn`.
- **is.popn**: Logical. If `TRUE` then argument `n` is the population size and what is returned may have substantially less rows than `n`. That is, if an animal has at least one one in its sequence then it is returned, else that animal is not returned because it never was captured.
- **Xmatrix**: Optional \( X \) matrix. If given, the \( X \) matrix is not generated internally.
- **cap.effect**: Numeric, the capture effect. Added to the linear predictor if captured previously. A positive or negative value corresponds to a trap-happy and trap-shy effect respectively.
- **pvars**: Number of other numeric covariates that make up the linear predictor. Labelled \( x_1, x_2, \ldots \), where the first is an intercept, and the others are independent standard `runif` random variates. The first `pvars` elements of `xcoeff` are used.
- **xcoeff**: The regression coefficients of the linear predictor. These correspond to \( x_1, x_2, \ldots \), and the first is for the intercept. The length of `xcoeff` must be at least `pvars`.
- **link, earg.link**: The former is used to generate the probabilities for capture at each occasion. Other details at `CommonVGAMffArguments`.
- **prob, prob0**: Matrix of probabilities for the numerator and denominators respectively. The default does *not* correspond to the \( M_b \) model since the \( M_b \) model has a denominator which involves the capture history.
- **log**: Logical. Return the logarithm of the answer?

Details

The form of the conditional likelihood is described in `posbernoulli.b` and/or `posbernoulli.t` and/or `posbernoulli.tb`. The denominator is equally shared among the elements of the matrix \( x \).

Value

`rposbern` returns a data frame with some attributes. The function generates random deviates (\( \tau \) columns labelled \( y_1, y_2, \ldots \)) for the response. Some indicator columns are also included (those starting with `ch` are for previous capture history). The default setting corresponds to a \( M_{bh} \) model that has a single trap-happy effect. Covariates \( x_1, x_2, \ldots \) have the same affect on capture/recapture at every sampling occasion (see the argument parallel.t in, e.g., `posbernoulli.tb`). The function `dposbern` gives the density.

Note

The `r`-type function is experimental only and does not follow the usual conventions of `r`-type R functions. It may change a lot in the future. The `d`-type function is more conventional and is less likely to change.
Author(s)
Thomas W. Yee.

See Also
posbernoulli.tb, posbernoulli.b, posbernoulli.t.

Examples
rposbern(n = 10)
attributes(pdata <- rposbern(n = 100))
M.bh <- vglm(cbind(y1, y2, y3, y4, y5) ~ x2 + x3, posbernoulli.b(I2 = FALSE),
            data = pdata, trace = TRUE)
constraints(M.bh)
summary(M.bh)

Description
Density, distribution function, quantile function and random generation for the positive-binomial
distribution.

Usage
dposbinom(x, size, prob, log = FALSE)
pposbinom(q, size, prob)
qposbinom(p, size, prob)
rposbinom(n, size, prob)

Arguments
x, q vector of quantiles.
p vector of probabilities.
n number of observations. Fed into runif.
size number of trials. It is the \(N\) symbol in the formula given in posbinomial.
prob probability of success on each trial. Should be in \((0, 1)\).
log See dbinom.

Details
The positive-binomial distribution is a binomial distribution but with the probability of a zero being
zero. The other probabilities are scaled to add to unity. The mean therefore is
\[
\mu/(1 - (1 - \mu)^N)
\]

where \(\mu\) is the argument prob above. As \(\mu\) increases, the positive-binomial and binomial distributions become more similar. Unlike similar functions for the binomial distribution, a zero value of
\texttt{prob} is not permitted here.
Value

dposbinom gives the density, pposbinom gives the distribution function, qposbinom gives the quantile function, and rposbinom generates random deviates.

Note

For dposbinom(), if arguments size or prob equal 0 then a NaN is returned.
The family function posbinomial estimates the parameters by maximum likelihood estimation.

Author(s)
T. W. Yee.

See Also

posbinomial, dposbern, zabinomial, zibinomial, rbinom.

Examples

```r
prob <- 0.2; size <- 10
table(y <- rposbinom(n = 1000, size, prob))
mean(y)  # Sample mean
size * prob / (1 - (1 - prob)^size)  # Population mean

(ii <- dposbinom(0:size, size, prob))
cumsum(ii) - pposbinom(0:size, size, prob)  # Should be 0s
table(rposbinom(100, size, prob))

table(qposbinom(runif(1000), size, prob))
round(dposbinom(1:10, size, prob) * 1000)  # Should be similar

## Not run: barplot(rbind(dposbinom(x = 0:size, size, prob),
  dbinom(x = 0:size, size, prob)),
  beside = TRUE, col = c("blue", "green"),
  main = paste("Positive-binomial", size, ",", prob, ") (blue) vs",
  " Binomial", size, ",", prob, ") (green)", sep = ""),
  names.arg = as.character(0:size), las = 1)
## End(Not run)

# Simulated data example
nn <- 1000; sizeval1 <- 10; sizeval2 <- 20
pdata <- data.frame(x = seq(0, 1, length = nn))
pdata <- transform(pdata, prob1 = logit(-2 + 2 * x, inverse = TRUE),
  prob2 = logit(-1 + 1 * x, inverse = TRUE),
  sizev1 = rep(sizeval1, len = nn),
  sizev2 = rep(sizeval2, len = nn))
pdata <- transform(pdata, y1 = rposbinom(nn, size = sizev1, prob = prob1),
  y2 = rposbinom(nn, size = sizev2, prob = prob2))
with(pdata, table(y1))
with(pdata, table(y2))
```
posbinomial

Description
Fits a positive binomial distribution.

Usage
posbinomial(link = "logit", multiple.responses = FALSE, parallel = FALSE, omit.constant = FALSE, p.small = 1e-4, no.warning = FALSE, zero = NULL)

Arguments
link, multiple.responses, parallel, zero
Details at CommonVGAMffArguments.
omit.constant Logical. If TRUE then the constant (lchoose(size, size * yprop) is omitted from the loglikelihood calculation. If the model is to be compared using AIC() or BIC() (see AICvlm or BICvlm) to the likes of posbernoulli.tb etc. then it is important to set omit.constant = TRUE because all models then will not have any normalizing constants in the likelihood function. Hence they become comparable. This is because the $M_0$ Otis et al. (1978) model coincides with posbinomial(). See below for an example. Also see posbernoulli.t regarding estimating the population size (N.hat and SE.N.hat) if the number of trials is the same for all observations.
p.small, no.warning
See posbernoulli.t.

Details
The positive binomial distribution is the ordinary binomial distribution but with the probability of zero being zero. Thus the other probabilities are scaled up (i.e., divided by $1 - P(Y = 0)$). The fitted values are the ordinary binomial distribution fitted values, i.e., the usual mean.

In the capture–recapture literature this model is called the $M_0$ if it is an intercept-only model. Otherwise it is called the $M_h$ when there are covariates. It arises from a sum of a sequence of $\tau$-Bernoulli random variates subject to at least one success (capture). Here, each animal has the same probability of capture or recapture, regardless of the $\tau$ sampling occasions. Independence between animals and between sampling occasions etc. is assumed.

# Multivariate response
fit2 <- vglm(cbind(y1, y2) ~ x2, posbinomial(multiple.responses = TRUE),
            trace = TRUE, data = pdata, weight = cbind(sizev1, sizev2))
coef(fit2, matrix = TRUE)
Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

Under- or over-flow may occur if the data is ill-conditioned.

Note

The input for this family function is the same as binomialff.

If multiple.responses = TRUE then each column of the matrix response should be a count (the number of successes), and the weights argument should be a matrix of the same dimension as the response containing the number of trials. If multiple.responses = FALSE then the response input should be the same as binomialff.

Yet to be done: a quasi.posbinomial() which estimates a dispersion parameter.

Author(s)

Thomas W. Yee

References


See Also

posbernoulli.b, posbernoulli.t, posbernoulli.tb, binomialff, AICvlm, BICvlm, simulate.vlm.

Examples

```r
# Number of albinotic children in families with 5 kids (from Patil, 1962),
albinos <- data.frame(y = c(rep(1, 25), rep(2, 23), rep(3, 10), 4, 5),
                      n = rep(5, 60))
fit1 <- vglm(cbind(y, n-y) ~ 1, posbinomial, albinos, trace = TRUE)
summary(fit1)
Coef(fit1) # = MLE of p = 0.3088
head(fitted(fit1))
sqrt(vcov(fit1, untransform = TRUE)) # SE = 0.0322

# Fit a M.0 model (Otis et al. 1978) to the deermice data,
M.0 <- vglm(cbind(y1 + y2 + y3 + y4 + y5 + y6,
                    6 - y1 - y2 - y3 - y4 - y5 - y6) ~ 1, trace = TRUE,
                      posbinomial(omit.constant = TRUE), data = deermice)
coef(M.0, matrix = TRUE)
```
Coefficients

summary(M.0)

c( N.hat = M.0@extra$N.hat, SE.N.hat = M.0@extra$SE.N.hat) # Since tau = 6, i.e., 6 Bernoulli trials per observation is the same for each observation

# Compare it to the M_b using AIC and BIC
M_b <- vglm(cbind(y1, y2, y3, y4, y5, y6) ~ 1, trace = TRUE, data = deermice)
sort(c(M.0 = AIC(M.0), M.b = AIC(M.b))) # Okay since omit.constant = TRUE
sort(c(M.0 = BIC(M.0), M.b = BIC(M.b))) # Okay since omit.constant = TRUE

---

**Posgeom**

**Positive-geometric Distribution**

**Description**

Density, distribution function, quantile function and random generation for the positive-geometric distribution.

**Usage**

\[
dposgeom(x, \text{prob}, \text{log} = \text{FALSE})
\]

\[
pposgeom(q, \text{prob})
\]

\[
qposgeom(p, \text{prob})
\]

\[
rposgeom(n, \text{prob})
\]

**Arguments**

- `x, q`: vector of quantiles.
- `p`: vector of probabilities.
- `n`: number of observations. Fed into `runif`.
- `prob`: vector of probabilities of success (of an ordinary geometric distribution). Short vectors are recycled.
- `log`: logical.

**Details**

The positive-geometric distribution is a geometric distribution but with the probability of a zero being zero. The other probabilities are scaled to add to unity. The mean therefore is \(1/\text{prob}\).

As `prob` decreases, the positive-geometric and geometric distributions become more similar. Like similar functions for the geometric distribution, a zero value of `prob` is not permitted here.

**Value**

dposgeom gives the density, pposgeom gives the distribution function, qposgeom gives the quantile function, and rposgeom generates random deviates.
Author(s)

T. W. Yee

See Also

zageometric, zigeometric, rgeom.

Examples

```r
prob <- 0.75; y <- rposgeom(n = 1000, prob)
table(y)
mean(y)  # Sample mean
1 / prob  # Population mean

(ii <- dposgeom(0:7, prob))
cumsum(ii) - pposgeom(0:7, prob)  # Should be 0s
table(rposgeom(100, prob))

table(dposgeom(runif(1000), prob))
round(dposgeom(1:10, prob) * 1000)  # Should be similar

## Not run:
x <- 0.5
barplot(rbind(dposgeom(x, prob), dgeom(x, prob)),
beside = TRUE, col = c("blue", "orange"),
main = paste("Positive geometric("", prob, ") (blue) vs",
" geometric("", prob, ") (orange)", sep = ""),
names.arg = as.character(x), las = 1, lwd = 2)
## End(Not run)
```

Description

Density, distribution function, quantile function and random generation for the positive-negative binomial distribution.

Usage

```r
dposnegbin(x, size, prob = NULL, munb = NULL, log = FALSE)
pposnegbin(q, size, prob = NULL, munb = NULL)
qposnegbin(p, size, prob = NULL, munb = NULL)
rposnegbin(n, size, prob = NULL, munb = NULL)
```
**Arguments**

- `x, q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. Fed into `runif`.
- `size, prob, mub, log`

  Same arguments as that of an ordinary negative binomial distribution (see `dnbinom`). Some arguments have been renamed slightly.

  Short vectors are recycled. The parameter `1/size` is known as a dispersion parameter; as `size` approaches infinity, the negative binomial distribution approaches a Poisson distribution.

**Details**

The positive-negative binomial distribution is a negative binomial distribution but with the probability of a zero being zero. The other probabilities are scaled to add to unity. The mean therefore is

\[ \mu/(1 - p(0)) \]

where \( \mu \) the mean of an ordinary negative binomial distribution.

**Value**

dposnegbin gives the density, pposnegbin gives the distribution function, qposnegbin gives the quantile function, and rposnegbin generates \( n \) random deviates.

**Author(s)**

T. W. Yee

**References**


**See Also**

`posnegbinomial`, `zanegbinomial`, `zinegbinomial`, `rnbinom`.

**Examples**

```r
mub <- 5; size <- 4; n <- 1000
table(y <- rposnegbin(n, mub = mub, size = size))
mean(y) # sample mean
mub / (1 - (size / (size + mub))^size) # population mean
mub / pnbinom(0, mu = mub, size = size, lower.tail = FALSE) # same as before

x <- (-1):17
(ii <- dposnegbin(x, mub = mub, size = size))
max(abs(cumsum(ii) - pposnegbin(x, mub = mub, size = size))) # Should be 0
```
'### Not run:
```r
x <- 0:10
barplot(rbind(dposnegbin(x, munb = munb, size = size), 
              dnbinom(x, mu = munb, size = size)), 
        beside = TRUE, col = c("blue","green"),
        main = paste("dposnegbin(munb = ", munb, ", size = ", size, ") (blue) vs",
                    " dnbinom(mu = ", munb, ", size = ", size, ") (green)"), sep = ""),
        names.arg = as.character(x))
```

## Another test for pposnegbin()
```r
nn <- 5000
mytab <- cumsum(table(rposnegbin(nn, munb = munb, size = size))) / nn
myans <- pposnegbin(sort(as.numeric(names(mytab))), munb = munb, size = size)
max(abs(mytab - myans)) # Should be 0
```

---

**posnegbinomial**  
*Positive Negative Binomial Distribution Family Function*

### Description
Maximum likelihood estimation of the two parameters of a positive negative binomial distribution.

### Usage
```r
posnegbinomial(lmunb = "loge", lsize = "loge",
                isize = NULL, zero = -2, nsimEIM = 250,
                ishrinkage = 0.95, imethod = 1)
```

### Arguments
- **lmunb**: Link function applied to the munb parameter, which is the mean $\mu_{nb}$ of an ordinary negative binomial distribution. See `Links` for more choices.
- **lsize**: Parameter link function applied to the dispersion parameter, called $k$. See `Links` for more choices.
- **isize**: Optional initial value for $k$, an index parameter. The value $1/k$ is known as a dispersion parameter. If failure to converge occurs try different values (and/or use `imethod`). If necessary this vector is recycled to length equal to the number of responses. A value `NULL` means an initial value for each response is computed internally using a range of values.
- **nsimEIM**, `zero`  
- **ishrinkage**, `imethod`  
  See `CommonVGAMffArguments`.

See `negbinomial`.
**Details**

The positive negative binomial distribution is an ordinary negative binomial distribution but with the probability of a zero response being zero. The other probabilities are scaled to sum to unity.

This family function is based on `negbinomial` and most details can be found there. To avoid confusion, the parameter `munb` here corresponds to the mean of an ordinary negative binomial distribution `negbinomial`. The mean of `posnegbinomial` is

\[ \mu_{nb}/(1 - p(0)) \]

where \( p(0) = (k/(k + \mu_{nb}))^k \) is the probability an ordinary negative binomial distribution has a zero value.

The parameters `munb` and `k` are not independent in the positive negative binomial distribution, whereas they are in the ordinary negative binomial distribution.

This function handles *multivariate* responses, so that a matrix can be used as the response. The number of columns is the number of species, say, and setting `zero = -2` means that *all* species have a `k` equalling a (different) intercept only.

**Value**

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

**Warning**

The Poisson model corresponds to `k` equalling infinity. If the data is Poisson or close to Poisson, numerical problems may occur. Possibly a loglog link could be added in the future to try help handle this problem.

This VGAM family function is computationally expensive and usually runs slowly; setting `trace = TRUE` is useful for monitoring convergence.

**Note**

This family function handles multiple responses.

**Author(s)**

Thomas W. Yee

**References**


**See Also**

`rposnegbin`, `pospoisson`, `negbinomial`, `zanegbinomial`, `rnbinom`, `CommonVGAMffArguments`, `corbet`, `logff`, `simulate.vlm`. 
Examples

```r
# Not run:
pdata <- data.frame(x2 = runif(nn <- 1000))
pdata <- transform(pdata, y1 = rposnegbin(nn, mumb = exp(0+2*x2), size = exp(1)),
y2 = rposnegbin(nn, mumb = exp(1+2*x2), size = exp(3)))
fit <- vglm(cbind(y1, y2) ~ x2, posnegbinomial, data = pdata, trace = TRUE)
coef(fit, matrix = TRUE)
dim(depvar(fit)) # dim(fit@y) is not as good

# Another artificial data example
pdata2 <- data.frame(mumb = exp(2), size = exp(3)); nn <- 1000
pdata2 <- transform(pdata2, y3 = rposnegbin(nn, mumb = mumb, size = size))
with(pdata2, table(y3))
fit <- vglm(y3 ~ 1, posnegbinomial, pdata2, trace = TRUE)
coef(fit, matrix = TRUE)
with(pdata2, mean(y3)) # Sample mean
head(with(pdata2, mumb/(1-((size/(size+mumb))^size))), 1) # Population mean
head(fitted(fit), 3)
head(predict(fit), 3)

# Example: Corbet (1943) butterfly Malaya data
fit <- vglm(ohfreq ~ 1, posnegbinomial, weights = species, data = corbet)
coef(fit, matrix = TRUE)
Coeff(f)
(khat <- Coef(fit)["size"])
pdf2 <- dposnegbin(x = with(corbet, ofreq), mu = fitted(fit), size = khat)
print(with(corbet, cbind(ofreq, species, fitted = pdf2*sum(species))), digits = 1)
with(corbet, matplot(ofreq, cbind(species, fitted = pdf2*sum(species)), las = 1,
xlab = "Observed frequency (of individual butterflies)",
type = "b", ylab = "Number of species", col = c("blue", "orange"),
main = "blue 1s = observe; orange 2s = fitted")
```

## End(Not run)

---

### The Positive-Normal Distribution

#### Description
Density, distribution function, quantile function and random generation for the univariate positive-normal distribution.

#### Usage

dposnorm(x, mean = 0, sd = 1, log = FALSE)
pposnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
quposnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rposnorm(n, mean = 0, sd = 1)
Arguments

- `x`, `q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If `length(n) > 1` then the length is taken to be the number required.
- `mean`, `sd`, `log`, `lower.tail`, `log.p`

Details

See `posnormal`, the VGAM family function for estimating the parameters, for the formula of the probability density function and other details.

Value

dposnorm gives the density, pposnorm gives the distribution function, qposnorm gives the quantile function, and rposnorm generates random deviates.

Author(s)

T. W. Yee

See Also

`posnormal`.

Examples

```r
## Not run
m <- 0.8; x <- seq(-1, 4, len = 50)
plot(x, dposnorm(x, m = m), type = "l", las = 1, ylim = 0:1,
     ylab = paste("Blue is density, orange is cumulative distribution function",
                 sub = "Purple lines are the 10,20,....,90 percentiles"),
     main = "Blue is density, orange is cumulative distribution function",
     abline(h = 0, col = "grey")
lines(x, pposnorm(x, m = m), col = "orange", type = "l")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qposnorm(probs, m = m)
lines(Q, dposnorm(Q, m = m), col = "purple", lty = 3, type = "h")
lines(Q, pposnorm(Q, m = m), col = "purple", lty = 3, type = "h")
abline(h = probs, col = "purple", lty = 3)
max(abs(pposnorm(Q, m = m) - probs))  # Should be 0
## End(Not run)
```
Description

Fits a positive (univariate) normal distribution.

Usage

posnormal(lmean = "identitylink", lsd = "loge",
        imean = NULL, isd = NULL, nsimEIM = 100, zero = NULL)

Arguments

lmean, lsd  
Link functions for the mean and standard deviation parameters of the usual univariate normal distribution. They are $\mu$ and $\sigma$ respectively. See Links for more choices.

imean, isd  
Optional initial values for $\mu$ and $\sigma$. A NULL means a value is computed internally.

nsimEIM  
See CommonVGAMffArguments for more information.

zero  
An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set {1,2} corresponding respectively to $\mu$, $\sigma$. If zero = NULL then all linear/additive predictors are modelled as a linear combination of the explanatory variables. For many data sets having zero = 2 is a good idea.

Details

The positive normal distribution is the ordinary normal distribution but with the probability of zero or less being zero. The rest of the probability density function is scaled up. Hence the probability density function can be written

$$f(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left( -\frac{1}{2}\frac{(y - \mu)^2}{\sigma^2} \right) \left[ 1 - \Phi\left( -\frac{\mu}{\sigma} \right) \right]$$

where $\Phi()$ is the cumulative distribution function of a standard normal (pnorm). Equivalently, this is

$$f(y) = \frac{1}{\sigma} \phi\left( \frac{y - \mu}{\sigma} \right) \left[ 1 - \Phi\left( -\frac{\mu}{\sigma} \right) \right]$$

where $\phi()$ is the probability density function of a standard normal distribution (dnorm).

The mean of $Y$ is

$$E(Y) = \mu + \sigma \frac{\phi\left( -\frac{\mu}{\sigma} \right)}{1 - \Phi\left( -\frac{\mu}{\sigma} \right)}.$$ 

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.
Warning
Under- or over-flow may occur if the data is ill-conditioned.

Note
The response variable for this family function is the same as uninormal except positive values are required. Reasonably good initial values are needed. Fisher scoring is implemented.
The distribution of the reciprocal of a positive normal random variable is known as an alpha distribution.

Author(s)
Thomas W. Yee

See Also
uninormal, tobit.

Examples
pdata <- data.frame(m = 1.0, SD = exp(1.0))
pdata <- transform(pdata, y = rposnorm(n <- 1000, m = m, sd = SD))

## Not run: with(pdata, hist(y, prob = TRUE, border = "blue",
   main = paste("posnorm(m =", m[1], ", sd =", round(SD[1], 2), ")")))
## End(Not run)
fit <- vglm(y ~ 1, posnormal, data = pdata, trace = TRUE)
coef(fit, matrix = TRUE)
(Cfit <- Coef(fit))
mygrid <- with(pdata, seq(min(y), max(y), len = 200))  # Add the fit to the histogram
## Not run: lines(mygrid, dposnorm(mygrid, Cfit[1], Cfit[2]), col = "red")

---

Description
Density, distribution function, quantile function and random generation for the positive-Poisson distribution.

Usage
dpospois(x, lambda, log = FALSE)
ppospois(q, lambda)
qpospois(p, lambda)
rpospois(n, lambda)
Arguments

\[ x, \mathbf{q} \] vector of quantiles.

\[ p \] vector of probabilities.

\[ n \] number of observations. Fed into \texttt{runif}.

\[ \lambda \] vector of positive means (of an ordinary Poisson distribution). Short vectors are recycled.

\[ \log \] logical.

Details

The positive-Poisson distribution is a Poisson distribution but with the probability of a zero being zero. The other probabilities are scaled to add to unity. The mean therefore is

\[
\lambda / (1 - \exp(-\lambda)).
\]

As \( \lambda \) increases, the positive-Poisson and Poisson distributions become more similar. Unlike similar functions for the Poisson distribution, a zero value of \( \lambda \) is not permitted here.

Value

dpospois gives the density, ppospois gives the distribution function, qpospois gives the quantile function, and rpospois generates random deviates.

Note

The family function \texttt{pospoisson} estimates \( \lambda \) by maximum likelihood estimation.

Author(s)

T. W. Yee

See Also

\texttt{pospoisson, zapoisson, zipoisson, rpois}.

Examples

```r
lambda <- 2; y = rpospois(n = 1000, lambda)
table(y)
mean(y)  # Sample mean
lambda / (1 - exp(-lambda))  # Population mean

(iii <- dpospois(0:7, lambda))
cumsum(iii) - qpospois(0:7, lambda)  # Should be 0s
table(rpospois(100, lambda))

table(qpospois(runif(1000), lambda))
round(dpospois(1:10, lambda) * 1000)  # Should be similar

## Not run: x <- 0:7
```
Description

Fits a positive Poisson distribution.

Usage

```r
pospoisson(link = "loge", expected = TRUE,
           ilambda = NULL, imethod = 1, zero = NULL)
```

Arguments

- `link`: Link function for the usual mean (lambda) parameter of an ordinary Poisson distribution. See Links for more choices.
- `expected`: Logical. Fisher scoring is used if `expected = TRUE`, else Newton-Raphson.
- `ilambda`, `imethod`, `zero`

See CommonVGAMffArguments for more information.

Details

The positive Poisson distribution is the ordinary Poisson distribution but with the probability of zero being zero. Thus the other probabilities are scaled up (i.e., divided by $1 - P[Y = 0]$). The mean, $\lambda/(1 - \exp(-\lambda))$, can be obtained by the extractor function `fitted` applied to the object.

A related distribution is the zero-inflated Poisson, in which the probability $P[Y = 0]$ involves another parameter $\phi$. See `zipoisson`.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as `vglm`, `rrvglm` and `vgam`.

Warning

Under- or over-flow may occur if the data is ill-conditioned.

Note

This family function can handle a multivariate response.

Yet to be done: a `quasi.pospoisson` which estimates a dispersion parameter.
powerlink

Author(s)

Thomas W. Yee

References


See Also

pospois, posnegbinomial, poissonff, zipoisson, simulate.vlm.

Examples

# Data from Coleman and James (1961)
cjdata <- data.frame(y = 1:6, freq = c(1486, 694, 195, 37, 10, 1))
fit <- vglm(y ~ 1, pospoisson, data = cjdata, weights = freq)
Coef(fit)
summary(fit)
fitted(fit)

pdata <- data.frame(x2 = runif(nn <- 1000)) # Artificial data
pdata <- transform(pdata, lambda = exp(1 - 2 * x2))
pdata <- transform(pdata, y1 = rpospois(nn, lambda))
with(pdata, table(y1))
fit <- vglm(y1 ~ x2, pospoisson, data = pdata, trace = TRUE, crit = "coef")
coef(fit, matrix = TRUE)

powerlink

Power Link Function

Description

Computes the power transformation, including its inverse and the first two derivatives.

Usage

powerlink(theta, power = 1, inverse = FALSE, deriv = 0,
          short = TRUE, tag = FALSE)

Arguments

theta Numeric or character. See below for further details.
power This denotes the power or exponent.
inverse, deriv, short, tag
          Details at Links.
Details

The power link function raises a parameter by a certain value of power. Care is needed because it is very easy to get numerical problems, e.g., if power=0.5 and theta is negative.

Value

For powerlink with deriv = 0, then theta raised to the power of power. And if inverse = TRUE then theta raised to the power of 1/power.

For deriv = 1, then the function returns d theta / d eta as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the reciprocal.

Note

Numerical problems may occur for certain combinations of theta and power. Consequently this link function should be used with caution.

Author(s)

Thomas W. Yee

See Also

Links, loge.

Examples

```r
powerlink("a", power = 2, short = FALSE, tag = TRUE)
powerlink(x <- 1:5)
powerlink(x, power = 2)
max(abs(powerlink(powerlink(x, power = 2),
                  power = 2, inverse = TRUE) - x)) # Should be 0
powerlink(x <- (-5):5, power = 0.5) # Has NAs

# 1/2 = 0.5
pdata <- data.frame(y = rbeta(n = 1000, shape1 = 2*2, shape2 = 3*2))
fit <- vglm(y ~ 1, betaR(ishape1 = powerlink(power = 0.5), i1 = 3,
                      ishape2 = powerlink(power = 0.5), i2 = 7), data = pdata)
t(coef(fit, matrix = TRUE))
Coef(fit) # Useful for intercept-only models
vcov(fit, untransform = TRUE)
```

prats  

Pregnant Rats Toxological Experiment Data

Description

A small toxological experiment data. The subjects are fetuses from two randomized groups of pregnant rats, and they were given a placebo or chemical treatment. The number with birth defects were recorded, as well as each litter size.
Usage

data(prats)

Format

A data frame with the following variables.

treatment A 0 means control; a 1 means the chemical treatment.
alive, litter.size The number of fetuses alive at 21 days, out of the number of fetuses alive at 4 days (the litter size).

Details

The data concerns a toxological experiment where the subjects are fetuses from two randomized groups of 16 pregnant rats each, and they were given a placebo or chemical treatment. The number with birth defects and the litter size were recorded. Half the rats were fed a control diet during pregnancy and lactation, and the diet of the other half was treated with a chemical. For each litter the number of pups alive at 4 days and the number of pups that survived the 21 day lactation period, were recorded.

Source

Weil, C. S. (1970) Selection of the valid number of sampling units and a consideration of their combination in toxicological studies involving reproduction, teratogenesis or carcinogenesis. *Food and Cosmetics Toxicology*, 8(2), 177–182.

References


See Also

betabinomial, betabinomialff.

Examples

prats
colSums(subset(prats, treatment == 0))
colSums(subset(prats, treatment == 1))
summary(prats)
predictqrrvglm

Predict Method for a CQO fit

Description

Predicted values based on a constrained quadratic ordination (CQO) object.

Usage

predictqrrvglm(object, newdata=NULL, 
    type = c("link", "response", "latvar", "terms"), 
    se.fit = FALSE, deriv = 0, dispersion = NULL, 
    extra = object@extra, varI.latvar = FALSE, refResponse = NULL, ...)

Arguments

object Object of class inheriting from "qrrvglm".
newdata An optional data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
type, se.fit, dispersion, extra
    See predictvglm.
deriv Derivative. Currently only 0 is handled.
varI.latvar, refResponse
    Arguments passed into Coef.qrrvglm.
...
    Currently undocumented.

Details

Obtains predictions from a fitted CQO object. Currently there are lots of limitations of this function; it is unfinished.

Value

See predictvglm.

Note

This function is not robust and has not been checked fully.

Author(s)

T. W. Yee
References


See Also
cqo.

Examples

```r
hsider[,1:6] <- scale(hsider[,1:6])  # Standardize the environmental vars
set.seed(1234)
# vvv pl <- cqo(cbind(Alopacce, Alopcur, Alopfa, Arctlute,
# vvv Arctperi, Auloalbi, Pardlugu, Pardmont,
# vvv Pardnigr, Pardpul, Trocterr, Zoraspin) ~
# vvv WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + RefLLux,
# vvv poissonff, data = hspider, Crow1positive = FALSE, I.tol = TRUE)
# vvv sort(deviance(pl, history = TRUE))  # A history of all the iterations
# vvv head(predict(pl))

# The following should be all zeros
# vvv max(abs(predict(pl, new = head(hspider)) - head(predict(pl))))
# vvv max(abs(predict(pl, new = head(hspider), type = "res")-head(fitted(pl))))
```

---

**predictvglm**  
*Predict Method for a VGLM fit*

**Description**

Predicted values based on a vector generalized linear model (VGLM) object.

**Usage**

```r
predictvglm(object, newdata = NULL,
            type = c("link", "response", "terms"),
            se.fit = FALSE, deriv = 0, dispersion = NULL,
            untransform = FALSE, extra = object@extra, ...)
```

**Arguments**

- `object`  
  Object of class inheriting from "vlm", e.g., `vglm`.

- `newdata`  
  An optional data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
type
The value of this argument can be abbreviated. The type of prediction required. The default is the first one, meaning on the scale of the linear predictors. This should be a $n \times M$ matrix.

The alternative “response” is on the scale of the response variable, and depending on the family function, this may or may not be the mean. Often this is the fitted value, e.g., fitted(vglmObject) (see fittedvlm). Note that the response is output from the @link inv slot, where the eta argument is the $n \times M$ matrix of linear predictors.

The "terms" option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale. The terms have been centered.

se.fit
logical: return standard errors?

deriv
Non-negative integer. Currently this must be zero. Later, this may be implemented for general values.

dispersion
Dispersion parameter. This may be inputted at this stage, but the default is to use the dispersion parameter of the fitted model.

extra
A list containing extra information. This argument should be ignored.

untransform
Logical. Reverses any parameter link function. This argument only works if type = "link", se.fit = FALSE, deriv = 0. Setting untransform = TRUE does not work for all VGAM family functions; only ones where there is a one-to-one correspondence between a simple link function and a simple parameter might work.

. . . Arguments passed into predictvlm.

Details
Obtains predictions and optionally estimates standard errors of those predictions from a fitted vglm object.

This code implements smart prediction (see smartpred).

Value
If se.fit = FALSE, a vector or matrix of predictions. If se.fit = TRUE, a list with components

fitted.values
Predictions

se.fit
Estimated standard errors

df
Degrees of freedom

sigma
The square root of the dispersion parameter

Warning
This function may change in the future.

Note
Setting se.fit = TRUE and type = "response" will generate an error.
**Author(s)**

Thomas W. Yee

**References**


**See Also**

`predict, vglm, predictvlm, smartpred`.

**Examples**

```r
# Illustrates smart prediction
pneumo <- transform(pneumo, let = log(exposure.time))
fit <- vglm(cbind(normal, mild, severe) ~ poly(c(scale(let)), 2),
            propodds, data = pneumo, trace = TRUE, x.arg = FALSE)
class(fit)

(q0 <- head(predict(fit)))
(q1 <- predict(fit, newdata = head(pneumo)))
(q2 <- predict(fit, newdata = head(pneumo)))
all.equal(q0, q1)  # Should be TRUE
all.equal(q1, q2)  # Should be TRUE

head(predict(fit))
head(predict(fit, untransform = TRUE))

p0 <- head(predict(fit, type = "response"))
p1 <- head(predict(fit, type = "response", newdata = pneumo))
p2 <- head(predict(fit, type = "response", newdata = pneumo))
p3 <- head(fitted(fit))
all.equal(p0, p1)  # Should be TRUE
all.equal(p1, p2)  # Should be TRUE
all.equal(p2, p3)  # Should be TRUE

predict(fit, type = "terms", se = TRUE)
```

---

**Prentice (1974) Log-gamma Distribution**

**Description**

Estimation of a 3-parameter log-gamma distribution described by Prentice (1974).

**Usage**

```r
prentice74(iloocation = "identitylink", lscale = "loge", lshape = "identitylink",
            ilocation = NULL, iscale = NULL, ishape = NULL, zero = 2:3)
```
Arguments

- ilocation, iscale, ilshape
  Parameter link function applied to the location parameter $a$, positive scale parameter $b$ and the shape parameter $q$, respectively. See Links for more choices.

- ilocation, iscale
  Initial value for $a$ and $b$, respectively. The defaults mean an initial value is determined internally for each.

- ishape
  Initial value for $q$. If failure to converge occurs, try some other value. The default means an initial value is determined internally.

- zero
  An integer-valued vector specifying which linear/additive predictors are modelled as intercepts-only. The values must be from the set {1,2,3}. See CommonVGAMffArguments for more information.

Details

The probability density function is given by

$$f(y; a, b, q) = |q| \exp(w / q^2 - e^w) / (b \Gamma(1/q^2)),$$

for shape parameter $q \neq 0$, positive scale parameter $b > 0$, location parameter $a$, and all real $y$. Here, $w = (y - a)q/b + \psi(1/q^2)$ where $\psi$ is the digamma function, digamma. The mean of $Y$ is $a$ (returned as the fitted values). This is a different parameterization compared to lgamma3.

Special cases: $q = 0$ is the normal distribution with standard deviation $b$, $q = -1$ is the extreme value distribution for maximums, $q = 1$ is the extreme value distribution for minima (Weibull). If $q > 0$ then the distribution is left skew, else $q < 0$ is right skew.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

The special case $q = 0$ is not handled, therefore estimates of $q$ too close to zero may cause numerical problems.

Note

The notation used here differs from Prentice (1974): $\alpha = a$, $\sigma = b$. Fisher scoring is used.

Author(s)

T. W. Yee

References

prinia

See Also

lgammaS, lgamma, gengamma.stacy.

Examples

pdata <- data.frame(x2 = runif(nn <- 1000))
pdata <- transform(pdata, loc = -1 + 2*x2, Scale = exp(1))
pdata <- transform(pdata, y = rlgamma(nn, loc = loc, scale = Scale, shape = 1))
fit <- vglm(y ~ x2, prentice74(zero = 2:3), data = pdata, trace = TRUE)
coef(fit, matrix = TRUE)  # Note the coefficients for location

prinia

Yellow-bellied Prinia

Description

A data frame with yellow-bellied Prinia.

Usage

data(prinia)

Format

A data frame with 151 observations on the following 23 variables.

length a numeric vector, the scaled wing length (zero mean and unit variance).

fat a numeric vector, fat index; originally 1 (no fat) to 4 (very fat) but converted to 0 (no fat) versus 1 otherwise.

cap a numeric vector, number of times the bird was captured or recaptured.

noncap a numeric vector, number of times the bird was not captured.

y01, y02, y03, y04, y05, y06 a numeric vector of 0s and 1s; for noncapture and capture resp.

y07, y08, y09, y10, y11, y12 same as above.

y13, y14, y15, y16, y17, y18, y19 same as above.

Details

The yellow-bellied Prinia Prinia flaviventris is a common bird species located in Southeast Asia. A capture–recapture experiment was conducted at the Mai Po Nature Reserve in Hong Kong during 1991, where captured individuals had their wing lengths measured and fat index recorded. A total of 19 weekly capture occasions were considered, where 151 distinct birds were captured.

More generally, the prinias are a genus of small insectivorous birds, and are sometimes referred to as wren-warblers. They are a little-known group of the tropical and subtropical Old World, the roughly 30 species being divided fairly equally between Africa and Asia.
Source

Thanks to Paul Yip for permission to make this data available.


Examples

```r
head(prinia)
summary(prinia)
rowSums(prinia[, c("cap", "noncap")]) # 19s

# Fit a positive-binomial distribution (M.h) to the data:
fit1 <- vglm(cbind(cap, noncap) ~ length + fat, posbinomial, data = prinia)

# Fit another positive-binomial distribution (M.h) to the data:
# The response input is suitable for posbernoulli.*-type functions.
fit2 <- vglm(cbind(y01, y02, y03, y04, y05, y06, y07, y08, y09, y10,
                   y11, y12, y13, y14, y15, y16, y17, y18, y19) ~
               length + fat, posbernoulli(b(drop.b = FALSE ~ 0), data = prinia)
```

---

probit

### Probit Link Function

**Description**

Computes the probit transformation, including its inverse and the first two derivatives.

**Usage**

```r
probit(theta, bvalue = NULL, inverse = FALSE, deriv = 0,
       short = TRUE, tag = FALSE)
```

**Arguments**

- `theta` Numeric or character. See below for further details.
- `bvalue` See [Links](#).
- `inverse`, `deriv`, `short`, `tag`

**Details**

The probit link function is commonly used for parameters that lie in the unit interval. Numerical values of `theta` close to 0 or 1 or out of range result in `Inf`, `~Inf`, `NA` or `NaN`.

--

[Links](#)
Value

For deriv = 0, the probit of theta, i.e., qnorm(theta) when inverse = FALSE, and if inverse = TRUE then pnorm(theta).

For deriv = 1, then the function returns \( \frac{d\ theta}{d\ eta} \) as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the reciprocal.

Note

Numerical instability may occur when theta is close to 1 or 0. One way of overcoming this is to use bvalue.

In terms of the threshold approach with cumulative probabilities for an ordinal response this link function corresponds to the univariate normal distribution (see uninormal).

Author(s)

Thomas W. Yee

References


See Also

Links, logit, cloglog, cauchit.

Examples

```r
p <- seq(0.01, 0.99, by = 0.01)
probit(p)
max(abs(probit(probit(p), inverse = TRUE) - p))  # Should be 0

p <- c(seq(-0.02, 0.02, by = 0.01), seq(0.97, 1.02, by = 0.01))
probit(p)  # Has NAs
probit(p, bvalue = .Machine$double.eps)  # Has no NAs

## Not run: p <- seq(0.01, 0.99, by = 0.01); par(lwd = (mylwd <- 2))
plot(p, logit(p), type = "l", col = "limegreen", ylab = "transformation",
     las = 1, main = "Some probability link functions")
lines(p, probit(p), col = "purple")
lines(p, cloglog(p), col = "chocolate")
lines(p, cauchit(p), col = "tan")
abline(v = 0.5, h = 0, lty = "dashed")
legend(0.1, 4.0, c("logit", "probit", "cloglog", "cauchit"),
       col = c("limegreen", "purple", "chocolate", "tan"), lwd = mylwd)
par(lwd = 1)
## End(Not run)
```
propodds

Proportional Odds Model for Ordinal Regression

Description

Fits the proportional odds model to a (preferably ordered) factor response.

Usage

propodds(reverse = TRUE, whitespace = FALSE)

Arguments

reverse, whitespace

Logical. Fed into arguments of the same name in cumulative.

Details

The proportional odds model is a special case from the class of cumulative link models. It involves a logit link applied to cumulative probabilities and a strong parallelism assumption. A parallelism assumption means there is less chance of numerical problems because the fitted probabilities will remain between 0 and 1; however, the parallelism assumption ought to be checked, e.g., via a likelihood ratio test. This VGAM family function is merely a shortcut for cumulative(reverse = reverse, link = "logit", parallel = TRUE). Please see cumulative for more details on this model.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

No check is made to verify that the response is ordinal if the response is a matrix; see ordered.

Author(s)

Thomas W. Yee

References


See Also
cumulative.

Examples

# Fit the proportional odds model, p.179, in McCullagh and Nelder (1989)
neumo <- transform(neumo, let = log(exposure.time))
(fit <- vglm(cbind(normal, mild, severe) ~ let, propodds, data = neumo))
depvar(fit) # Sample proportions
weights(fit, type = "prior") # Number of observations
coef(fit, matrix = TRUE)
constraints(fit) # Constraint matrices
summary(fit)

# Check that the model is linear in let ----------------------------
fit2 <- vglm(cbind(normal, mild, severe) ~ s(let, df = 2), propodds, data = neumo)
## Not run: plot(fit2, se = TRUE, lcol = 2, scol = 2)

# Check the proportional odds assumption with a LRT -----------
(fit3 <- vglm(cbind(normal, mild, severe) ~ let, 
           cumulative(parallel = FALSE, reverse = TRUE), data = neumo))
pchisq(deviance(fit3) - deviance(fit3),
       df = df.residual(fit3) - df.residual(fit3), lower.tail = FALSE)
lrtest(fit3, fit) # Easier

prplot

Probability Plots for Categorical Data Analysis

Description

Plots the fitted probabilities for some very simplified special cases of categorical data analysis models.

Usage

prplot(object, control = prplot.control(...), ...)

prplot.control(xlab = NULL, ylab = "Probability", main = NULL, xlim = NULL, 
                ylim = NULL, lty = par()$lty, col = par()$col, rcol = par()$col, 
                lwd = par()$lwd, rlwd = par()$lwd, las = par()$las, rug.arg = FALSE, ...)

Arguments

object Currently only an cumulative object. This includes a propodds object since that VGAM family function is a special case of cumulative.
control List containing some basic graphical parameters.
Details

For models involving one term in the RHS of the formula this function plots the fitted probabilities against the single explanatory variable.

Value

The object is returned invisibly with the preplot slot assigned. This is obtained by a call to plotvglm().

Note

This function is rudimentary.

See Also

cumulative.

Examples

pneumo <- transform(pneumo, let = log(exposure.time))
fit <- vglm(cbind(normal, mild, severe) ~ let, propodds, data = pneumo)
M <- npred(fit)  # Or fit@misc$M
## Not run: prplot(fit)
prplot(fit, lty = 1:M, col = (1:M)+2, rug = TRUE, las = 1,
      ylim = c(0, 1), rlwd = 2)
## End(Not run)
Arguments

smart a list containing parameters needed later for smart prediction.

Details

`put.smart` is used in "write" mode within a smart function. It saves parameters at the time of model fitting, which are later used for prediction. The function `put.smart` is the opposite of `get.smart`, and both deal with the same contents.

Value

Nothing is returned.

Side Effects

The variable `.smart.prediction.counter` in `smartpredenv` is incremented beforehand, and `.smart.prediction[[.smart]]` is assigned the list `smart`. If the list `.smart.prediction` in `smartpredenv` is not long enough to hold `smart`, then it is made larger, and the variable `.max.smart` in `smartpredenv` is adjusted accordingly.

See Also

`get.smart`.

Examples

```r
print(sm.min)
```

---

$qrrvglm.control$ Control function for $QRR$-VGLMs ($CQO$)

Description

Algorithmic constants and parameters for a constrained quadratic ordination ($CQO$), by fitting a quadratic reduced-rank vector generalized linear model (QRR-VGLM), are set using this function. It is the control function for `cqo`.

Usage

```r
qrrvglm.control(Rank = 1,
    Bestof = if (length(Cinit)) 1 else 10,
    checkwz = TRUE,
    Cinit = NULL,
    Crowapositive = TRUE,
    epsilon = 1.0e-06,
    EqualTolerances = NULL,
    eq.tolerances = TRUE,
    Etamat.colmax = 10,
```
Arguments

In the following, $R$ is the Rank, $M$ is the number of linear predictors, and $S$ is the number of responses (species). Thus $M = S$ for binomial and Poisson responses, and $M = 2S$ for the negative binomial and 2-parameter gamma distributions.

The numerical rank $R$ of the model, i.e., the number of ordination axes. Must be an element from the set $\{1,2,\ldots,\min(M,p_2)\}$ where the vector of explanatory variables $x$ is partitioned into $(x_1,x_2)$, which is of dimension $p_1 + p_2$. The variables making up $x_1$ are given by the terms in the noRRR argument, and the rest of the terms comprise $x_2$.

**Rank**

Integer. The best of Bestof models fitted is returned. This argument helps guard against local solutions by (hopefully) finding the global solution from many fits. The argument has value 1 if an initial value for $C$ is inputted using Cinit.

**checkwz**

logical indicating whether the diagonal elements of the working weight matrices should be checked whether they are sufficiently positive, i.e., greater than wzepsilon. If not, any values less than wzepsilon are replaced with this value.

**Cinit**

Optional initial $C$ matrix, which must be a $p_2$ by $R$ matrix. The default is to apply .Init.Poisson.QO() to obtain initial values.

**Crowlpositive**

Logical vector of length Rank (recycled if necessary): are the elements of the first row of $C$ positive? For example, if Rank is 4, then specifying Crowlpositive = c(FALSE, TRUE, TRUE, TRUE) will force $C[1,1]$ and $C[1,3]$ to be negative, and $C[1,2]$ and $C[1,4]$ to be positive. This argument allows for a reflection in the ordination axes because the coefficients of the latent variables are unique up to a sign.

**epsilon**

Positive numeric. Used to test for convergence for GLMs fitted in C. Larger values mean a loosening of the convergence criterion. If an error code of 3 is reported, try increasing this value.
eq.tolerances Logical indicating whether each (quadratic) predictor will have equal tolerances. Having eq.tolerances = TRUE can help avoid numerical problems, especially with binary data. Note that the estimated (common) tolerance matrix may or may not be positive-definite. If it is then it can be scaled to the $R$ by $R$ identity matrix, i.e., made equivalent to I.tolerances = TRUE. Setting I.tolerances = TRUE will force a common $R$ by $R$ identity matrix as the tolerance matrix to the data even if it is not appropriate. In general, setting I.tolerances = TRUE is preferred over eq.tolerances = TRUE because, if it works, it is much faster and uses less memory. However, I.tolerances = TRUE requires the environmental variables to be scaled appropriately. See Details for more details.

EqualTolerances Defunct argument. Use eq.tolerances instead.

Etamat.colmax Positive integer, no smaller than Rank. Controls the amount of memory used by .Init.Poisson.Q0(). It is the maximum number of columns allowed for the pseudo-response and its weights. In general, the larger the value, the better the initial value. Used only if Use.Init.Poisson.Q0 = TRUE.

FastAlgorithm Logical. Whether a new fast algorithm is to be used. The fast algorithm results in a large speed increases compared to Yee (2004). Some details of the fast algorithm are found in Appendix A of Yee (2006). Setting FastAlgorithm = FALSE will give an error.

GradientFunction Logical. Whether optim’s argument gr is used or not, i.e., to compute gradient values. Used only if FastAlgorithm is TRUE. The default value is usually faster on most problems.

Hstep Positive value. Used as the step size in the finite difference approximation to the derivatives by optim.

isd.latvar Initial standard deviations for the latent variables (site scores). Numeric, positive and of length $R$ (recycled if necessary). This argument is used only if I.tolerances = TRUE. Used by .Init.Poisson.Q0() to obtain initial values for the constrained coefficients $C$ adjusted to a reasonable value. It adjusts the spread of the site scores relative to a common species tolerance of 1 for each ordination axis. A value between 0.5 and 10 is recommended; a value such as 10 means that the range of the environmental space is very large relative to the niche width of the species. The successive values should decrease because the first ordination axis should have the most spread of site scores, followed by the second ordination axis, etc.

iKvector, ishape Numeric, recycled to length $S$ if necessary. Initial values used for estimating the positive $k$ and $\lambda$ parameters of the negative binomial and 2-parameter gamma distributions respectively. For further information see negbinomial and gamma2. These arguments override the iK and ishape arguments in negbinomial and gamma2.

I.tolerances Logical. If TRUE then the (common) tolerance matrix is the $R$ by $R$ identity matrix by definition. Note that having I.tolerances = TRUE implies eq.tolerances = TRUE, but not vice versa. Internally, the quadratic terms will be treated as offsets (in GLM jargon) and so the models can potentially be
fitted very efficiently. However, it is a very good idea to center and scale all numerical variables in the $x_2$ vector. See Details for more details. The success of iNtolerances = TRUE often depends on suitable values for isd.latvar and/or MUXfactor.

**ITol**erances
Defunct argument. Use I.tolera§es instead.

**maxitl**
Maximum number of times the optimizer is called or restarted. Most users should ignore this argument.

**imethod**
Method of initialization. A positive integer 1 or 2 or 3 etc. depending on the VGAM family function. Currently it is used for negbinomial and gamma2 only, and used within the C.

**Maxit.optim**
Positive integer. Number of iterations given to the function optim at each of the optim.maxit iterations.

**MUXfactor**
Multiplication factor for detecting large offset values. Numeric, positive and of length $R$ (recycled if necessary). This argument is used only if I.tolera§es = TRUE. Offsets are $-0.5$ multiplied by the sum of the squares of all $R$ latent variable values. If the latent variable values are too large then this will result in numerical problems. By too large, it is meant that the standard deviation of the latent variable values are greater than MUXfactor[r] * isd.latvar[r] for r=1:Rank (this is why centering and scaling all the numerical predictor variables in $x_2$ is recommended). A value about 3 or 4 is recommended. If failure to converge occurs, try a slightly lower value.

**optim.maxit**
Positive integer. Number of times optim is invoked. At iteration $i$, the $i$th value of Maxit.optim is fed into optim.

**noRRR**
Formula giving terms that are not to be included in the reduced-rank regression (or formation of the latent variables), i.e., those belong to $x_1$. Those variables which do not make up the latent variable (reduced-rank regression) correspond to the $B_1$ matrix. The default is to omit the intercept term from the latent variables.

**Norrr**
Defunct. Please use noRRR. Use of Norrr will become an error soon.

**parscale**
Numerical and positive-valued vector of length $C$ (recycled if necessary). Passed into optim(..., control = list(parscale = Parscale)); the elements of $C$ become $C / Parscale$. Setting I.tolera§es = TRUE results in line searches that are very large, therefore $C$ has to be scaled accordingly to avoid large step sizes. See Details for more information. It’s probably best to leave this argument alone.

**sd.Cinit**
Standard deviation of the initial values for the elements of $C$. These are normally distributed with mean zero. This argument is used only if Use.Init.Poisson.QO = FALSE and $C$ is not inputted using Cinit.

**trace**
Logical indicating if output should be produced for each iteration. The default is TRUE because the calculations are numerically intensive, meaning it may take a long time, so that the user might think the computer has locked up if trace = FALSE.

**SmallNo**
Positive numeric between .Machine$double.eps and 0.0001. Used to avoid under- or over-flow in the IRLS algorithm. Used only if FastAlgorithm is TRUE.
Use.Init.Poisson.QO

Logical. If TRUE then the function .Init.Poisson.QO() is used to obtain initial values for the canonical coefficients \( C \). If FALSE then random numbers are used instead.

\[ \text{wzepsilon} \]

Small positive number used to test whether the diagonals of the working weight matrices are sufficiently positive.

... Ignored at present.

Details

Recall that the central formula for CQO is

\[ \eta = B^T x_1 + A \nu + \sum_{m=1}^{M} (\nu^T D_m \nu) e_m \]

where \( x_1 \) is a vector (usually just a 1 for an intercept), \( x_2 \) is a vector of environmental variables, \( \nu = C^T x_2 \) is a \( R \)-vector of latent variables, \( e_m \) is a vector of 0s but with a 1 in the \( m \)th position.

QRR-VGLMs are an extension of RR-VGLMs and allow for maximum likelihood solutions to constrained quadratic ordination (CQO) models.

Having \( \text{iNtolerances} = \text{TRUE} \) means all the tolerance matrices are the order-\( R \) identity matrix, i.e., it forces bell-shaped curves/surfaces on all species. This results in a more difficult optimization problem (especially for 2-parameter models such as the negative binomial and gamma) because of overflow errors and it appears there are more local solutions. To help avoid the overflow errors, scaling \( C \) by the factor \( \text{parscale} \) can help enormously. Even better, scaling \( C \) by specifying \( \text{isd.latvar} \) is more understandable to humans. If failure to converge occurs, try adjusting \( \text{parscale} \), or better, setting \( \text{eq.tolerances} = \text{TRUE} \) (and hope that the estimated tolerance matrix is positive-definite). To fit an equal-tolerances model, it is firstly best to try setting \( \text{i.tolerances} = \text{TRUE} \) and varying \( \text{isd.latvar} \) and/or \( \text{MUXfactor} \) if it fails to converge. If it still fails to converge after many attempts, try setting \( \text{eq.tolerances} = \text{TRUE} \), however this will usually be a lot slower because it requires a lot more memory.

With a \( R > 1 \) model, the latent variables are always uncorrelated, i.e., the variance-covariance matrix of the site scores is a diagonal matrix.

If setting \( \text{eq.tolerances} = \text{TRUE} \) is used and the common estimated tolerance matrix is positive-definite then that model is effectively the same as the \( \text{i.tolerances} = \text{TRUE} \) model (the two are transformations of each other). In general, \( \text{i.tolerances} = \text{TRUE} \) is numerically more unstable and presents a more difficult problem to optimize; the arguments \( \text{isd.latvar} \) and/or \( \text{MUXfactor} \) often must be assigned some good value(s) (possibly found by trial and error) in order for convergence to occur. Setting \( \text{i.tolerances} = \text{TRUE} \) forces a bell-shaped curve or surface onto all the species data, therefore this option should be used with deliberation. If unsuitable, the resulting fit may be very misleading. Usually it is a good idea for the user to set \( \text{eq.tolerances} = \text{FALSE} \) to see which species appear to have a bell-shaped curve or surface. Improvements to the fit can often be achieved using transformations, e.g., nitrogen concentration to log nitrogen concentration.

Fitting a CAO model (see \texttt{cao}) first is a good idea for pre-examining the data and checking whether it is appropriate to fit a CQO model.

Value

A list with components matching the input names.
**qrvglm.control**

**Warning**

The default value of `bestof` is a bare minimum for many datasets, therefore it will be necessary to increase its value to increase the chances of obtaining the global solution.

**Note**

When `I.tolerances = TRUE` it is a good idea to apply `scale` to all the numerical variables that make up the latent variable, i.e., those of \( x_2 \). This is to make them have mean 0, and hence avoid large offset values which cause numerical problems.

This function has many arguments that are common with `rrvglm.control` and `vglm.control`. It is usually a good idea to try fitting a model with `I.tolerances = TRUE` first, and if convergence is unsuccessful, then try `eq.tolerances = TRUE` and `I.tolerances = FALSE`. Ordination diagrams with `eq.tolerances = TRUE` have a natural interpretation, but with `eq.tolerances = FALSE` they are more complicated and requires, e.g., contours to be overlaid on the ordination diagram (see `lvplot.qrrvglm`).

In the example below, an equal-tolerances CQO model is fitted to the hunting spiders data. Because `I.tolerances = TRUE`, it is a good idea to center all the \( x_2 \) variables first. Upon fitting the model, the actual standard deviation of the site scores are computed. Ideally, the `isd.latvar` argument should have had this value for the best chances of getting good initial values. For comparison, the model is refitted with that value and it should run more faster and reliably.

**Author(s)**

Thomas W. Yee

**References**


**See Also**

cqo, rcqo, Coef.qrrvglm, Coef.qrrvglm-class, optim, binomialff, poissonff, negbinomial, gamma2, gaussianff.

**Examples**

```r
## Not run: # Poisson CQO with equal tolerances
set.seed(111)  # This leads to the global solution
hspider[,1:6] <- scale(hspider[,1:6])  # Good idea when I.tolerances = TRUE
p1 <- cqo(cbind(Alopacc, Alopcur, Alopabr, Arctlute, Arctperi, Auloalbi,
                Pardugu, Pardmoun, Pardnigr, Pardpull, Trocterr, Zoraspin) ~
              WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
              quasipoissonff, data = hspider, eq.tolerances = TRUE)
sort(deviance(p1, history = TRUE))  # A history of all the iterations
(isd.latvar <- apply(latvar(p1), 2, sd))  # Should be approx isd.latvar
```
# Refit the model with better initial values
set.seed(111)  # This leads to the global solution
pl <- cpo(cbind(Alopacce, Alopocene, Alopenf, Arctlute, Arctperi, Auloalbi,
                 Pardlugu, Pardmont, Pardnigr, Pardpull, Trocterr, Zoraspin) ~
               WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + ReflLux,
               I.tolerances = TRUE, quasipoissonff, data = hspider,
               isd.latvar = isd.latvar)  # Note the use of isd.latvar here
sort(deviance(pl, history = TRUE))  # A history of all the iterations

## End(Not run)

### qtplot.gumbel

**Quantile Plot for Gumbel Regression**

**Description**

Plots quantiles associated with a Gumbel model.

**Usage**

```r
qtplot.gumbel(object, show.plot = TRUE,
               y.arg = TRUE, spline.fit = FALSE, label = TRUE,
               R = object@misc$R, percentiles = object@misc$percentiles,
               add.arg = FALSE, mpv = object@misc$mpv,
               xlab = NULL, ylab = "", main = "",
               pch = par()$pch, pcol.arg = par()$col,
               llty.arg = par()$lty, lcol.arg = par()$col, llwd.arg = par()$lwd,
               tcol.arg = par()$col, tadj = 1, ...)
```

**Arguments**

- **object** A VGAM extremes model of the Gumbel type, produced by modelling functions such as `vglm` and `vgam` with a family function either "gumbel" or "egumbel".
- **show.plot** Logical. Plot it? If `FALSE` no plot will be done.
- **y.arg** Logical. Add the raw data on to the plot?
- **spline.fit** Logical. Use a spline fit through the fitted percentiles? This can be useful if there are large gaps between some values along the covariate.
- **label** Logical. Label the percentiles?
- **R** See `gumbel`.
- **percentiles** See `gumbel`.
- **add.arg** Logical. Add the plot to an existing plot?
- **mpv** See `gumbel`.
- **xlab** Caption for the x-axis. See `par`.
- **ylab** Caption for the y-axis. See `par`.

---

 qtplot.gumbel  

**Quantile Plot for Gumbel Regression**

**Description**

Plots quantiles associated with a Gumbel model.

**Usage**

```r
qtplot.gumbel(object, show.plot = TRUE,
               y.arg = TRUE, spline.fit = FALSE, label = TRUE,
               R = object@misc$R, percentiles = object@misc$percentiles,
               add.arg = FALSE, mpv = object@misc$mpv,
               xlab = NULL, ylab = "", main = "",
               pch = par()$pch, pcol.arg = par()$col,
               llty.arg = par()$lty, lcol.arg = par()$col, llwd.arg = par()$lwd,
               tcol.arg = par()$col, tadj = 1, ...)
```

**Arguments**

- **object** A VGAM extremes model of the Gumbel type, produced by modelling functions such as `vglm` and `vgam` with a family function either "gumbel" or "egumbel".
- **show.plot** Logical. Plot it? If `FALSE` no plot will be done.
- **y.arg** Logical. Add the raw data on to the plot?
- **spline.fit** Logical. Use a spline fit through the fitted percentiles? This can be useful if there are large gaps between some values along the covariate.
- **label** Logical. Label the percentiles?
- **R** See `gumbel`.
- **percentiles** See `gumbel`.
- **add.arg** Logical. Add the plot to an existing plot?
- **mpv** See `gumbel`.
- **xlab** Caption for the x-axis. See `par`.
- **ylab** Caption for the y-axis. See `par`.  

---

 qtplot.gumbel  

**Quantile Plot for Gumbel Regression**

**Description**

Plots quantiles associated with a Gumbel model.

**Usage**

```r
qtplot.gumbel(object, show.plot = TRUE,
               y.arg = TRUE, spline.fit = FALSE, label = TRUE,
               R = object@misc$R, percentiles = object@misc$percentiles,
               add.arg = FALSE, mpv = object@misc$mpv,
               xlab = NULL, ylab = "", main = "",
               pch = par()$pch, pcol.arg = par()$col,
               llty.arg = par()$lty, lcol.arg = par()$col, llwd.arg = par()$lwd,
               tcol.arg = par()$col, tadj = 1, ...)
```

**Arguments**

- **object** A VGAM extremes model of the Gumbel type, produced by modelling functions such as `vglm` and `vgam` with a family function either "gumbel" or "egumbel".
- **show.plot** Logical. Plot it? If `FALSE` no plot will be done.
- **y.arg** Logical. Add the raw data on to the plot?
- **spline.fit** Logical. Use a spline fit through the fitted percentiles? This can be useful if there are large gaps between some values along the covariate.
- **label** Logical. Label the percentiles?
- **R** See `gumbel`.
- **percentiles** See `gumbel`.
- **add.arg** Logical. Add the plot to an existing plot?
- **mpv** See `gumbel`.
- **xlab** Caption for the x-axis. See `par`.
- **ylab** Caption for the y-axis. See `par`.
main

Title of the plot. See title.

pch

Plotting character. See par.

pcol.arg

Color of the points. See the col argument of par.

lty.arg

Line type. Line type. See the lty argument of par.

lcol.arg

Color of the lines. See the col argument of par.

llwd.arg

Line width. See the lwd argument of par.

tcol.arg

Color of the text (if label is TRUE). See the col argument of par.

tadj

Text justification. See the adj argument of par.

Arguments passed into the plot function when setting up the entire plot. Useful arguments here include sub and las.

Details

There should be a single covariate such as time. The quantiles specified by percentiles are plotted.

Value

The object with a list called qtplot in the post slot of object. (If show.plot = FALSE then just the list is returned.) The list contains components

fitted.values

The percentiles of the response, possibly including the MPV.

percentiles

The percentiles (small vector of values between 0 and 100).

Note

Unlike gumbel, one cannot have percentiles = NULL.

Author(s)

Thomas W. Yee

See Also
gumbel.

Examples

ymat <- as.matrix(venice[, paste("r", 1:10, sep = ""))
fit1 <- vgam(ymat ~ s(year, df = 3), gumbel(R = 365, mpv = TRUE),
  data = venice, trace = TRUE, na.action = na.pass)
head(fitted(fit1))

## Not run: par(mfrow = c(1, 1), bty = "l", xpd = TRUE, las = 1)
qtplot(fit1, mpv = TRUE, lcol = c(1, 2, 5), tcol = c(1, 2, 5),
  lwd = 2, pcol = "blue", tadj = 0.4, ylab = "Sea level (cm)"
)

qtplot(fit1, perc = 97, mpv = FALSE, lcol = 3, tcol = 3,
  lwd = 2, tadj = 0.4, add = TRUE) -> saved
qtplot.lmscreg

Quantile Plot for LMS Quantile Regression

Description

Plots quantiles associated with a LMS quantile regression.

Usage

qtplot.lmscreg(object, newdata = NULL,
                percentiles = object@misc$percentiles,
                show.plot = TRUE, ...)

Arguments

object
  A VGAM quantile regression model, i.e., an object produced by modelling functions such as vglm and vgam with a family function beginning with "lms."., e.g., lms.yjn.
newdata
  Optional data frame for computing the quantiles. If missing, the original data is used.
percentiles
  Numerical vector with values between 0 and 100 that specify the percentiles (quantiles). The default are the percentiles used when the model was fitted.
show.plot
  Logical. Plot it? If FALSE no plot will be done.
...
  Graphical parameter that are passed into plotqtplot.lmscreg.

Details

The ‘primary’ variable is defined as the main covariate upon which the regression or smoothing is performed. For example, in medical studies, it is often the age. In VGAM, it is possible to handle more than one covariate, however, the primary variable must be the first term after the intercept.

Value

A list with the following components.

fitted.values
  A vector of fitted percentile values.
percentiles
  The percentiles used.

Note

plotqtplot.lmscreg does the actual plotting.
Author(s)

Thomas W. Yee

References


See Also

`plotqtplot.lmscreg`, `deplot.lmscreg`, `lms.bcn`, `lms.bcg`, `lms.yjn`.

Examples

```r
## Not run:
fit <- vgam(BMI ~ s(age, df = c(4, 2)), lms.bcn(zero=1), data = bmi.nz)
qtplot(fit)
qtplot(fit, perc = c(25, 50, 75, 95), lcol = "blue", tcol = "blue", llwd = 2)
## End(Not run)
```

quasibinomialff

Quasi-Binomial Family Function

Description

Family function for fitting generalized linear models to binomial responses, where the dispersion parameters are unknown.

Usage

```r
quasibinomialff(link = "logit", multiple.responses = FALSE,
onedpar = !multiple.responses, parallel = FALSE, zero = NULL)
```

Arguments

- **link** Link function. See `Links` for more choices.
- **multiple.responses** Multiple responses? If TRUE, then the response is interpreted as \( M \) binary responses, where \( M \) is the number of columns of the response matrix. In this case, the response matrix should have zero/one values only.
  - If FALSE and the response is a (2-column) matrix, then the number of successes is given in the first column and the second column is the number of failures.
- **onedpar** One dispersion parameter? If `multiple.responses`, then a separate dispersion parameter will be computed for each response (column), by default. Setting `onedpar=TRUE` will pool them so that there is only one dispersion parameter to be estimated.
parallel
A logical or formula. Used only if `multiple.responses` is TRUE. This argument allows for the parallelism assumption whereby the regression coefficients for a variable is constrained to be equal over the \( M \) linear/additive predictors.

zero
An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \{1,2,\ldots, M\}, where \( M \) is the number of columns of the matrix response.

Details
The final model is not fully estimated by maximum likelihood since the dispersion parameter is unknown (see pp.124–8 of McCullagh and Nelder (1989) for more details).

A dispersion parameter that is less/greater than unity corresponds to under-/over-dispersion relative to the binomial model. Over-dispersion is more common in practice.

Setting `multiple.responses=TRUE` is necessary when fitting a Quadratic RR-VGLM (see `cqo`) because the response will be a matrix of \( M \) columns (e.g., one column per species). Then there will be \( M \) dispersion parameters (one per column of the response).

Value
An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `vgam`, `rrvglm`, `cqo`, and `cao`.

Warning
The log-likelihood pertaining to the ordinary family is used to test for convergence during estimation, and is printed out in the summary.

Note
If `multiple.responses` is FALSE (the default), then the response can be of one of three formats: a factor (first level taken as success), a vector of proportions of success, or a 2-column matrix (first column = successes) of counts. The argument `weights` in the modelling function can also be specified. In particular, for a general vector of proportions, you will need to specify `weights` because the number of trials is needed.

If `multiple.responses` is TRUE, then the matrix response can only be of one format: a matrix of 1s and 0s (1=success).

This function is only a front-end to the `VGAM` family function `binomialff()`: indeed, `quasibinomialff(...)` is equivalent to `binomialff(..., dispersion=0)`. Here, the argument `dispersion=0` signifies that the dispersion parameter is to be estimated.

Regardless of whether the dispersion parameter is to be estimated or not, its value can be seen from the output from the `summary()` of the object.

Author(s)
Thomas W. Yee
quasipoissonff

References


See Also

binomialff, rrvglm, cqu, cao, logit, probit, cloglog, cauchit, poissonff, quasipoissonff, quasibinomial.

Examples

quasibinomialff()
quasibinomialff(link = "probit")

# Nonparametric logistic regression
hunua <- transform(hunua, a.5 = sqrt(altitude))  # Transformation of altitude
fit1 <- vglm(aagaus ~ poly(a.5, 2), quasibinomialff, hunua)
fit2 <- vgam(aagaus ~ s(a.5, df = 2), quasibinomialff, hunua)
# Not run:
plot(fit2, se = TRUE, llwd = 2, lcol = "orange", scol = "orange",
     xlab = "sqrt(altitude)", ylim = c(-3, 1),
     main = "GAM and quadratic GLM fitted to species data")
plotvgam(fit1, se = TRUE, lcol = "blue", scol = "blue", add = TRUE, llwd = 2)

# Here, the dispersion parameter defaults to 1
fit0 <- vglm(aagaus ~ poly(a.5, 2), binomialff, hunua)
fit0@misc$dispersion # dispersion parameter

quasipoissonff

*Quasi-Poisson Family Function*

Description

Fits a generalized linear model to a Poisson response, where the dispersion parameter is unknown.

Usage

quasipoissonff(link = "loge", onedpar = FALSE,
               parallel = FALSE, zero = NULL)
Arguments

- **link**: Link function. See `Links` for more choices.
- **onedpar**: One dispersion parameter? If the response is a matrix, then a separate dispersion parameter will be computed for each response (column), by default. Setting `onedpar=TRUE` will pool them so that there is only one dispersion parameter to be estimated.
- **parallel**: A logical or formula. Used only if the response is a matrix.
- **zero**: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \{1,2,...,M\}, where \(M\) is the number of columns of the matrix response.

Details

\(M\) defined above is the number of linear/additive predictors.

If the dispersion parameter is unknown, then the resulting estimate is not fully a maximum likelihood estimate.

A dispersion parameter that is less/greater than unity corresponds to under-/over-dispersion relative to the Poisson model. Over-dispersion is more common in practice.

When fitting a Quadratic RR-VGLM, the response is a matrix of \(M\), say, columns (e.g., one column per species). Then there will be \(M\) dispersion parameters (one per column of the response matrix).

Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, `vgam`, `rrvglm`, `cqo`, and `cao`.

Warning

See the warning in `quasibinomialff`.

Note

This function will handle a matrix response automatically.

The call `poissonff(dispersion = 0, ...)` is equivalent to `quasipoissonff(...).` The latter was written so that R users of `quasipoisson()` would only need to add a "ff" to the end of the family function name.

Regardless of whether the dispersion parameter is to be estimated or not, its value can be seen from the output from the `summary()` of the object.

Author(s)

Thomas W. Yee

References

**Qvar**

*Quasi-variances Preprocessing Function*

**Description**

Takes a *vglm* fit or a variance-covariance matrix, and preprocesses it for *rcim* and *uninormal* so that quasi-variances can be computed.

**Usage**

```r
Qvar(object, factorname = NULL, which.linpred = 1, 
     coef.indices = NULL, labels = NULL, 
     dispersion = NULL, reference.name = "(reference)", estimates = NULL)
```

**Arguments**

- **object**
  A "*vglm*" object or a variance-covariance matrix, e.g., `vcov(vglm.object)`. The former is preferred since it contains all the information needed. If a matrix then `factorname` and/or `coef.indices` should be specified to identify the factor.

- **which.linpred**
  A single integer from the set 1:M. Specifies which linear predictor to use. Let the value of `which.linpred` be called `j`. Then the factor should appear in that linear predictor, hence the `j`th row of the constraint matrix corresponding to the factor should have at least one nonzero value. Currently the `j`th row must have exactly one nonzero value because programming it for more than one nonzero value is difficult.

- **factorname**
  Character. If the *vglm* object contains more than one factor as explanatory variable then this argument should be the name of the factor of interest. If `object` is a variance-covariance matrix then this argument should also be specified.

- **labels**
  Character. Optional, for labelling the variance-covariance matrix.

**Examples**

```r
quasipoissonff()

## Not run: n <- 200; p <- 5; S <- 5
mydata <- rcqo(n, p, S, fam = "poisson", eq.tol = FALSE)
myform <- attr(mydata, "formula")
p1 <- cpo(myform, fam = quasipoissonff, eq.tol = FALSE, data = mydata)
sort(deviance(p1, history = TRUE))  # A history of all the iterations
lplot(p1, y = TRUE, lcol = 1:S, pch = 1:S, pcol = 1:S)
summary(p1)  # The dispersion parameters are estimated

## End(Not run)
```
dispersion Numeric. Optional, passed into vcov() with the same argument name.
reference.name Character. Label for the reference level.
coef.indices Optional numeric vector of length at least 3 specifying the indices of the factor from the variance-covariance matrix.
estimates an optional vector of estimated coefficients (redundant if object is a model).

Details
Suppose a factor with \( L \) levels is an explanatory variable in a regression model. By default, R treats the first level as baseline so that its coefficient is set to zero. It estimates the other \( L - 1 \) coefficients, and with its associated standard errors, this is the conventional output. From the complete variance-covariance matrix one can compute \( L \) quasi-variances based on all pairwise difference of the coefficients. They are based on an approximation, and can be treated as uncorrelated. In minimizing the relative (not absolute) errors it is not hard to see that the estimation involves a RCIM (rcim) with an exponential link function (explink).

If object is a model, then at least one of factorname or coef.indices must be non-NULL. The value of coef.indices, if non-NULL, determines which rows and columns of the model’s variance-covariance matrix to use. If coef.indices contains a zero, an extra row and column are included at the indicated position, to represent the zero variances and covariances associated with a reference level. If coef.indices is NULL, then factorname should be the name of a factor effect in the model, and is used in order to extract the necessary variance-covariance estimates.

Quasi-variances were first implemented in R with qvcalc. This implementation draws heavily from that.

Value
A \( L \) by \( L \) matrix whose \( i-j \) element is the logarithm of the variance of the \( i \)th coefficient minus the \( j \)th coefficient, for all values of \( i \) and \( j \). The diagonal elements are arbitrary and are set to zero.

The matrix has an attribute that corresponds to the prior weight matrix; it is accessed by uninormal and replaces the usual weights argument of vglm. This weight matrix has ones on the off-diagonals and some small positive number on the diagonals.

Warning
Negative quasi-variances may occur (one of them and only one), though they are rare in practice. If so then numerical problems may occur. See qvcalc() for more information.

Note
This is an adaptation of qvcalc() in qvcalc. It should work for all vglm models with one linear predictor, i.e., \( M = 1 \). For \( M > 1 \) the factor should appear only in one of the linear predictors.

It is important to set maxit to be larger than usual for rcim since convergence is slow. Upon successful convergence the \( i \)th row effect and the \( i \)th column effect should be equal. A simple computation involving the fitted and predicted values allows the quasi-variances to be extracted (see example below).

A function to plot comparison intervals has not been written here.
Author(s)

T. W. Yee, based heavily on \texttt{qvcalc()} in \texttt{qvcalc} written by David Firth.

References


See Also

\texttt{rcim, vglm, qvar, uninormal, explink, qvcalc()} in \texttt{qvcalc}, \texttt{ships}.

Examples

```r
# Example 1
data("ships", package = "MASS")

Shipmodel <- vglm(incidents ~ type + year + period,
    quasipoissonff, offset = log(service),
    # trace = TRUE, model = TRUE,
    data = ships, subset = (service > 0))

# Easiest form of input
fit1 <- rcim(Qvar(Shipmodel, "type"), uninormal("explink"), maxit = 99)
qvar(fit1)           # Easy method to get the quasi-variances
qvar(fit1, se = TRUE) # Easy method to get the quasi-standard errors

(quasiVar <- exp(diag(fitted(fit1))) / 2)  # Version 1
(quasiVar <- diag(predict(fit1)[, c(TRUE, FALSE)]) / 2)  # Version 2
(quasiSE <- sqrt(quasiVar))

# Another form of input
fit2 <- rcim(Qvar(Shipmodel, coef.ind = c(0, 2:5), reference.name = "typeA"),
    uninormal("explink"), maxit = 99)
## Not run: qvplot(fit2, col = "green", lwd = 3, scl = "blue", slwd = 2, las = 1)

# The variance-covariance matrix is another form of input (not recommended)
fit3 <- rcim(Qvar(cbind(0, rbind(0, vcov(Shipmodel)[2:5, 2:5])),
    labels = c("typeA", "typeB", "typeC", "typeD", "typeE"),
    estimates = c(typeA = 0, coef(Shipmodel)[2:5])),
    uninormal("explink"), maxit = 99)

(QualiVar <- exp(diag(fitted(fit3))) / 2)  # Version 1
(QualiVar <- diag(predict(fit3)[, c(TRUE, FALSE)]) / 2)  # Version 2
(QualiSE <- sqrt(QuasiVar))
## Not run: qvplot(fit3)

# Example 2: a model with \textit{M} \textgreater{} 1 linear predictors
```
## Not run

```r
# Not run: require("VGAMdata")
x$s.nz.f <- subset(x$s.nz, sex == "F")
x$s.nz.f <- subset(x$s.nz.f, !is.na(babies) & !is.na(age) & !is.na(ethnicity))
x$s.nz.f <- subset(x$s.nz.f, ethnicity != "Other")

clist <- list("sm.bs(age, df = 4)" = rbind(1, 0),
   "sm.bs(age, df = 3)" = rbind(0, 1),
   "ethnicity" = diag(2),
   "(Intercept)" = diag(2))

fit1 <- vglm(babies ~ sm.bs(age, df = 4) + sm.bs(age, df = 3) + ethnicity,
   zipoissonff(zero = NULL), x$s.nz.f,
   constraints = clist, trace = TRUE)

Fit1 <- rcim(Qvar(fit1, "ethnicity", which.linpred = 1),
   uninormal("explink", imethod = 1), maxit = 99, trace = TRUE)

Fit2 <- rcim(Qvar(fit1, "ethnicity", which.linpred = 2),
   uninormal("explink", imethod = 1), maxit = 99, trace = TRUE)
```

## End(Not run)

## Not run: par(mfrow = c(1, 2))

qvplot(Fit1, scl = "blue", pch = 16, main = expression(eta[1]),
   slwd = 1.5, las = 1, length.arrows = 0.07)

qvplot(Fit2, scl = "blue", pch = 16, main = expression(eta[2]),
   slwd = 1.5, las = 1, length.arrows = 0.07)

## End(Not run)

---

### qvar

**Quasi-variances Extraction Function**

#### Description

Takes a `rcim` fit of the appropriate format and returns either the quasi-variances or quasi-standard errors.

#### Usage

```
qvar(object, se = FALSE, ...)
```

#### Arguments

- **object**: A `rcim` object that has family function `uninormal` with the `explink` link. See below for an example.
- **se**: Logical. If FALSE then the quasi-variances are returned, else the square root of them, called quasi-standard errors.
- **...**: Currently unused.
Details

This simple function is ad hoc and simply is equivalent to computing the quasi-variances by \( \text{diag}(\text{predict}(\text{fit1})[,\ c(\text{TRUE}, \text{FALSE})]) \). This function is for convenience only. Serious users of quasi-variances ought to understand why and how this function works.

Value

A vector of quasi-variances or quasi-standard errors.

Author(s)

T. W. Yee.

See Also

\texttt{rcim, uninormal, explink, Qvar, ships}.

Examples

data("ships", package = "MASS")
Shipmodel <- \texttt{vglm(incidents ~ type + year + period, quasipoissonff, offset = log(service), data = ships, subset = (service > 0))}

# Easiest form of input
fit1 <- \texttt{rcimQvar(Shipmodel, "type"), uninormal("explink"), maxit = 99)
qvar(fit1) # Quasi-variances
qvar(fit1, se = \text{TRUE}) # Quasi-standard errors

# Manually compute them:
(quasiVar <- \text{exp(diag(fitted(fit1))) / 2}) \quad \# Version 1
(quasiVar <- \text{diag(predict(fit1)[, \ c(\text{TRUE}, \text{FALSE})]) / 2}) \quad \# Version 2
(quasiSE <- sqrt(quasiVar))

## Not run: \texttt{qvplot(fit1, col = "green", lwd = 3, scol = "blue", slwd = 2, las = 1)}

Rayleigh

The Rayleigh Distribution

Description

Density, distribution function, quantile function and random generation for the Rayleigh distribution with parameter \( a \).

Usage

drayleigh(x, scale = 1, log = \text{FALSE})
prayleigh(q, scale = 1, lower.tail = \text{TRUE}, log.p = \text{FALSE})
qrayleigh(p, scale = 1, lower.tail = \text{TRUE}, log.p = \text{FALSE})
rrayleigh(n, scale = 1)
Arguments

- `x`, `q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. Fed into `runif`.
- `scale` the scale parameter $b$.
- `log` Logical. If `log = TRUE` then the logarithm of the density is returned.
- `lower.tail, log.p` Same meaning as in `pnorm` or `qnorm`.

Details

See `rayleigh`, the VGAM family function for estimating the scale parameter $b$ by maximum likelihood estimation, for the formula of the probability density function and range restrictions on the parameter $b$.

Value

drayleigh gives the density, prayleigh gives the distribution function, qrayleigh gives the quantile function, and rrayleigh generates random deviates.

Note

The Rayleigh distribution is related to the Maxwell distribution.

Author(s)

T. W. Yee and Kai Huang

References


See Also

`rayleigh`, `maxwell`.

Examples

```r
# Not run:  Scale <- 2; x <- seq(-1, 8, by = 0.1)
plot(x, drayleigh(x, scale = Scale), type = "l", ylim = c(0,1),
     las = 1, ylab = "",
     main = "Rayleigh density divided into 10 equal areas; orange = cdf")
abline(h = 0, col = "blue", lty = 2)
qq <- qrayleigh(seq(0.1, 0.9, by = 0.1), scale = Scale)
lines(qq, drayleigh(qq, scale = Scale), col = "purple", lty = 3, type = "h")
lines(x, prayleigh(x, scale = Scale), col = "orange")
# End(Not run)
```
Description

Estimating the parameter of the Rayleigh distribution by maximum likelihood estimation. Right-censoring is allowed.

Usage

```r
rayleigh(lscale = "log", nrfs = 1/3 + 0.01,
        oim.mean = TRUE, zero = NULL)
cens.rayleigh(lscale = "log", oim = TRUE)
```

Arguments

- `lscale`: Parameter link function applied to the scale parameter \( b \). See [Links](#) for more choices. A log link is the default because \( b \) is positive.
- `nrfs`: Numeric, of length one, with value in \([0, 1]\). Weighting factor between Newton-Raphson and Fisher scoring. The value 0 means pure Newton-Raphson, while 1 means pure Fisher scoring. The default value uses a mixture of the two algorithms, and retaining positive-definite working weights.
- `oim.mean`: Logical, used only for intercept-only models. TRUE means the mean of the OIM elements are used as working weights. If TRUE then this argument has top priority for working out the working weights. FALSE means use another algorithm.
- `oim`: Logical. For censored data only, TRUE means the Newton-Raphson algorithm, and FALSE means Fisher scoring.
- `zero`: Details at [CommonVGAMffArguments](#).

Details

The Rayleigh distribution, which is used in physics, has a probability density function that can be written

\[
f(y) = y \exp(-0.5(y/b)^2)/b^2\]

for \( y > 0 \) and \( b > 0 \). The mean of \( Y \) is \( b\sqrt{\pi/2} \) (returned as the fitted values) and its variance is \( b^2(4 - \pi)/2 \).

The [VGAM](#) family function `cens.rayleigh` handles right-censored data (the true value is greater than the observed value). To indicate which type of censoring, input `extra = list(rightcensored = vec2)` where `vec2` is a logical vector the same length as the response. If the component of this list is missing then the logical values are taken to be FALSE. The fitted object has this component stored in the `extra` slot.

The [VGAM](#) family function `rayleigh` handles multiple responses.
Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

The theory behind the argument oim is not fully complete.

Note

The poisson.points family function is more general so that if statistic = 1 and dimension = 2 then it coincides with rayleigh. Other related distributions are the Maxwell and Weibull distributions.

Author(s)

T. W. Yee

References


See Also

Rayleigh, genrayleigh, riceff, maxwell, weibullR, poisson.points, simulate.vlm.

Examples

```r
nn <- 1000; Scale <- exp(2)
rdata <- data.frame(ystar = rrayleigh(nn, scale = Scale))
fit <- vglm(ystar ~ 1, rayleigh, data = rdata, trace = TRUE, crit = "coef")
head(fitted(fit))
with(rdata, mean(ystar))
coef(fit, matrix = TRUE)
Coef(fit)

# Censored data
rdata <- transform(rdata, U = runif(nn, 5, 15))
rdata <- transform(rdata, y = pmin(U, ystar))
## Not run: par(mfrow = c(1,2))
hist(with(rdata, ystar)); hist(with(rdata, y))
## End(Not run)
extra <- with(rdata, list(rightcensored = ystar > U))
fit <- vglm(y ~ 1, cens.rayleigh, data = rdata, trace = TRUE, extra = extra)
table(fit@extra$rightcen)
coef(fit, matrix = TRUE)
head(fitted(fit))
```
Rcim

Mark the Baseline of Row and Column on a Matrix data

Description

Rearrange the rows and columns of the input so that the first row and first column are baseline. This function is for rank-zero row-column interaction models (RCIMs; i.e., general main effects models).

Usage

Rcim(mat, rbaseline = 1, cbaseline = 1)

Arguments

- **mat**: Matrix, of dimension \(r \times c\). It is best that it is labelled with row and column names.
- **rbaseline**, **cbaseline**: Numeric (row number of the matrix mat) or character (matching a row name of mat) that the user wants as the row baseline or reference level. Similarly cbaseline for the column.

Details

This is a data preprocessing function for rcim. For rank-zero row-column interaction models this function establishes the baseline (or reference) levels of the matrix response with respect to the row and columns—these become the new first row and column.

Value

Matrix of the same dimension as the input, with rbaseline and cbaseline specifying the first rows and columns. The default is no change in mat.

Note

This function is similar to moffset; see moffset for information about the differences. If numeric, the arguments rbaseline and cbaseline differ from arguments roffset and coffset in moffset by 1 (when elements of the matrix agree).

Author(s)

Alfian F. Hadi and T. W. Yee.

See Also

moffset, rcim, plotrcim.
Examples

(alcoff.e <- moffset(alcoff, roffset = "6", postfix = "x"))
(aa <- Rcim(alcoff, rbaseline = "11", cbaseline = "Sun"))
(bb <- moffset(alcoff, "11", "Sun", postfix = "x"))
aa - bb # Note the difference!

rcqo

Constrained Quadratic Ordination

Description

Random generation for constrained quadratic ordination (CQO).

Usage

rcqo(n, p, S, Rank = 1,
family = c("poisson", "negbinomial", "binomial-poisson",
"Binomial-negbinomial", "ordinal-poisson",
"Ordinal-negbinomial", "gamma2"),
eq.maximums = FALSE, eq.tolerances = TRUE, es.optimums = FALSE,
lo.abundance = if (eq.maximums) hi.abundance else 10,
hi.abundance = 100, sd.latvar = head(1.5/2^(0:3), Rank),
sd.optimums = ifelse(es.optimums, 1.5/Rank, 1) *
    ifelse(scale.latvar, sd.latvar, 1),
sd.tolerances = 0.25, Kvec = 1, Shape = 1,
sqrt.arg = FALSE, log.arg = FALSE, rhox = 0.5, breaks = 4,
seed = NULL, optimums1.arg = NULL, Crowspositive = TRUE,
xmat = NULL, scale.latvar = TRUE)

Arguments

n Number of sites. It is denoted by \( n \) below.
p Number of environmental variables, including an intercept term. It is denoted by \( p \) below. Must be no less than \( 1 + R \) in value.
S Number of species. It is denoted by \( S \) below.
Rank The rank or the number of latent variables or true dimension of the data on the reduced space. This must be either 1, 2, 3 or 4. It is denoted by \( R \).
family What type of species data is to be returned. The first choice is the default. If binomial then a 0 means absence and 1 means presence. If ordinal then the breaks argument is passed into the breaks argument of \texttt{cut}. Note that either the Poisson or negative binomial distributions are used to generate binomial and ordinal data, and that an upper-case choice is used for the negative binomial distribution (this makes it easier for the user). If "gamma2" then this is the 2-parameter gamma distribution.
eq.maximums Logical. Does each species have the same maximum? See arguments \( \text{lo}\_\text{abundance} \) and \( \text{hi}\_\text{abundance} \).
eq.tolerances Logical. Does each species have the same tolerance? If TRUE then the common value is 1 along every latent variable, i.e., all species' tolerance matrices are the order-$R$ identity matrix.

es.optimums Logical. Do the species have equally spaced optimums? If TRUE then the quantity $S^{1/R}$ must be an integer with value 2 or more. That is, there has to be an appropriate number of species in total. This is so that a grid of optimum values is possible in $R$-dimensional latent variable space in order to place the species' optimums. Also see the argument sd.tolerances.

lo.abundance, hi.abundance Numeric. These are recycled to a vector of length $S$. The species have a maximum between lo.abundance and hi.abundance. That is, at their optimal environment, the mean abundance of each species is between the two componentwise values. If eq.maximums is TRUE then lo.abundance and hi.abundance must have the same values. If eq.maximums is FALSE then the logarithm of the maximums are uniformly distributed between $\log(lo.abundance)$ and $\log(hi.abundance)$.

sd.latvar Numeric, of length $R$ (recycled if necessary). Site scores along each latent variable have these standard deviation values. This must be a decreasing sequence of values because the first ordination axis contains the greatest spread of the species’ site scores, followed by the second axis, followed by the third axis, etc.

sd.optimums Numeric, of length $R$ (recycled if necessary). If es.optimums = FALSE then, for the $r$th latent variable axis, the optimums of the species are generated from a normal distribution centered about 0. If es.optimums = TRUE then the $S$ optimums are equally spaced about 0 along every latent variable axis. Regardless of the value of es.optimums, the optimums are then scaled to give standard deviation sd.optimums[$r$].

sd.tolerances Logical. If eq.tolerances = FALSE then, for the $r$th latent variable, the species' tolerances are chosen from a normal distribution with mean 1 and standard deviation sd.tolerances[$r$]. However, the first species y1 has its tolerance matrix set equal to the order-$R$ identity matrix. All tolerance matrices for all species are diagonal in this function. This argument is ignored if eq.tolerances is TRUE, otherwise it is recycled to length $R$ if necessary.

Kvector A vector of positive $k$ values (recycled to length $S$ if necessary) for the negative binomial distribution (see negbinomial for details). Note that a natural default value does not exist, however the default value here is probably a realistic one, and that for large values of $\mu$ one has $Var(Y) = \mu^2/k$ approximately.

Shape A vector of positive $\lambda$ values (recycled to length $S$ if necessary) for the 2-parameter gamma distribution (see gamma2 for details). Note that a natural default value does not exist, however the default value here is probably a realistic one, and that $Var(Y) = \mu^2/\lambda$.

sqrt.arg Logical. Take the square-root of the negative binomial counts? Assigning sqrt.arg = TRUE when family="negbinomial" means that the resulting species data can be considered very crudely to be approximately Poisson distributed. They will not integers in general but much easier (less numerical problems) to estimate using something like cpo(. . . , family="poissonff").

log.arg Logical. Take the logarithm of the gamma random variates? Assigning log.arg = TRUE when family="gamma2" means that the resulting species data can be considered very crudely to be approximately Gaussian distributed about its (quadratic)
The result is that it is much easier (less numerical problems) to estimate using something like \texttt{cqo(..., family="gaussianff")}.

**rhox**

Numeric, less than 1 in absolute value. The correlation between the environmental variables. The correlation matrix is a matrix of 1's along the diagonal and \( \text{rhox} \) in the off-diagonals. Note that each environmental variable is normally distributed with mean 0. The standard deviation of each environmental variable is chosen so that the site scores have the determined standard deviation, as given by argument \texttt{sd.latvar}.

**breaks**

If \texttt{family} is assigned an ordinal value then this argument is used to define the cutpoints. It is fed into the \texttt{breaks} argument of \texttt{cut}.

**seed**

If given, it is passed into \texttt{set.seed}. This argument can be used to obtain reproducible results. If set, the value is saved as the "seed" attribute of the returned value. The default will not change the random generator state, and return \texttt{Random.seed} as "seed" attribute.

**optimums1.arg**

If assigned and \texttt{Rank = 1} then these are the explicity optimums. Recycled to length \( S \).

**Crowl.positive**

See \texttt{qrrvglm.control} for details.

**xmat**

The \( n \times (p - 1) \) environmental matrix can be inputted.

**scale.latvar**

Logical. If \texttt{FALSE} the argument \texttt{sd.latvar} is ignored and no scaling of the latent variable values is performed.

### Details

This function generates data coming from a constrained quadratic ordination (CQO) model. In particular, data coming from a \textit{species packing model} can be generated with this function. The species packing model states that species have equal tolerances, equal maximums, and optimums which are uniformly distributed over the latent variable space. This can be achieved by assigning the arguments \texttt{es.optimums = TRUE}, \texttt{eq.maximums = TRUE}, \texttt{eq.tolerances = TRUE}.

At present, the Poisson and negative binomial abundances are generated first using \texttt{lo.abundance} and \texttt{hi.abundance}, and if \texttt{family} is binomial or ordinal then it is converted into these forms.

In CQO theory the \( n \times p \) matrix \( X \) is partitioned into two parts \( X_1 \) and \( X_2 \). The matrix \( X_2 \) contains the 'real' environmental variables whereas the variables in \( X_1 \) are just for adjustment purposes; they contain the intercept terms and other variables that one wants to adjust for when (primarily) looking at the variables in \( X_2 \). This function has \( X_1 \) only being a matrix of ones, i.e., containing an intercept only.

### Value

A \( n \times (p - 1 + S) \) data frame with components and attributes. In the following the attributes are labelled with double quotes.

- \( x2, x3, x4, \ldots, xp \)

  The environmental variables. This makes up the \( n \times (p - 1) \times X_2 \) matrix. Note that \( x1 \) is not present; it is effectively a vector of ones since it corresponds to an intercept term when \texttt{cqo} is applied to the data.
y1, y2, x3, ..., yS
    The species data. This makes up the n by S matrix Y. This will be of the form
described by the argument family.

"concoefficients"
    The p - 1 by R matrix of constrained coefficients (or canonical coefficients).
    These are also known as weights or loadings.

"formula"
    The formula involving the species and environmental variable names. This can
be used directly in the formula argument of cqq.

"log.maximums"
    The S-vector of species’ maximums, on a log scale. These are uniformly dis-
tributed between log(lo.abundance) and log(hi.abundance).

"latvar"
    The n by R matrix of site scores. Each successive column (latent variable) has
sample standard deviation equal to successive values of sd.latvar.

"eta"
    The linear/additive predictor value.

"optimums"
    The S by R matrix of species’ optimums.

"tolerances"
    The S by R matrix of species’ tolerances. These are the square root of the
diagonal elements of the tolerance matrices (recall that all tolerance matrices
are restricted to being diagonal in this function).

Other attributes are "break", "family", "Rank", "lo.abundance", "hi.abundance", "eq.tolerances",
"eq.maximums", "seed" as used.

Note

This function is under development and is not finished yet. There may be a few bugs.

Yet to do: add an argument that allows absences to be equal to the first level if ordinal data is
requested.

Author(s)

T. W. Yee

References

logical Monographs, 74, 685–701.


Research, 18, 271–317.

See Also
cqq, qrrvglm.control, cut, binomialff, poissonff, negbinomial, gamma2, gaussianff.
Examples

## Not run:

### Example 1: Species packing model:

```r
n <- 100; p <- 5; S <- 5
mydata <- rcqo(n, p, S, es.opt = TRUE, eq.max = TRUE)
names(mydata)
(myform <- attr(mydata, "formula"))
fit <- cco(myform, poissonff, mydata, Bestof = 3)  # eq.tol = TRUE
matplot(attr(mydata, "latvar"), mydata[, -(1:(p-1))], col = 1:S)
persp(fit, col = 1:S, add = TRUE)
Lvplot(fit, lcol = 1:S, y = TRUE, pcol = 1:S)  # The same plot as above

# Compare the fitted model with the 'truth'
concoef(fit)  # The fitted model
attr(mydata, "concoefficients")  # The 'truth'

```

```r
c(apply(attr(mydata, "latvar"), 2, sd),
  apply(latvar(fit), 2, sd))  # Both values should be approx equal
```

### Example 2: Negative binomial data fitted using a Poisson model:

```r
n <- 200; p <- 5; S <- 5
mydata <- rcqo(n, p, S, fam = "negbin", sqrt = TRUE)
myform <- attr(mydata, "formula")
fit <- cco(myform, fam = poissonff, dat = mydata)  # I.tol = TRUE,
Lvplot(fit, lcol = 1:S, y = TRUE, pcol = 1:S)
# Compare the fitted model with the 'truth'
concoef(fit)  # The fitted model
attr(mydata, "concoefficients")  # The 'truth'
```

### Example 3: Gamma2 data fitted using a Gaussian model:

```r
n <- 200; p <- 5; S <- 3
mydata <- rcqo(n, p, S, fam = "gamma2", log.arg = TRUE)
fit <- cco(attr(mydata, "formula"),
    fam = gaussianff, data = mydata)  # I.tol = TRUE,
matplot(attr(mydata, "latvar"),
    exp(mydata[, -(1:(p-1))]), col = 1:S)  # 'raw' data
# Fitted model to transformed data:
Lvplot(fit, lcol = 1:S, y = TRUE, pcol = 1:S)
# Compare the fitted model with the 'truth'
concoef(fit)  # The fitted model
attr(mydata, "concoefficients")  # The 'truth'
```

## End(Not run)
Description

Generates Dirichlet random variates.

Usage

rdiric(n, shape, dimension = NULL, is.matrix.shape = FALSE)

Arguments

n  number of observations. Note it has two meanings, see is.matrix.shape below.
shape  the shape parameters. These must be positive. If dimension is specified, values are recycled if necessary to length dimension.
dimension  the dimension of the distribution. If dimension is not numeric then it is taken to be length(shape) (or ncol(shape) if is.matrix.shape == TRUE).
is.matrix.shape  Logical. If TRUE then shape must be a matrix, and then n is no longer the number of rows of the answer but the answer has n * nrow(shape) rows. If FALSE (the default) then shape is a vector and each of the n rows of the answer have shape as its shape parameters.

Details

This function is based on a relationship between the gamma and Dirichlet distribution. Random gamma variates are generated, and then Dirichlet random variates are formed from these.

Value

A n by dimension matrix of Dirichlet random variates. Each element is positive, and each row will sum to unity. If shape has names then these will become the column names of the answer.

Author(s)

Thomas W. Yee

References


See Also

dirichlet is a VGAM family function for fitting a Dirichlet distribution to data.

Examples

ddata <- data.frame(rdiric(n = 1000, shape = c(y1 = 3, y2 = 1, y3 = 4)))
fit <- vglm(cbind(y1, y2, y3) ~ 1, dirichlet, data = ddata, trace = TRUE)
Coeff(fit)
coef(fit, matrix = TRUE)
rec.exp1  

Upper Record Values from a 1-parameter Exponential Distribution

Description

Maximum likelihood estimation of the rate parameter of a 1-parameter exponential distribution when the observations are upper record values.

Usage

rec.exp1(lrate = "loge", irate = NULL, imethod = 1)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lrate</td>
<td>Link function applied to the rate parameter. See Links for more choices.</td>
</tr>
<tr>
<td>irate</td>
<td>Numeric. Optional initial values for the rate. The default value NULL means they are computed internally, with the help of imethod.</td>
</tr>
<tr>
<td>imethod</td>
<td>Integer, either 1 or 2 or 3. Initial method, three algorithms are implemented. Choose the another value if convergence fails, or use irate.</td>
</tr>
</tbody>
</table>

Details

The response must be a vector or one-column matrix with strictly increasing values.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

By default, this family function has the intercept-only MLE as the initial value, therefore convergence may only take one iteration. Fisher scoring is used.

Author(s)

T. W. Yee

References


See Also

exponential.
Examples

```r
rawy <- rexp(n <- 10000, rate = exp(1))
y <- unique(cummax(rawy)) # Keep only the records

length(y) / y[length(y)] # MLE of rate

fit <- vglm(y ~ 1, rec.exp1, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
```

Description

Maximum likelihood estimation of the two parameters of a univariate normal distribution when the observations are upper record values.

Usage

```r
rec.normal(lmean, lsd, imean = NULL, isd = NULL, imethod = 1, zero = NULL)
```

Arguments

- `lmean, lsd` Link functions applied to the mean and sd parameters. See `Links` for more choices.
- `imean, isd` Numeric. Optional initial values for the mean and sd. The default value NULL means they are computed internally, with the help of `imethod`.
- `imethod` Integer, either 1 or 2 or 3. Initial method, three algorithms are implemented. Choose the another value if convergence fails, or use `imean` and/or `isd`.
- `zero` An integer vector, containing the value 1 or 2. If so, the mean or standard deviation respectively are modelled as an intercept only. Usually, setting `zero = 2` will be used, if used at all. The default value NULL means both linear/additive predictors are modelled as functions of the explanatory variables.

Details

The response must be a vector or one-column matrix with strictly increasing values.

Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.
Note

This family function tries to solve a difficult problem, and the larger the data set the better. Convergence failure can commonly occur, and convergence may be very slow, so set maxit = 200, trace = TRUE, say. Inputting good initial values are advised.

This family function uses the BFGS quasi-Newton update formula for the working weight matrices. Consequently the estimated variance-covariance matrix may be inaccurate or simply wrong! The standard errors must be therefore treated with caution; these are computed in functions such as vcov() and summary().

Author(s)

T. W. Yee

References


See Also

uninormal, double.cens.normal.

Examples

```r
nn <- 10000; mymean <- 100
# First value is reference value or trivial record
Rdata <- data.frame(rawy = c(mymean, rnorm(nn, me = mymean, sd = exp(3))))
# Keep only observations that are records:
rdata <- data.frame(y = unique(cummax(with(Rdata, rawy))))

fit <- vglm(y ~ 1, rec.normal, data = rdata, trace = TRUE, maxit = 200)
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)
```

reciprocal

Reciprocal link function

Description

Computes the reciprocal transformation, including its inverse and the first two derivatives.

Usage

```r
reciprocal(theta, bvalue = NULL, inverse = FALSE, deriv = 0,
  short = TRUE, tag = FALSE)
negreciprocal(theta, bvalue = NULL, inverse = FALSE, deriv = 0,
  short = TRUE, tag = FALSE)
```
Arguments

theta numeric or character. See below for further details.

bvalue See Links.

inverse, deriv, short, tag
Details at Links.

Details

The reciprocal link function is a special case of the power link function. Numerical values of
theta close to 0 result in Inf, -Inf, NA or NaN.

The negreciprocal link function computes the negative reciprocal, i.e., \(-1/\theta\).

Value

For reciprocal: for deriv = 0, the reciprocal of theta, i.e., 1/theta when inverse = FALSE.
and if inverse = TRUE then 1/theta. For deriv = 1, then the function returns \( d \theta / d \eta \)
as a function of theta if inverse = FALSE, else if inverse = TRUE then it returns the reciprocal.

Note

Numerical instability may occur when theta is close to 0.

Author(s)

Thomas W. Yee

References

Hall.

See Also

identity, powerlink.

Examples

reciprocal(1:5)
reciprocal(1:5, inverse = TRUE, deriv = 2)
negreciprocal(1:5)
negreciprocal(1:5, inverse = TRUE, deriv = 2)

x <- (-3):3
reciprocal(x) # Has Inf
reciprocal(x, bvalue = .Machine$double.eps) # Has no Inf
rhobit  

**Rhobit Link Function**

**Description**

Computes the rhobit link transformation, including its inverse and the first two derivatives.

**Usage**

```r
rhobit(theta, bminvalue = NULL, bmaxvalue = NULL,
       inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)
```

**Arguments**

- `theta` Numeric or character. See below for further details.
- `bminvalue`, `bmaxvalue` Optional boundary values, e.g., values of `theta` which are less than or equal to -1 can be replaced by `bminvalue` before computing the link function value. And values of `theta` which are greater than or equal to 1 can be replaced by `bmaxvalue` before computing the link function value. See [Links](#).
- `inverse`, `deriv`, `short`, `tag` Details at [Links](#).

**Details**

The rhobit link function is commonly used for parameters that lie between $-1$ and $1$. Numerical values of `theta` close to $-1$ or $1$ or out of range result in `Inf`, `-Inf`, `NA` or `NaN`.

**Value**

- For `deriv = 0`, the rhobit of `theta`, i.e., $\log((1 + \theta)/(1 - \theta))$ when `inverse = FALSE`, and if `inverse = TRUE` then $(\exp(\theta) - 1)/(\exp(\theta) + 1)$.
- For `deriv = 1`, then the function returns $d\theta/d\eta$ as a function of `theta` if `inverse = FALSE`, else if `inverse = TRUE` then it returns the reciprocal.

**Note**

Numerical instability may occur when `theta` is close to $-1$ or $1$. One way of overcoming this is to use `bminvalue`, etc.

The correlation parameter of a standard bivariate normal distribution lies between $-1$ and $1$, therefore this function can be used for modelling this parameter as a function of explanatory variables.

The link function `rhobit` is very similar to `fisherz`, e.g., just twice the value of `fisherz`.

**Author(s)**

Thomas W. Yee


See Also

Links, binom2.rho, fisherz.

Examples

```r
theta <- seq(-0.99, 0.99, by = 0.01)
y <- rhobit(theta)
## Not run:
plot(theta, y, type = "l", las = 1, ylab = "", main = "rhobit(theta)"
abline(v = 0, h = 0, lty = 2)
## End(Not run)

x <- c(seq(-1.02, -0.98, by = 0.01), seq(0.97, 1.02, by = 0.01))
rhobit(x) # Has NAs
rhobit(x, bminvalue = -1 + .Machine$double.eps,
       bmaxvalue = 1 - .Machine$double.eps) # Has no NAs
```

---

Rice

The Rice Distribution

Description

Density, distribution function, quantile function and random generation for the Rician distribution.

Usage

```r
drice(x, sigma, vee, log = FALSE)
price(q, sigma, vee, lower.tail = TRUE, log.p = FALSE, ...)
qrice(p, sigma, vee, lower.tail = TRUE, log.p = FALSE, ...)
rrice(n, sigma, vee)
```

Arguments

- `x, q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. Same as in `runif`.
- `vee, sigma` See `riceff`.
- `...` Other arguments such as `lower.tail`.
- `lower.tail, log.p` Same meaning as in `pnorm` or `qnorm`.
- `log` Logical. If `log = TRUE` then the logarithm of the density is returned.

Details

See `riceff`, the VGAM family function for estimating the two parameters, for the formula of the probability density function and other details.

Formulas for `price()` and `qrice()` are based on the Marcum-Q function.
Value

drice gives the density, price gives the distribution function, qrice gives the quantile function, and rrice generates random deviates.

Author(s)

T. W. Yee and Kai Huang

See Also

riceff.

Examples

```r
## Not run:  x <- seq(0.01, 7, len = 201)
plot(x, drice(x, vee = 0, sigma = 1), type = "n", las = 1, ylab = "",
     main = "Density of Rice distribution for various values of v")
sigma <- 1; vee <- c(0, 0.5, 1, 2, 4)
for (ii in 1:length(vee))
  lines(x, drice(x, vee = vee[ii], sigma), col = ii)
legend(x = 5, y = 0.6, legend = as.character(vee),
       col = 1:length(vee), lty = 1)

x <- seq(0, 4, by = 0.01); vee <- 1; sigma <- 1
probs <- seq(0.05, 0.95, by = 0.05)
plot(x, drice(x, vee = vee, sigma = sigma), type = "l", col = "blue",
     main = "Blue is density, orange is cumulative distribution function",
     ylim = c(0, 1), sub = "Purple are 5, 10, ..., 95 percentiles",
     las = 1, ylab = "", cex.main = 0.9)
abline(h = 0:1, col = "black", lty = 2)
Q <- qrice(probs, sigma, vee = vee)
lines(Q, drice(qrice(probs, sigma, vee = vee),
              sigma, vee = vee), col = "purple", lty = 3, type = "h")
lines(x, price(x, sigma, vee = vee), type = "l", col = "orange")
lines(Q, price(Q, sigma, vee = vee), col = "purple", lty = 3, type = "h")
lines(Q, price(Q, sigma, vee = vee), col = "purple", lty = 3, type = "h")
abline(h = probs, col = "purple", lty = 3)
max(abs(price(Q, sigma, vee = vee) - probs)) # Should be 0

## End(Not run)
```

riceff

**Rice Distribution Family Function**

**Description**

Estimates the two parameters of a Rice distribution by maximum likelihood estimation.
Usage

riceff(lsigma = "loge", lvee = "loge", isigma = NULL,
    ivee = NULL, nsimEIM = 100, zero = NULL, nowarning = FALSE)

Arguments

nowarning Logical. Suppress a warning? Ignored for VGAM 0.9-7 and higher.

lvee, lsigma Link functions for the $v$ and $\sigma$ parameters. See Links for more choices and for general information.

ivee, isigma Optional initial values for the parameters. See CommonVGAMffArguments for more information. If convergence failure occurs (this VGAM family function seems to require good initial values) try using these arguments.

nsimEIM, zero See CommonVGAMffArguments for more information.

details

The Rician distribution has density function

$$f(y; v, \sigma) = \frac{y}{\sigma^2} \exp\left(-\frac{y^2 + v^2}{2\sigma^2}\right) I_0\left(\frac{yv}{\sigma^2}\right)$$

where $y > 0$, $v > 0$, $\sigma > 0$ and $I_0$ is the modified Bessel function of the first kind with order zero. When $v = 0$ the Rice distribution reduces to a Rayleigh distribution. The mean is $\sigma \sqrt{\pi/2} \exp(z/2)((1 - z)I_0(-z/2) - zI_1(-z/2))$ (returned as the fitted values) where $z = -v^2/(2\sigma^2)$. Simulated Fisher scoring is implemented.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

Convergence problems may occur for data where $v = 0$; if so, use rayleigh or possibly use an identity link.

When $v$ is large (greater than 3, say) then the mean is approximately $v$ and the standard deviation is approximately $\sigma$.

Author(s)

T. W. Yee

References


See Also

drice, rayleigh, besseli, simulate.vlm.
Examples

## Not run: sigma <- exp(1); vee <- exp(2)
        rdata <- data.frame(y = rrice(n <- 1000, sigma, vee = vee))
        fit <- vglm(y ~ 1, riceff, data = rdata, trace = TRUE, crit = "coef")
        c(with(rdata, mean(y)), fitted(fit)[1])
        coef(fit, matrix = TRUE)
        Coef(fit)
        summary(fit)

## End(Not run)

---

rioff

### Reciprocal Inverse Gaussian distribution

#### Description

Estimation of the parameters of a reciprocal inverse Gaussian distribution.

#### Usage

rioff(lmu = "identitylink", llambda = "loge", imu = NULL, ilambda = 1)

#### Arguments

- **lmu, llambda**: Link functions for mu and lambda. See `Links` for more choices.
- **imu, ilambda**: Initial values for mu and lambda. A NULL means a value is computed internally.

#### Details

See Jorgensen (1997) for details.

#### Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

#### Note

This distribution is potentially useful for dispersion modelling.

#### Author(s)

T. W. Yee

#### References

Return Level Plot for GEV Fits

Description

A return level plot is constructed for a GEV-type model.

Usage

```
rlplot.egev(object, show.plot = TRUE,
  probability = c((1:9)/100, (1:9)/10, 0.95, 0.99, 0.995, 0.999),
  add.arg = FALSE, xlab = if(log.arg) "Return Period (log-scale)" else
  "Return Period", ylab = "Return Level",
  main = "Return Level Plot",
  pch = par()$pch, pcol.arg = par()$col, pcex = par()$cex,
  llty.arg = par()$lty, lcol.arg = par()$col, llwd.arg = par()$lwd,
  slty.arg = par()$lty, scol.arg = par()$col, slwd.arg = par()$lwd,
  ylim = NULL, log.arg = TRUE, CI = TRUE, epsilon = 1e-05, ...)
```

Arguments

- **object**: A VGAM extremes model of the GEV-type, produced by `vglm` with a family function either "gev" or "egev".
- **show.plot**: Logical. Plot it? If FALSE no plot will be done.
- **probability**: Numeric vector of probabilities used.
- **add.arg**: Logical. Add the plot to an existing plot?
- **xlab**: Caption for the x-axis. See `par`.
- **ylab**: Caption for the y-axis. See `par`.
- **main**: Title of the plot. See `title`.
- **pch**: Plotting character. See `par`.
- **pcol.arg**: Color of the points. See the col argument of `par`.
- **pcex**: Character expansion of the points. See the cex argument of `par`.
- **llty.arg**: Line type. Line type. See the lty argument of `par`.
- **lcol.arg**: Color of the lines. See the col argument of `par`. 

Examples

```
   rdata <- data.frame(y = rchisq(n = 100, df = 14)) # Not 'proper' data!!
   fit <- vglm(y ~ 1, rigff, data = rdata, trace = TRUE)
   fit <- vglm(y ~ 1, rigff, data = rdata, trace = TRUE, eps = 1e-9, crit = "coef")
   summary(fit)
```
The confidence intervals are approximate, being based on finite-difference approximations to derivatives.

Author(s)
T. W. Yee

References
**rrar**

*Nested reduced-rank autoregressive models for multiple time series*

**Description**

Estimates the parameters of a nested reduced-rank autoregressive model for multiple time series.

**Usage**

```r
rrar(ranks = 1, coefstart = NULL)
```

**Arguments**

- **ranks**
  Vector of integers: the ranks of the model. Each value must be at least one and no more than \( M \), where \( M \) is the number of response variables in the time series. The length of \( \text{ranks} \) is the **lag**, which is often denoted by the symbol \( L \) in the literature.

- **coefstart**
  Optional numerical vector of initial values for the coefficients. By default, the family function chooses these automatically.

**Details**

Full details are given in Ahn and Reinsel (1988). Convergence may be very slow, so setting `maxits = 50`, say, may help. If convergence is not obtained, you might like to try inputting different initial values.

Setting `trace = TRUE` in `vglm` is useful for monitoring the progress at each iteration.

**Examples**

```r
gdata <- data.frame(y = regev(n = 100, scale = 2, shape = -0.1))
fit <- vglm(y ~ 1, egev, data = gdata, trace = TRUE)

# Identity link for all parameters:
fit2 <- vglm(y ~ 1, egev(1shape = identitylink, 1scale = identitylink,
                     iscale = 10), data = gdata, trace = TRUE)
coef(fit2, matrix = TRUE)
```

```r
## Not run:
par(mfrow = c(1, 2))
rlplot(fit) -> i1
rlplot(fit2, pcol = "darkorange", lcol = "blue", log.arg = FALSE,
       scol = "darkgreen", slty = "dashed", las = 1) -> i2
range(i2@post$rlplot$upper - i1@post$rlplot$upper) # Should be near 0
range(i2@post$rlplot$lower - i1@post$rlplot$lower) # Should be near 0
```

```r
## End(Not run)
```
Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

This family function should be used within vglm and not with rrvglm because it does not fit into the RR-VGLM framework exactly. Instead, the reduced-rank model is formulated as a VGLM!

A methods function Coef.rrar, say, has yet to be written. It would return the quantities \( a_1, c, \phi, \omega \), etc. as slots, and then show.Coef.rrar would also need to be written.

Author(s)

T. W. Yee

References


See Also

vglm, grain.us.

Examples

```r
## Not run:
year <- seq(1961 + 1/12, 1972 + 10/12, by = 1/12)
par(mar = c(4, 4, 2, 2) + 0.1, mfrow = c(2, 2))
for (ii in 1:4) {
  plot(year, grain.us[, ii], main = names(grain.us)[ii], las = 1,
       type = "l", xlab = "", ylab = "", col = "blue")
  points(year, grain.us[, ii], pch = "*", col = "blue")
}
apply(grain.us, 2, mean) # mu vector
cgrain <- scale(grain.us, scale = FALSE) # Center the time series only
fit <- vglm(cgrain ~ 1, rrar(Ranks = c(4, 1)), trace = TRUE)
summary(fit)

print(fit@misc$Ak1, digits = 2)
print(fit@misc$Cmatrices, digits = 3)
print(fit@misc$Dmatrices, digits = 3)
print(fit@misc$omegahat, digits = 3)
print(fit@misc$Phimatrices, digits = 2)

par(mar = c(4, 4, 2, 2) + 0.1, mfrow = c(4, 1))
for (ii in 1:4) {
  plot(year, fit@misc$Z[, ii], main = paste("Z", ii, sep = ""),
       type = "l", xlab = "", ylab = "", las = 1, col = "blue")
  points(year, fit@misc$Z[, ii], pch = "*", col = "blue")
}
```
Fitting Reduced-Rank Vector Generalized Linear Models (RR-VGLMs)

Description

A reduced-rank vector generalized linear model (RR-VGLM) is fitted. RR-VGLMs are VGLMs but some of the constraint matrices are estimated. In this documentation, $M$ is the number of linear predictors.

Usage

```r
rrvglm(formula, family, data = list(), weights = NULL, subset = NULL,
   na.action = na.fail, etastart = NULL, mustart = NULL,
   coefstart = NULL, control = rrvglm.control(...), offset = NULL,
   method = "rrvglm.fit", model = FALSE, x.arg = TRUE, y.arg = TRUE,
   contrasts = NULL, constraints = NULL, extra = NULL,
   qr.arg = FALSE, smart = TRUE, ...)
```

Arguments

- `formula`, `family`, `weights`
  See `vglm`.
- `data` an optional data frame containing the variables in the model. By default the variables are taken from `environment(formula)`, typically the environment from which `rrvglm` is called.
- `subset`, `na.action`
  See `vglm`.
- `etastart`, `mustart`, `coefstart`
  See `vglm`.
- `control` a list of parameters for controlling the fitting process. See `rrvglm.control` for details.
- `offset`, `model`, `contrasts`
  See `vglm`.
- `method` the method to be used in fitting the model. The default (and presently only) method `rrvglm.fit` uses iteratively reweighted least squares (IRLS).
- `x.arg`, `y.arg` logical values indicating whether the model matrix and response vector/matrix used in the fitting process should be assigned in the `x` and `y` slots. Note the model matrix is the LM model matrix; to get the VGLM model matrix type `model.matrix(vglmfit)` where `vglmfit` is a `vglm` object.
- `constraints`
  See `vglm`.
- `extra`, `smart`, `qr.arg`
  See `vglm`.
- `...` further arguments passed into `rrvglm.control`.
Details

The central formula is given by

$$\eta = B^T x_1 + A \nu$$

where $x_1$ is a vector (usually just a 1 for an intercept), $x_2$ is another vector of explanatory variables, and $\nu = C^T x_2$ is an $R$-vector of latent variables. Here, $\eta$ is a vector of linear predictors, e.g., the $m$th element is $\eta_m = \log(E[Y_m])$ for the $m$th Poisson response. The matrices $B$, $A$ and $C$ are estimated from the data, i.e., contain the regression coefficients. For ecologists, the central formula represents a constrained linear ordination (CLO) since it is linear in the latent variables. It means that the response is a monotonically increasing or decreasing function of the latent variables.

For identifiability it is common to enforce corner constraints on $A$: by default, the top $R$ by $R$ submatrix is fixed to be the order-$R$ identity matrix and the remainder of $A$ is estimated.

The underlying algorithm of RR-VGLMs is iteratively reweighted least squares (IRLS) with an optimizing algorithm applied within each IRLS iteration (e.g., alternating algorithm).

In theory, any VGAM family function that works for `vglm` and `vgam` should work for `rrvglm` too. The function that actually does the work is `rrvglmNfit`; it is `vglmNfit` with some extra code.

Value

An object of class "rrvglm", which has the the same slots as a "vglm" object. The only difference is that the some of the constraint matrices are estimates rather than known. But VGAM stores the models the same internally. The slots of "vglm" objects are described in `vglm-class`.

Note

The arguments of `rrvglm` are in general the same as those of `vglm` but with some extras in `rrvglm.control`.

The smart prediction (`smartpred`) library is packed with the VGAM library.

In an example below, a rank-1 stereotype model of Anderson (1984) is fitted to some car data. The reduced-rank regression is performed, adjusting for two covariates. Setting a trivial constraint matrix for the latent variable variables in $x_2$ avoids a warning message when it is overwritten by a (common) estimated constraint matrix. It shows that German cars tend to be more expensive than American cars, given a car of fixed weight and width.

If `fit <- rrvglm(..., data = mydata)` then `summary(fit)` requires corner constraints and no missing values in `mydata`. Often the estimated variance-covariance matrix of the parameters is not positive-definite; if this occurs, try refitting the model with a different value for `Index.corner`.

For constrained quadratic ordination (CQO) see `cqo` for more details about QRR-VGLMs.

With multiple binary responses, one must use `binomialff(multiple.responses = TRUE)` to indicate that the response is a matrix with one response per column. Otherwise, it is interpreted as a single binary response variable.

Author(s)

Thomas W. Yee
References


See Also

`rrvglm.control`, `lvplot.rrvglm` (same as `biplot.rrvglm`), `rrvglm-class`, `grc`, `cqt`, `vglmff-class`, `vglm`, `vglm-class`, `smartpred.rrvglm.fit`. Special family functions include `negbinomial`, `ziopoisson` and `zinegbinomial`. (see Yee (2014) and COZIGAM). Methods functions include `Coef.rrvglm`, `summary.rrvglm`, etc. Data include `crashi`.

Examples

```r
# Not run:
# Example 1: RR negative binomial with Var(Y) = mu + delta1 * mu^delta2
nn <- 1000       # Number of observations
delta1 <- 3.0     # Specify this
delta2 <- 1.5     # Specify this; should be greater than unity
a21 <- 2 - delta2
mydata <- data.frame(x2 = runif(nn), x3 = runif(nn))
mydata <- transform(mydata, mu = exp(2 + 3 * x2 + 0 * x3))
mydata <- transform(mydata, 
  y2 = rnbinom(nn, mu = mu, size = (1/delta1)*mu^a21))
plot(y2 ~ x2, data = mydata, pch = "+", col = "blue", las = 1,
     main = paste("Var(Y) = mu + ", delta1, "+ mu", delta2, sep = ""))
rrnb2 <- rrvglm(y2 ~ x2 + x3, negbinomial(zero = NULL), 
                 data = mydata, trace = TRUE)

a21.hat <- (Coef(rrnb2)[@A]["log(size)", 1]
beta1.hat <- Coef(rrnb2)[@B]["(Intercept)", "log(mu)"]
beta2.hat <- Coef(rrnb2)[@B]["(Intercept)", "log(size)"]
(delta1.hat <- exp(a21.hat * beta1.hat - beta2.hat))
(delta2.hat <- 2 - a21.hat)
# exp(a21.hat * predict(rrnb2)[1,1] - predict(rrnb2)[1,2])  # delta1.hat
summary(rrnb2)

# Obtain a 95 percent confidence interval for delta2:
se.a21.hat <- sqrt(vcov(rrnb2)["I(latvar.mat)","I(latvar.mat)"])
ci.a21 <- a21.hat + c(-1, 1) * 1.96 * se.a21.hat
(ci.delta2 <- ci - rev(ci.a21))  # The 95 percent confidence interval
Confint.rrnb(rrnb2)  # Quick way to get it

# Plot the abundances and fitted values against the latent variable
```

```r
plot(y2 ~ latvar(rrnb2), data = mydata, col = "blue",
     xlab = "Latent variable", las = 1)
ooo <- order(latvar(rrnb2))
lines(fitted(rrnb2)[ooo] ~ latvar(rrnb2)[ooo], col = "orange")

# Example 2: stereotype model (reduced-rank multinomial logit model)
data(car.all)
sca <- subset(car.all, 
    is.element(Country, c("Germany", "USA", "Japan", "Korea")))
fcols <- c(13,14,18:20,22:26,29:31,33,34,36) # These are factors
sca[, -fcols] <- scale(sca[, -fcols]) # Standardize all numerical vars
ones <- matrix(1, 3, 1)
clist <- list("(Intercept)" = diag(3), Width = ones, Weight = ones,
              Disp. = diag(3), Tank = diag(3), Price = diag(3),
              Frt.Leg.Room = diag(3))
set.seed(111)
fit <- rrvglm(Country ~ Width + Weight + Disp. + Tank +
              Price + Frt.Leg.Room,
              multinomial, data = sca, Rank = 2, trace = TRUE,
              constraints = clist, norrr = ~ 1 + Width + Weight,
              Uncor = TRUE, Corner = FALSE, Bestof = 2)
fit@misc$deviance # A history of the fits
Coef(fit)
biplot(fit, chull = TRUE, scores = TRUE, clty = 2, Ccex = 2,
       ccol = "blue", scol = "orange", Ccol = "darkgreen", Clwd = 2,
       main = "1=Germany, 2=Japan, 3=Korea, 4=USA")

## End(Not run)
```

### rrvglm-class

Class "rrvglm"

**Description**
Reduced-rank vector generalized linear models.

**Objects from the Class**

Objects can be created by calls to `rrvglm`.

**Slots**

- **extra**: Object of class "list"; the extra argument on entry to vglm. This contains any extra information that might be needed by the family function.
- **family**: Object of class "vglmsf". The family function.
- **iter**: Object of class "numeric". The number of IRLS iterations used.
- **predictors**: Object of class "matrix" with $M$ columns which holds the $M$ linear predictors.
- **assign**: Object of class "list", from class "vlm". This named list gives information matching the columns and the (LM) model matrix terms.
call: Object of class "call", from class "vlm". The matched call.
coefficients: Object of class "numeric", from class "vlm". A named vector of coefficients.
constraints: Object of class "list", from class "vlm". A named list of constraint matrices used in the fitting.
contrasts: Object of class "list", from class "vlm". The contrasts used (if any).
control: Object of class "list", from class "vlm". A list of parameters for controlling the fitting process. See vglm.control for details.
criterion: Object of class "list", from class "vlm". List of convergence criterion evaluated at the final IRLS iteration.
df.residual: Object of class "numeric", from class "vlm". The residual degrees of freedom.
df.total: Object of class "numeric", from class "vlm". The total degrees of freedom.
dispersion: Object of class "numeric", from class "vlm". The scaling parameter.
effects: Object of class "numeric", from class "vlm". The effects.
fitted.values: Object of class "matrix", from class "vlm". The fitted values. This is usually the mean but may be quantiles, or the location parameter, e.g., in the Cauchy model.
misc: Object of class "list", from class "vlm". A named list to hold miscellaneous parameters.
model: Object of class "data.frame", from class "vlm". The model frame.
na.action: Object of class "list", from class "vlm". A list holding information about missing values.
offset: Object of class "matrix", from class "vlm". If non-zero, a \( M \)-column matrix of offsets.
post: Object of class "list", from class "vlm" where post-analysis results may be put.
preplot: Object of class "list", from class "vlm" used by plotvgam; the plotting parameters may be put here.
prior.weights: Object of class "matrix", from class "vlm" holding the initially supplied weights.
qr: Object of class "list", from class "vlm". QR decomposition at the final iteration.
R: Object of class "matrix", from class "vlm". The \( \mathbf{R} \) matrix in the QR decomposition used in the fitting.
rank: Object of class "integer", from class "vlm". Numerical rank of the fitted model.
residuals: Object of class "matrix", from class "vlm". The working residuals at the final IRLS iteration.
ResSS: Object of class "numeric", from class "vlm". Residual sum of squares at the final IRLS iteration with the adjusted dependent vectors and weight matrices.
smart.prediction: Object of class "list", from class "vlm". A list of data-dependent parameters (if any) that are used by smart prediction.
terms: Object of class "list", from class "vlm". The terms object used.
weights: Object of class "matrix", from class "vlm". The weight matrices at the final IRLS iteration. This is in matrix-band form.
x: Object of class "matrix", from class "vlm". The model matrix (LM, not VGLM).
xlevels: Object of class "list", from class "vlm". The levels of the factors, if any, used in fitting.
y: Object of class "matrix", from class "vlm". The response, in matrix form.
Xm2: Object of class "matrix", from class "vlm". See vglm-class).
Ym2: Object of class "matrix", from class "vlm". See vglm-class).
callXm2: Object of class "call", from class "vlm". The matched call for argument form2.

Extends
Class "vglm", directly. Class "vlm", by class "vglm".

Methods

biplot signature(x = "rrvglm"): biplot.
Coef signature(object = "rrvglm"): more detailed coefficients giving A, B1, C, etc.
biplot signature(object = "rrvglm"): biplot.
print signature(x = "rrvglm"): short summary of the object.
summary signature(object = "rrvglm"): a more detailed summary of the object.

Note
The slots of "rrvglm" objects are currently identical to "vlm" objects.

Author(s)
Thomas W. Yee

References


See Also

rrvglm, lvplot.rrvglm, vglmff-class.

Examples

## Not run: # Rank-1 stereotype model of Anderson (1984)
pneumo <- transform(pneumo, let = log(exposure.time),
               x3 = runif(nrow(pneumo))) # x3 is unrelated
fit <- rrvglm(cbind(normal, mild, severe) ~ let + x3,
             multinomial, data = pneumo, Rank = 1)
Coef(fit)

## End(Not run)
Control function for rrvglm

Description

Algorithmic constants and parameters for running rrvglm are set using this function.

Usage

```r
rrvglm.control(Rank = 1, Algorithm = c("alternating", "derivative"),
               Corner = TRUE, Uncorrelated.latvar = FALSE,
               Wmat = NULL, Svd.arg = FALSE,
               Index.corner = if (length(str0))
                              head((1:1000)[-str0], Rank) else 1:Rank,
               Ainits = NULL, Alpha = 0.5, Bestof = 1, Cinit = NULL,
               Etamat.colmax = 10,
               sd.Ainit = 0.02, sd.Cinit = 0.02, str0 = NULL,
               noRRR = ~1, Norrr = NA,
               noWarning = FALSE,
               trace = FALSE, Use.Init.Poisson.QO = FALSE,
               checkwz = TRUE, Check.rank = TRUE, Check.cm.rank = TRUE,
               wzepsilon = .Machine$double.eps^0.75, ...)
```

Arguments

- **Rank**
  The numerical rank \( R \) of the model. Must be an element from the set \{1,2,\ldots,\min(M,p2)\}. Here, the vector of explanatory variables \( x \) is partitioned into \( (x_1,x_2) \), which is of dimension \( p1+p2 \). The variables making up \( x_1 \) are given by the terms in noRRR argument, and the rest of the terms comprise \( x_2 \).

- **Algorithm**
  Character string indicating what algorithm is to be used. The default is the first one.

- **Corner**
  Logical indicating whether corner constraints are to be used. This is one method for ensuring a unique solution. If TRUE, Index.corner specifies the \( R \) rows of the constraint matrices that are used as the corner constraints, i.e., they hold an order-\( R \) identity matrix.

- **Uncorrelated.latvar**
  Logical indicating whether uncorrelated latent variables are to be used. This is normalization forces the variance-covariance matrix of the latent variables to be \( \text{diag}(\text{Rank}) \), i.e., unit variance and uncorrelated. This constraint does not lead to a unique solution because it can be rotated.

- **Wmat**
  Yet to be done.

- **Svd.arg**
  Logical indicating whether a singular value decomposition of the outer product is to computed. This is another normalization which ensures uniqueness. See the argument Alpha below.

- **Index.corner**
  Specifies the \( R \) rows of the constraint matrices that are used for the corner constraints, i.e., they hold an order-\( R \) identity matrix.
**Alpha**
The exponent in the singular value decomposition that is used in the first part: if the SVD is $UDV^T$ then the first and second parts are $UD^\alpha$ and $D^{1-\alpha}V^T$ respectively. A value of 0.5 is 'symmetrical'. This argument is used only when Svd.arg=TRUE.

**Bestof**
Integer. The best of Bestof models fitted is returned. This argument helps guard against local solutions by (hopefully) finding the global solution from many fits. The argument works only when the function generates its own initial value for $C$, i.e., when $C$ is not passed in as initial values.

**Ainit, Cinit**
Initial $A$ and $C$ matrices which may speed up convergence. They must be of the correct dimension.

**Etamat.colmax**
Positive integer, no smaller than Rank. Controls the amount of memory used by .Init.Poisson.QO(). It is the maximum number of columns allowed for the pseudo-response and its weights. In general, the larger the value, the better the initial value. Used only if Use.Init.Poisson.QO=TRUE.

**str0**
Integer vector specifying which rows of the estimated constraint matrices ($A$) are to be all zeros. These are called *structural zeros*. Must not have any common value with Index.corner, and be a subset of the vector $1:M$. The default, str0 = NULL, means no structural zero rows at all.

**sd.Ainit, sd.Cinit**
Standard deviation of the initial values for the elements of $A$ and $C$. These are normally distributed with mean zero. This argument is used only if Use.Init.Poisson.QO = FALSE.

**norrR**
Formula giving terms that are *not* to be included in the reduced-rank regression. That is, norrR specifies which explanatory variables are in the $x_1$ vector of rrvglm, and the rest go into $x_2$. The $x_1$ variables constitute the $B_1$ matrix in Yee and Hastie (2003). Those $x_2$ variables which are subject to the reduced-rank regression correspond to the $B_2$ matrix. Set norrR = NULL for the reduced-rank regression to be applied to every explanatory variable including the intercept.

**Norrr**
Defunct. Please use norrR. Use of Norrr will become an error soon.

**trace**
Logical indicating if output should be produced for each iteration.

**Use.Init.Poisson.QO**
Logical indicating whether the .Init.Poisson.QO() should be used to obtain initial values for the $C$. The function uses a new method that can work well if the data are Poisson counts coming from an equal-tolerances QRR-VGLM (CQO). This option is less realistic for RR-VGLMs compared to QRR-VGLMs.

**checkwz**
Logical indicating whether the diagonal elements of the working weight matrices should be checked whether they are sufficiently positive, i.e., greater than wzepsilon. If not, any values less than wzepsilon are replaced with this value.

**noWarning, Check.rank, Check.cm.rank**
Same as vglm.control. Ignored for VGAM 0.9-7 and higher.

**wzepsilon**
Small positive number used to test whether the diagonals of the working weight matrices are sufficiently positive.

... Variables in ... are passed into vglm.control. If the derivative algorithm is used then ... are also passed into rrvglm.optim.control; and if the alternating algorithm is used then ... are also passed into valt.control.

In the above, $R$ is the Rank and $M$ is the number of linear predictors.
Details

VGAM supports three normalizations to ensure a unique solution. Of these, only corner constraints will work with summary of RR-VGLM objects.

Value

A list with components matching the input names. Some error checking is done, but not much.

Note

The arguments in this function begin with an upper case letter to help avoid interference with those of vglm.control.

In the example below a rank-1 stereotype model (Anderson, 1984) is fitted.

Author(s)

Thomas W. Yee

References


See Also

rrvglm, rrvglm.optim.control, rrvglm-class, vglm, vglm.control, cqt.

Examples

```r
## Not run:
set.seed(111)
pneumo <- transform(pneumo, let = log(exposure.time),
                    x3 = runif(nrow(pneumo)))  # x3 is random noise
fit <- rrvglm(cbind(normal, mild, severe) ~ let + x3,
              multinomial, data = pneumo, Rank = 1, Index.corner = 2)
constraints(fit)
vcov(fit)
summary(fit)
## End(Not run)
```
Control function for rrvglm() calling optim()

Description

Algorithmic constants and parameters for running optim within rrvglm are set using this function.

Usage

```
rrvglm.optim.control(Fnscale = 1, Maxit = 100,
                      Switch.optimizer = 3, Abstol = -Inf,
                      Reltol = sqrt(.Machine$double.eps), ...)
```

Arguments

- `Fnscale`: Passed into optim as fnscale.
- `Maxit`: Passed into optim as maxit.
- `Switch.optimizer`: Iteration number when the "Nelder-Mead" method of optim is switched to the quasi-Newton "BFGS" method. Assigning `Switch.optimizer` a negative number means always BFGS, while assigning `Switch.optimizer` a value greater than `Maxit` means always use Nelder-Mead.
- `Abstol`: Passed into optim as abstol.
- `Reltol`: Passed into optim as reltol.
- `...`: Ignored.

Details

See optim for more details.

Value

A list with components equal to the arguments.

Note

The transition between optimization methods may be unstable, so users may have to vary the value of `Switch.optimizer`.

Practical experience with `Switch.optimizer` shows that setting it to too large a value may lead to a local solution, whereas setting it to a low value will obtain the global solution. It appears that, if BFGS kicks in too late when the Nelder-Mead algorithm is starting to converge to a local solution, then switching to BFGS will not be sufficient to bypass convergence to that local solution.

Author(s)

Thomas W. Yee
See Also

rrvglm.control, optim.

### ruge

**Rutherford-Geiger Polonium Data**

**Description**

Decay counts of polonium recorded by Rutherford and Geiger (1910).

**Usage**

data(ruge)

**Format**

This data frame contains the following columns:

- **counts**: a numeric vector, counts or frequencies
- **number**: a numeric vector, the number of decays

**Details**

These are the radioactive decay counts of polonium recorded by Rutherford and Geiger (1910) representing the number of scintillations in 2608 1/8 minute intervals. For example, there were 57 frequencies of zero counts. The counts can be thought of as being approximately Poisson distributed.

**Source**


**Examples**

```r
lambdahat <- with(ruge, weighted.mean(number, w = counts))
(N <- with(ruge, sum(counts)))
with(ruge, cbind(number, counts,
    fitted = round(N * dpois(number, lam = lambdahat))))
```
Defining smooths in VGAM formulae

Description

`s` is used in the definition of (vector) smooth terms within `vgam` formulae.

Usage

`s(x, df = 4, spar = 0, ...)`

Arguments

- **x**: covariate (abscissae) to be smoothed. Note that `x` must be a `single` variable and not a function of a variable. For example, `s(x)` is fine but `s(log(x))` will fail. In this case, let `logx <- log(x)` (in the data frame), say, and then use `s(logx)`. At this stage bivariate smoothers (`x` would be a two-column matrix) are not implemented.
- **df**: numerical vector of length `r`. Effective degrees of freedom: must lie between 1 (linear fit) and `n` (interpolation). Thus one could say that `df-1` is the *nonlinear degrees of freedom* of the smooth. Recycling of values will be used if `df` is not of length `r`. If `spar` is positive then this argument is ignored.
- **spar**: numerical vector of length `r`. Positive smoothing parameters (after scaling). Larger values mean more smoothing so that the solution approaches a linear fit for that component function. A zero value means that `df` is used. Recycling of values will be used if `spar` is not of length `r`.
- **...**: Ignored for now.

Details

In this help file `M` is the number of additive predictors and `r` is the number of component functions to be estimated (so that `r` is an element from the set `{1,2,...,M}`). Also, if `n` is the number of *distinct* abscissae, then `s` will fail if `n < 7`.

`s`, which is symbolic and does not perform any smoothing itself, only handles a single covariate. Note that `s` works in `vgam` only. It has no effect in `vglm` (actually, it is similar to the identity function `I` so that `s(x2)` is the same as `x2` in the LM model matrix). It differs from the `s` of the `gam` and `mgcv` packages; they should not be mixed together. Also, terms involving `s` should be simple additive terms, and not involving interactions and nesting etc. For example, `myfactor:s(x2)` is not a good idea.

Value

A vector with attributes that are (only) used by `vgam`. 
Note

The vector cubic smoothing spline which \( s() \) represents is computationally demanding for large \( M \). The cost is approximately \( O(nM^3) \) where \( n \) is the number of unique abscissae.

An alternative to using \( s \) with \texttt{vgam} is \texttt{bs} and/or \texttt{ns} with \texttt{vglm}. The latter implements half-stepping, which is helpful if convergence is difficult.

Author(s)

Thomas W. Yee

References


See Also

\texttt{vgam}, \texttt{is.buggy}, \texttt{vsmooth.spline}.

Examples

```r
# Nonparametric logistic regression
fit1 <- vgam(agaus ~ s(altitude, df = 2), binomialff, data = hunua)
## Not run: plot(fit1, se = TRUE)

# Bivariate logistic model with artificial data
nn <- 300
bdata <- data.frame(x1 = runif(nn), x2 = runif(nn))
bdata <- transform(bdata,
  y1 = rbinom(nn, size = 1, prob = logit(sin(2 * x2), inverse = TRUE)),
  y2 = rbinom(nn, size = 1, prob = logit(sin(2 * x2), inverse = TRUE)))
fit2 <- vgam(cbind(y1, y2) ~ x1 + s(x2, 3), trace = TRUE,
  binom2.or(exchangeable = TRUE), data = bdata)
coef(fit2, matrix = TRUE)  # Hard to interpret
## Not run: plot(fit2, se = TRUE, which.term = 2, scol = "blue")
```

---

**sc.studentt2**

\textit{Scaled Student t Distribution with 2 df Family Function}

Description

Estimates the location and scale parameters of a scaled Student t distribution with 2 degrees of freedom, by maximum likelihood estimation.

Usage

```r
sc.studentt2(percentile = 50, llocation = "identitylink", lscale = "loge",
  ilocation = NULL, iscale = NULL, imethod = 1, zero = 2)
```
Arguments

percentile  A numerical vector containing values between 0 and 100, which are the quantiles and expectiles. They will be returned as ‘fitted values’.

location, iscale

See Links for more choices, and CommonVGAMffArguments.

imethod, zero

See CommonVGAMffArguments for details.

Details

Koenker (1993) solved for the distribution whose quantiles are equal to its expectiles. Its canonical form has mean and mode at 0, and has a heavy tail (in fact, its variance is infinite).

The standard (“canonical”) form of this distribution can be endowed with a location and scale parameter. The standard form has a density that can be written as

\[ f(z) = \frac{2}{(4 + z^2)^{3/2}} \]

for real \( y \). Then \( z = (y - a)/b \) for location and scale parameters \( a \) and \( b > 0 \). The mean of \( Y \) is \( a \). By default, \( \eta_1 = a \) and \( \eta_2 = \log(b) \). The expectiles/quantiles corresponding to percentile are returned as the fitted values; in particular, percentile \( = 50 \) corresponds to the mean (0.5 expectile) and median (0.5 quantile).

Note that if \( Y \) has a standard \( \text{dsc.t2} \) then \( Y = \sqrt{2} T_2 \) where \( T_2 \) has a Student-t distribution with 2 degrees of freedom. The two parameters here can also be estimated using \( \text{studentt2} \) by specifying \( \text{df} = 2 \) and making an adjustment for the scale parameter, however, this VGAM family function is more efficient since the EIM is known (Fisher scoring is implemented.)

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Author(s)

T. W. Yee

References


See Also

dsc.t2, studentt2.

Examples

set.seed(123); nn <- 1000
kdata <- data.frame(x2 = sort(runif(nn)))
kdata <- transform(kdata, mylocat = 1 + 3 * x2,
                    myscale = 1)
kdata <- transform(kdata, y = rsc.t2(nn, loc = mylocat, scale = myscale))
fit <- vglm(y ~ x2, sc.studentt2(perc = c(1, 50, 99)), data = kdata)
fit2 <- vglm(y ~ x2, studentt2(df = 2), data = kdata) # 'same' as fit

coef(fit, matrix = TRUE)
head(fitted(fit))
head(predict(fit))

# Nice plot of the results
## Not run: plot(y ~ x2, data = kdata, col = "blue", las = 1,
## sub = paste("Fitted quantiles/expectiles using the sc.studentt2() distribution"),
## main = "Fitted quantiles/expectiles using the sc.studentt2() distribution")
matplot(with(kdata, x2), fitted(fit), add = TRUE, type = "l", lwd = 3)
legend("bottomright", lty = 1:3, lwd = 3, legend = colnames(fitted(fit)),
       col = 1:3)
## End(Not run)

fit@extra$percentile # Sample quantiles

---

**Select**

Select variables for a formula response or the RHS of a formula

**Description**

Select variables from a data frame whose names begin with a certain character string.

**Usage**

```r
data = list(), prefix = "y",
   lhs = NULL, rhs = NULL, rhs2 = NULL, rhs3 = NULL,
   as.character = FALSE, as.formula.arg = FALSE, tilde = TRUE,
   exclude = NULL, sort.arg = TRUE)
```

**Arguments**

- `data` A data frame or a matrix.
- `prefix` A vector of character strings, or a logical. If a character then the variables chosen from `data` begin with the value of `prefix`. If a logical then only `TRUE` is accepted and all the variables in `data` are chosen.
- `lhs` A character string. The response of a formula.
- `rhs` A character string. Included as part of the RHS a formula. Set `rhs` = "\( \emptyset \)" to suppress the intercept.
- `rhs2, rhs3` Same as `rhs` but appended to its RHS, i.e., `paste0(rhs, " + ", rhs2, " + ", rhs3)`.
  If used, `rhs` should be used first, and then possibly `rhs2` and then possibly `rhs3`.
- `as.character` Logical. Return the answer as a character string?
- `as.formula.arg` Logical. Is the answer a formula?
tilde Logical. If as.character and as.formula.arg are both TRUE then include the tilde in the formula?
exclude Vector of character strings. Exclude these variables explicitly.
sort.arg Logical. Sort the variables?

Details
This is meant as a utility function to avoid manually: (i) making a cbind call to construct a big matrix response, and (ii) constructing a formula involving a lot of terms. The savings can be made because the variables of interest begin with some prefix, e.g., with the character "y".

Value
If as.character = FALSE and as.formula.arg = FALSE then a matrix such as cbind(y1, y2, y3).
If as.character = TRUE and as.formula.arg = FALSE then a character string such as "cbind(y1, y2, y3)".
If as.character = FALSE and as.formula.arg = TRUE then a formula such as \( \text{lhs} \sim y_1 + y_2 + y_3 \).
If as.character = TRUE and as.formula.arg = TRUE then a character string such as \( \text{lhs} \sim y_1 + y_2 + y_3 \).
See the examples below. By default, if no variables beginning the the value of prefix is found then a NULL is returned. Setting prefix = "" is a way of selecting no variables.

Note
This function is a bit experimental at this stage and may change in the short future. Some of its utility may be better achieved using subset and its select argument, e.g., subset(pdata, TRUE, select = y01:y10).
For some models such as posbernoulli.t the order of the variables in the xij argument is crucial, therefore care must be taken with the argument sort.arg. In some instances, it may be good to rename variables y1 to y01, y2 to y02, etc. when there are variables such as y14.
Currently subsetcol() and Select() are identical. One of these functions might be withdrawn in the future.

Author(s)
T. W. Yee.

See Also
vglm, cbind, subset, formula, fill.

Examples

```r
t <- tnorm

cbind(t, t, y)
```

```r
Select(data = tnorm) # Same as with(tnorm, cbind(y1, y2, y3))
Select(t, "x")
Select(t, "x", sort = FALSE, as.char = TRUE)
Select(t, "x", exclude = "x1")
Select(t, "x", exclude = "x1", as.char = TRUE)
```
Select(Pneumo, c("x", "y"))
Select(Pneumo, "z") # Now returns a NULL
Select(Pneumo, "") # Now returns a NULL
Select(Pneumo, prefix = TRUE, as.formula = TRUE)
Select(Pneumo, "x", exclude = c("x3", "x1"), as.formula = TRUE,
   lhs = "cbind(y1, y2, y3)", rhs = "0")
Select(Pneumo, "x", exclude = "x1", as.formula = TRUE, as.char = TRUE,
   lhs = "cbind(y1, y2, y3)", rhs = "0")

# Now a 'real' example:
Huggins89table1 <- transform(Huggins89table1, x3.tij = t01)
tab1 <- subset(Huggins89table1,
   rowSums(Select(Huggins89table1, "y")) > 0)
# Same as
# subset(Huggins89table1, y1 + y2 + y3 + y4 + y5 + y6 + y7 + y8 + y9 + y10 > 0)

# Long way to do it:
fit.th <-
   vglm(cbind(y01, y02, y03, y04, y05, y06, y07, y08, y09, y10) ~ x2 + x3.tij,
   xij = list(x3.tij ~ t01 + t02 + t03 + t04 + t05 + t06 + t07 + t08 +
   t09 + t10 - 1),
   posbernoulli.t(parallel.t = TRUE ~ x2 + x3.tij),
   data = tab1, trace = TRUE,
   form2 = ~ x2 + x3.tij + t01 + t02 + t03 + t04 + t05 + t06 + t07 + t08 +
   t09 + t10)
# Short way to do it:
Fit.th <- vglm(Select(tab1, "y") ~ x2 + x3.tij,
   xij = list(Select(tab1, "t", as.formula = TRUE, 
   sort = FALSE, lhs = "x3.tij", rhs = "0")),
   posbernoulli.t(parallel.t = TRUE ~ x2 + x3.tij),
   data = tab1, trace = TRUE,
   form2 = Select(tab1, prefix = TRUE, as.formula = TRUE))

seq2binomial

The Two-stage Sequential Binomial Distribution Family Function

Description

Estimation of the probabilities of a two-stage binomial distribution.

Usage

seq2binomial(lprob1 = "logit", lprob2 = "logit",
   iprob1 = NULL, iprob2 = NULL,
   parallel = FALSE, zero = NULL)

Arguments

lprob1, lprob2 Parameter link functions applied to the two probabilities, called \( p \) and \( q \) below. See \text{Links} for more choices.
seq2binomial

iprob1, iprob2  Optional initial value for the first and second probabilities respectively. A NULL means a value is obtained in the initialize slot.

parallel, zero  Details at Links. If parallel = TRUE then the constraint also applies to the intercept.

Details

This VGAM family function fits the model described by Crowder and Sweeting (1989) which is described as follows. Each of \( m \) spores has a probability \( p \) of germinating. Of the \( y_1 \) spores that germinate, each has a probability \( q \) of bending in a particular direction. Let \( y_2 \) be the number that bend in the specified direction. The probability model for this data is

\[
P(y_1, y_2) = \binom{m}{y_1} p^{y_1} (1 - p)^{m - y_1} \binom{y_1}{y_2} q^{y_2} (1 - q)^{y_1 - y_2}
\]

for \( 0 < p < 1, 0 < q < 1, y_1 = 1, \ldots, m \) and \( y_2 = 1, \ldots, y_1 \). Here, \( p \) is \( \text{prob1} \), \( q \) is \( \text{prob2} \).

Although the Authors refer to this as the bivariate binomial model, I have named it the (two-stage) sequential binomial model. Fisher scoring is used.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

The response must be a two-column matrix of sample proportions corresponding to \( y_1 \) and \( y_2 \). The \( m \) values should be inputted with the weights argument of vglm and vgam. The fitted value is a two-column matrix of estimated probabilities \( p \) and \( q \). A common form of error is when there are no trials for \( y_1 \), e.g., if mvector below has some values which are zero.

Author(s)

Thomas W. Yee

References


See Also

binomialff, cfibrosis.

Examples

sdata <- data.frame(mvector = round(rnorm(nn <- 100, m = 10, sd = 2)), x2 = runif(nn))
sdata <- transform(sdata, prob1 = logit(+2 + x2, inverse = TRUE), prob2 = logit(-2 + x2, inverse = TRUE))
setup.smart

Smart Prediction Setup

Description

Sets up smart prediction in one of two modes: "write" and "read".

Usage

setup.smart(mode.arg, smart.prediction = NULL, max.smart = 30)

Arguments

mode.arg  mode.arg must be "write" or "read". If in "read" mode then smart.prediction must be assigned the data structure .smart.prediction that was created while fitting. This is stored in object@smart.prediction or object$smart.prediction where object is the name of the fitted object.

smart.prediction  If in "read" mode then smart.prediction must be assigned the list of data dependent parameters, which is stored on the fitted object. Otherwise, smart.prediction is ignored.

max.smart  max.smart is the initial length of the list .smart.prediction. It is not important because .smart.prediction is made larger if needed.

Details

This function is only required by programmers writing a modelling function such as lm and glm, or a prediction functions of such, e.g., predict.lm. The function setup.smart operates by mimicking the operations of a first-in first-out stack (better known as a queue).

```r
sdata <- transform(sdata, successes1 = rbinom(nn, size = mvector, prob = prob1))
sdata <- transform(sdata, successes2 = rbinom(nn, size = successes1, prob = prob2))
sdata <- transform(sdata, y1 = successes1 / mvector)
sdata <- transform(sdata, y2 = successes2 / successes1)
fit <- vglm(cbind(y1, y2) ~ x2, seq2binomial, weight = mvector, data = sdata, trace = TRUE)
coef(fit)
coef(fit, matrix = TRUE)
head(fitted(fit))
head(depvar(fit))
head(weights(fit, type = "prior"))  # Same as with(sdata, mvector)
  # Number of first successes:
  head(depvar(fit)[, 1] * c(weights(fit, type = "prior")))
  # Number of second successes:
  head(depvar(fit)[, 2] * c(weights(fit, type = "prior")) * depvar(fit)[, 1])
```
Simplex Distribution

Description

Density function, and random generation for the simplex distribution.

Usage

dsimplex(x, mu = 0.5, dispersion = 1, log = FALSE)
rsimplex(n, mu = 0.5, dispersion = 1)

Arguments

- **x**: Vector of quantiles. The support of the distribution is the interval (0, 1).
- **mu, dispersion**: Mean and dispersion parameters. The former lies in the interval (0, 1) and the latter is positive.
- **n, log**: Same usage as `runif`.

Value

Nothing is returned.

Side Effects

In "write" mode .smart.prediction in smartpredenv is assigned an empty list with max.smart components. In "read" mode .smart.prediction in smartpredenv is assigned smart.prediction. Then .smart.prediction.counter in smartpredenv is assigned the value 0, and .smart.prediction.mode and .max.smart are written to smartpredenv too.

See Also

`lm, predict.lm`.

Examples

```r
## Not run:
ssetup.smart("write")  # Put at the beginning of lm
## End(Not run)

## Not run: # Put at the beginning of predict.lm
setup.smart("read", smart.prediction = object$smart.prediction)
## End(Not run)
```
Details

The `VGAM` family function `simplex` fits this model; see that online help for more information. For `rsimplex()` the rejection method is used; it may be very slow if the density is highly peaked, and will fail if the density asymptotes at the boundary.

Value

dsimplex(x) gives the density function, rsimplex(n) gives n random variates.

Author(s)

T. W. Yee

See Also

`simplex`.

Examples

```r
sigma <- c(4, 2, 1)  # Dispersion parameter
mymu <- c(0.1, 0.5, 0.7); xxx <- seq(0, 1, len = 501)
## Not run: par(mfrow = c(3, 3))  # Figure 2.1 of Song (2007)
for (iii in 1:3)
  for (jjj in 1:3)
    plot(xxx, dsimplex(xxx, mymu[jjj], sigma[iii]),
         type = "l", col = "blue", xlab = "", ylab = "", main =
         paste("mu = ", mymu[jjj], ", sigma = ", sigma[iii], sep = ""))
## End(Not run)
```

Description

The two parameters of the univariate standard simplex distribution are estimated by full maximum likelihood estimation.

Usage

```r
simplex(lmu = "logit", lsigma = "loge",
       imu = NULL, isigma = NULL,
       imethod = 1, ishrinkage = 0.95, zero = 2)
```
Arguments

- imu, isigma: Link function for mu and sigma. See Links for more choices.
- imu, isigma: Optional initial values for mu and sigma. A NULL means a value is obtained internally.
- imethod, ishrinkage, zero

See CommonVGAMffArguments for more information.

Details

The probability density function can be written

\[ f(y; \mu, \sigma) = \left[2\pi\sigma^2(y(1-y))^3\right]^{-0.5} \exp\left[-0.5(y - \mu)^2/(\sigma^2 y(1-y)\mu^2(1-\mu)^2)\right] \]

for \(0 < y < 1, 0 < \mu < 1, \) and \(\sigma > 0.\) The mean of \(Y\) is \(\mu\) (called mu, and returned as the fitted values).

The second parameter, sigma, of this standard simplex distribution is known as the dispersion parameter. The unit variance function is \(V(\mu) = \mu^3(1-\mu)^3.\) Fisher scoring is applied to both parameters.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

This distribution is potentially useful for dispersion modelling. Numerical problems may occur when mu is very close to 0 or 1.

Author(s)

T. W. Yee

References


See Also

dsimplex, dirichlet, rig, binomialff.

Examples

```r
sdata <- data.frame(x2 = runif(nn <- 1000))
sdata <- transform(sdata, eta1 = 1 + 2 * x2, eta2 = 1 - 2 * x2)
sdata <- transform(sdata, y = rsimplex(nn, mu = logit(eta1, inverse = TRUE), dispersion = exp(eta2)))
```
simulate.vlm

Simulate Responses for VGLMs and VGAMs

Description

Simulate one or more responses from the distribution corresponding to a fitted model object.

Usage

```r
## S3 method for class 'vglm'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

- `object` an object representing a fitted model. Usually an object of class `vglm-class` or `vgam-class`.
- `nsim, seed` Same as `simulate`.
- `...` additional optional arguments.

Details

This is a methods function for `simulate` and hopefully should behave in a very similar manner. Only VGAM family functions with a `simslot` slot have been implemented for `simulate`.

Value

Similar to `simulate`. Note that many VGAM family functions can handle multiple responses. This can result in a longer data frame with more rows (nsim multiplied by n rather than the ordinary n). In the future an argument may be available so that there is always n rows no matter how many responses were inputted.

Warning

With multiple response and/or multivariate responses, the order of the elements may differ. For some VGAM families, the order is \( n \times N \times F \), where \( n \) is the sample size, \( N \) is nsim and \( F \) is `ncol(fitted(vglmObject))`. For other VGAM families, the order is \( n \times F \times N \). An example of each is given below.

```r
(fit <- vglm(y ~ x2, simplex(zero = NULL), data = sdata, trace = TRUE))
coef(fit, matrix = TRUE)
summary(fit)
```
See Also

Currently the VGAM family functions with a simslot slot are: alaplace1, alaplace2, betabinomial, betabinomialff, betaR, betaF, biamhcop, bifrankcop, bilogistic, binomialff, binormal, binormalcop, biclaytoncop, cauchy, cauchy1, chisq, dirichlet, dagum, erlang, exponential, bifgmcop, fisk, gamma1, gamma2, gammaR, gengamma, stacy, geometric, gompertz, gumbelII, hzeta, inv.lomax, inv.paralogistic, kumar, lgamma1, lgamma3, lindley, lino, logff, logistic1, logistic, lognormal, lomax, makeham, negbinomial, negbinomial.size, paralognormal, poissonff, posnegbinomial, posnormal, pospoisson, polya, polyaR, posbinomial, rayleigh, riceff, simplex, sinmad, slash, studentt, studentt2, studentt3, triangle, uninormal, yulesimon, zageometric, zageometricff, zanegbinomial, zaposbinomial, zapossonff, zageometric, zageometricff, zinegbinomial, zipt, zipoisson, zipoissonff.

See also RNG about random number generation in R, vglm, vgam for model fitting.

Examples

```r
nn <- 10; mysize <- 20; set.seed(123)
bdata <- data.frame(x2 = rnorm(nn))
bdata <- transform(bdata,
y1 = rbinom(nn, size = mysize, p = logit(1+x2, inverse = TRUE)),
y2 = rbinom(nn, size = mysize, p = logit(1+x2, inverse = TRUE)),
f1 = factor(as.numeric(rbinom(nn, size = 1,
p = logit(1+x2, inverse = TRUE))))
)
(fit1 <- vglm(cbind(y1, aaa = mysize - y1) ~ x2, # Matrix response (2-cols)
binomialff, data = bdata))
(fit2 <- vglm(f1 ~ x2, binomialff, model = TRUE, data = bdata)) # Factor response

set.seed(123); simulate(fit1, nsim = 8)
set.seed(123); c(simulate(fit2, nsim = 3)) # Use c() when model = TRUE

# An n x N x F example
set.seed(123); n <- 100
bdata <- data.frame(x2 = runif(n), x3 = runif(n))
bdata <- transform(bdata, y1 = rnorm(n, 1 + 2 * x2),
y2 = rnorm(n, 3 + 4 * x2))
(fit1 <- vglm(cbind(y1, y2) ~ x2, binormal(eq.sd = TRUE), data = bdata))
snim <- 1000 # Number of simulations for each observation
my.sims <- simulate(fit1, nsim = nsim)
dim(my.sims) # A data frame
aaa <- array(unlist(my.sims), c(n, nsim, ncol(fitted(fit1)))) # n by N by F
summary(rowMeans(aaa[, , 1]) - fitted(fit1[, 1])) # Should be all 0s
summary(rowMeans(aaa[, , 2]) - fitted(fit1[, 2])) # Should be all 0s

# An n x F x N example
n <- 100; set.seed(111); nsim <- 1000
zdata <- data.frame(x2 = runif(n))
zdata <- transform(zdata, lambda1 = loge(-0.5 + 2 * x2, inverse = TRUE),
lambda2 = loge( 0.5 + 2 * x2, inverse = TRUE),
pstr01 = logit( 0, inverse = TRUE),
pstr02 = logit(-1.0, inverse = TRUE))
zdata <- transform(zdata, y1 = rzipois(n, lambda = lambda1, pstr0 = pstr01),
y2 = rzipois(n, lambda = lambda2, pstr0 = pstr02))
```
The Singh-Maddala Distribution

Description
Density, distribution function, quantile function and random generation for the Singh-Maddala distribution with shape parameters \( a \) and \( q \), and scale parameter \( \text{scale} \).

Usage
```r
dsinmad(x, scale = 1, shape1.a, shape3.q, log = FALSE)
psinmad(q, scale = 1, shape1.a, shape3.q, lower.tail = TRUE, log.p = FALSE)
qsinmad(p, scale = 1, shape1.a, shape3.q, lower.tail = TRUE, log.p = FALSE)
rsinmad(n, scale = 1, shape1.a, shape3.q)
```

Arguments
- `x`, `q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
- `shape1.a`, `shape3.q` shape parameters.
- `scale` scale parameter.
- `log` Logical. If `log = TRUE` then the logarithm of the density is returned.
- `lower.tail`, `log.p` Same meaning as in `pnorm` or `qnorm`.

Details
See `sinmad`, which is the VGAM family function for estimating the parameters by maximum likelihood estimation.

Value
dsinmad gives the density, psinmad gives the distribution function, qsinmad gives the quantile function, and rsinmad generates random deviates.

```r
zip.fit <- vglm(chbind(y1, y2) ~ x2, zipoissonff, data = zdata, crit = "coef")
my.sims <- simulate(zip.fit, nsim = nsim)
dim(my.sims) # A data frame
aaa <- array(unlist(my.sims), c(n, ncol(fitted(zip.fit)), nsim)) # n by F by N
summary(rowMeans(aaa[, 1, ])) - fitted(zip.fit)[, 1]) # Should be all 0s
summary(rowMeans(aaa[, 2, ]) - fitted(zip.fit)[, 2]) # Should be all 0s
```
Note

The Singh-Maddala distribution is a special case of the 4-parameter generalized beta II distribution.

Author(s)

T. W. Yee and Kai Huang

References


See Also

`sinmad`, `genbetaII`.

Examples

```r
sdata <- data.frame(y = rsinmad(n = 3000, scale = exp(2),
                         shape1 = exp(1), shape3 = exp(1)))
fit <- vglm(y ~ 1, sinmad(lss = FALSE, ishape1.a = 2.1), data = sdata,
            trace = TRUE, crit = "coef")
coef(fit, matrix = TRUE)
Coef(fit)
```

---

**sinmad**

*Singh-Maddala Distribution Family Function*

Description

Maximum likelihood estimation of the 3-parameter Singh-Maddala distribution.

Usage

```r
sinmad(lscale = "log", lshape1.a = "log", lshape3.q = "log",
      iscale = NULL, ishape1.a = NULL, ishape3.q = NULL, imethod = 1,
      lss = TRUE, gscale = exp(-5:5), gshape1.a = exp(-5:5),
      gshape3.q = exp(-5:5), probs.y = c(0.25, 0.5, 0.75),
      zero = ifelse(lss, -(2:3), -.c(1, 3)))
```

Arguments

- `lss` See `CommonVGAMffArguments` for important information.
- `lshape1.a`, `lscale`, `lshape3.q`
  Parameter link functions applied to the (positive) parameters `a`, `scale`, and `q`. See `Links` for more choices.
The 3-parameter Singh-Maddala distribution is the 4-parameter generalized beta II distribution with shape parameter \( p = 1 \). It is known under various other names, such as the Burr XII (or just the Burr distribution), Pareto IV, beta-P, and generalized log-logistic distribution. More details can be found in Kleiber and Kotz (2003).

Some distributions which are special cases of the 3-parameter Singh-Maddala are the Lomax \((a = 1)\), Fisk \((q = 1)\), and paralogistic \((a = q)\).

The Singh-Maddala distribution has density

\[
f(y) = aqy^{a-1} / [b^a \{1 + (y/b)^a\}^{1+q}]\]

for \( a > 0, b > 0, q > 0, y \geq 0 \). Here, \( b \) is the scale parameter, and the others are shape parameters. The cumulative distribution function is

\[
F(y) = 1 - [1 + (y/b)^a]^{-q}.
\]

The mean is

\[
E(Y) = b \Gamma(1+1/a) \Gamma(q-1/a)/\Gamma(q)
\]

provided \(-a < 1 < aq\); these are returned as the fitted values. This family function handles multiple responses.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

See the notes in genbetaII.

Author(s)

T. W. Yee

References

Skellam

The Skellam Distribution

Description

Density and random generation for the Skellam distribution.

Usage

\[ \text{dskellam}(x, \mu_1, \mu_2, \text{log} = \text{FALSE}) \]
\[ \text{rskellam}(n, \mu_1, \mu_2) \]
Arguments

- `x`: vector of quantiles.
- `n`: number of observations. Same as `runif`.
- `mu1, mu2`: See `skellam`.
- `log`: Logical; if TRUE, the logarithm is returned.

Details

See `skellam`, the VGAM family function for estimating the parameters, for the formula of the probability density function and other details.

Value

dskellam gives the density, and rskellam generates random deviates.

Warning

Numerical problems may occur for data if \( \mu_1 \) and/or \( \mu_2 \) are large. The normal approximation for this case has not been implemented yet.

See Also

`skellam, dpois`.

Examples

```r
## Not run
mu1 <- 1; mu2 <- 2; x <- (-7):7
plot(x, dskellam(x, mu1, mu2), type = "h", las = 1, col = "blue",
     main = paste("Density of Skellam distribution with mu1 = ", mu1,
                 " and mu2 = ", mu2, sep = ""))
## End(Not run)
```

Description

Estimates the two parameters of a Skellam distribution by maximum likelihood estimation.

Usage

```r
skellam(lmu1 = "loge", lmu2 = "loge", imu1 = NULL, imu2 = NULL,
        nsimEIM = 100, parallel = FALSE, zero = NULL)
```
Arguments

\texttt{lmu1, lmu2} \quad \text{Link functions for the } \mu_1 \text{ and } \mu_2 \text{ parameters. See \texttt{Links} for more choices and for general information.}

\texttt{imu1, imu2} \quad \text{Optional initial values for the parameters. See \texttt{CommonVGAMffArguments} for more information. If convergence failure occurs (this \texttt{VGAM} family function seems to require good initial values) try using these arguments.}

\texttt{nsimEIM, parallel, zero} \quad \text{See \texttt{CommonVGAMffArguments} for more information. In particular, setting parallel=TRUE will constrain the two means to be equal.}

Details

The Skellam distribution models the difference between two independent Poisson distributions (with means \( \mu_j \), say). It has density function

\[ f(y; \mu_1, \mu_2) = \left( \frac{\mu_1}{\mu_2} \right)^{y/2} \exp(-\mu_1 - \mu_2) I_{|y|} \left(2\sqrt{\mu_1 \mu_2}\right) \]

where \( y \) is an integer, \( \mu_1 > 0, \mu_2 > 0 \). Here, \( I_v \) is the modified Bessel function of the first kind with order \( v \).

The mean is \( \mu_1 - \mu_2 \) (returned as the fitted values), and the variance is \( \mu_1 + \mu_2 \). Simulated Fisher scoring is implemented.

Value

An object of class "vglmff" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm} and \texttt{vgam}.

Warning

This \texttt{VGAM} family function seems fragile and very sensitive to the initial values. Use very cautiously!!

Note

Numerical problems may occur for data if \( \mu_1 \) and/or \( \mu_2 \) are large.

References


See Also

dskellam, dpois, poissonff.
Examples

```r
## Not run:
sdata <- data.frame(x2 = runif(nn <- 1000))
sdata <- transform(sdata, mu1 = exp(1 + x2), mu2 = exp(1 + x2))
sdata <- transform(sdata, y = rskellam(nn, mu1, mu2))
fit1 <- vglm(y ~ x2, skellam, data = sdata, trace = TRUE, crit = "coef")
fit2 <- vglm(y ~ x2, skellam(parallel = TRUE), data = sdata, trace = TRUE)
coef(fit1, matrix = TRUE)
coef(fit2, matrix = TRUE)
summary(fit1)
# Likelihood ratio test for equal means:
pchisq(2 * (logLik(fit1) - logLik(fit2)),
       df = df.residual(fit2) - df.residual(fit1), lower.tail = FALSE)
lrtest(fit1, fit2) # Alternative
## End(Not run)
```

skewnorm

*Skew-Normal Distribution*

**Description**

Density and random generation for the univariate skew-normal distribution.

**Usage**

```r
dskewnorm(x, location = 0, scale = 1, shape = 0, log = FALSE)
rskewnorm(n, location = 0, scale = 1, shape = 0)
```

**Arguments**

- `x`: vector of quantiles.
- `n`: number of observations. Same as `runif`.
- `location`: The location parameter $\xi$. A vector.
- `scale`: The scale parameter $\omega$. A positive vector.
- `shape`: The shape parameter. It is called $\alpha$ in `skewnormal`.
- `log`: Logical. If log=TRUE then the logarithm of the density is returned.

**Details**

See `skewnormal`, which currently only estimates the shape parameter. More generally here, $Z = \xi + \omega Y$ where $Y$ has a standard skew-normal distribution (see `skewnormal`), $\xi$ is the location parameter and $\omega$ is the scale parameter.

**Value**

dskewnorm gives the density, rskewnorm generates random deviates.
Note

The default values of all three parameters corresponds to the skew-normal being the standard normal distribution.

Author(s)

T. W. Yee

References

http://tango.stat.unipd.it/SN.

See Also

skewnormal.

Examples

```r
## Not run: N <- 200  # Grid resolution
shape <- 7; x <- seq(-4, 4, len = N)
plot(x, dskewnorm(x, shape = shape), type = "l", col = "blue", las = 1,
     ylab = "", lty = 1, lwd = 2)
abline(v = 0, h = 0, col = "grey")
lines(x, dnorm(x), col = "orange", lty = 2, lwd = 2)
legend("topleft", leg = c(paste("Blue = dskewnorm(x,"", shape,""),
         "Orange = standard normal density"), lty = 1:2, lwd = 2,
         col = c("blue", "orange"))
## End(Not run)
```

skewnormal  

Univariate Skew-Normal Distribution Family Function

Description

Maximum likelihood estimation of the shape parameter of a univariate skew-normal distribution.

Usage

```r
skewnormal(lshape = "identitylink", ishape = NULL, nsimEIM = NULL)
```

Arguments

lshape, ishape, nsimEIM

See Links and CommonVGAMffArguments.
Details

The univariate skew-normal distribution has a density function that can be written

\[ f(y) = 2 \phi(y) \Phi(\alpha y) \]

where \( \alpha \) is the shape parameter. Here, \( \phi \) is the standard normal density and \( \Phi \) its cumulative distribution function. When \( \alpha = 0 \) the result is a standard normal distribution. When \( \alpha = 1 \) it models the distribution of the maximum of two independent standard normal variates. When the absolute value of the shape parameter increases the skewness of the distribution increases. The limit as the shape parameter tends to positive infinity results in the folded normal distribution or half-normal distribution. When the shape parameter changes its sign, the density is reflected about \( y = 0 \).

The mean of the distribution is \( \mu = \alpha \sqrt{2/(\pi(1 + \alpha^2))} \) and these are returned as the fitted values. The variance of the distribution is \( 1 - \mu^2 \). The Newton-Raphson algorithm is used unless the nsimeim argument is used.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

It is well known that the EIM of Azzalini’s skew-normal distribution is singular for skewness parameter tending to zero, and thus produces influential problems.

Note

It is a good idea to use several different initial values to ensure that the global solution is obtained.

This family function will be modified (hopefully soon) to handle a location and scale parameter too.

Author(s)

Thomas W. Yee

References


See Also

skewnorn, uninormal, foldnormal.
Examples

```r
ddata <- data.frame(y1 = rskewnorm(nn <- 1000, shape = 5))
f1 <- vglm(y1 ~ 1, skewnormal, data = ddata, trace = TRUE)
coef(f1, matrix = TRUE)
head(fitted(f1), 1)
with(ddata, mean(y1))
## Not run: with(ddata, hist(y1, prob = TRUE))
x <- with(ddata, seq(min(y1), max(y1), len = 200))
with(ddata, lines(x, dskewnorm(x, shape = coef(f1)), col = "blue"))
## End(Not run)

sdata <- data.frame(x2 = runif(nn))
sdata <- transform(ddata, y2 = rskewnorm(nn, shape = 1 + 2*x2))
f2 <- vglm(y2 ~ x2, skewnormal, data = sdata, trace = TRUE, crit = "coef")
summary(f2)
```

Slash

**Slash Distribution**

Description

Density function, distribution function, and random generation for the slash distribution.

Usage

```r
dslash(x, mu = 0, sigma = 1, log = FALSE, smallno = .Machine$double.eps*1000)
pslash(q, mu = 0, sigma = 1, very.negative = -10000,
     lower.tail = TRUE, log.p = FALSE)
rslash(n, mu = 0, sigma = 1)
```

Arguments

- `x, q` vector of quantiles.
- `n` Same as `runif`.
- `mu, sigma` the mean and standard deviation of the univariate normal distribution.
- `log` Logical. If TRUE then the logarithm of the density is returned.
- `very.negative` Numeric, of length 1. A large negative value. For \((q-mu)/sigma\) values less than this, the value 0 is returned because \texttt{integrate} tends to fail. A warning is issued. Similarly, if \((q-mu)/sigma\) is greater than abs(very.negative) then 1 is returned with a warning.
- `smallno` See \texttt{slash}.
- `lower.tail, log.p` Same meaning as in \texttt{pnorm} or \texttt{qnorm}.  

slash

**Slash Distribution**

Density function, distribution function, and random generation for the slash distribution.

Usage

```r
dslash(x, mu = 0, sigma = 1, log = FALSE, smallno = .Machine$double.eps*1000)
pslash(q, mu = 0, sigma = 1, very.negative = -10000,
     lower.tail = TRUE, log.p = FALSE)
rslash(n, mu = 0, sigma = 1)
```

Arguments

- `x, q` vector of quantiles.
- `n` Same as `runif`.
- `mu, sigma` the mean and standard deviation of the univariate normal distribution.
- `log` Logical. If TRUE then the logarithm of the density is returned.
- `very.negative` Numeric, of length 1. A large negative value. For \((q-mu)/sigma\) values less than this, the value 0 is returned because \texttt{integrate} tends to fail. A warning is issued. Similarly, if \((q-mu)/sigma\) is greater than abs(very.negative) then 1 is returned with a warning.
- `smallno` See \texttt{slash}.
- `lower.tail, log.p` Same meaning as in \texttt{pnorm} or \texttt{qnorm}.
Details

See slash, the VGAM family function for estimating the two parameters by maximum likelihood estimation, for the formula of the probability density function and other details.

Function pslash uses a for () loop and integrate, meaning it's very slow. It may also be inaccurate for extreme values of q, and returns with 1 or 0 values when too extreme compared to very.negative.

Value

dslash gives the density, and pslash gives the distribution function, rslash generates random deviates.

Note

pslash is very slow.

Author(s)

Thomas W. Yee and C. S. Chee

See Also

slash.

Examples

```r
## Not run:
curve(dslash, col = "blue", ylab = "f(x)", -5, 5, ylim = c(0, 0.4), las = 1,
   main = "Standard slash, normal and Cauchy densities", lwd = 2)
curve(dnorm, col = "black", lty = 2, lwd = 2, add = TRUE)
curve(dcauchy, col = "orange", lty = 3, lwd = 2, add = TRUE)
legend("topleft", c("slash", "normal", "Cauchy"), lty = 1:3,
   col = c("blue","black","orange"), lwd = 2)
curve(pslash, col = "blue", -5, 5, ylim = 0:1)
pslash(c(-Inf, -20000, 20000, Inf)) # Gives a warning

## End(Not run)
```
Usage

```r
slash(lmu = "identitylink", lsigma = "loge",
    imu = NULL, isigma = NULL, iprobs = c(0.1, 0.9), nsimEIM = 250,
    zero = NULL, smallno = .Machine$double.eps*1000)
```

Arguments

- **lmu, lsigma**: Parameter link functions applied to the \( \mu \) and \( \sigma \) parameters, respectively. See Links for more choices.
- **imu, isigma**: Initial values. A NULL means an initial value is chosen internally. See CommonVGAMffArguments for more information.
- **iprobs**: Used to compute the initial values for \( \mu \). This argument is fed into the probs argument of quantile, and then a grid between these two points is used to evaluate the log-likelihood. This argument must be of length two and have values between 0 and 1.
- **nsimEIM, zero**: See CommonVGAMffArguments for more information.
- **smallno**: Small positive number, used to test for the singularity.

Details

The standard slash distribution is the distribution of the ratio of a standard normal variable to an independent standard uniform(0,1) variable. It is mainly of use in simulation studies. One of its properties is that it has heavy tails, similar to those of the Cauchy.

The general slash distribution can be obtained by replacing the univariate normal variable by a general normal \( N(\mu, \sigma) \) random variable. It has a density that can be written as

\[ f(y) = \begin{cases} 
  \frac{1}{(2\sigma \sqrt{2\pi})} & \text{if } y = \mu, \\
  1 - \exp(-(((y - \mu)/\sigma)^2)/2))/((\sqrt{2\pi} )\sigma((y - \mu)/\sigma)^2) & \text{if } y \neq \mu.
\end{cases} \]

where \( \mu \) and \( \sigma \) are the mean and standard deviation of the univariate normal distribution respectively.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

Fisher scoring using simulation is used. Convergence is often quite slow. Numerical problems may occur.

Author(s)

T. W. Yee and C. S. Chee
smart.expression

References


See Also

rslash, simulate.vlm.

Examples

## Not run:
sdata <- data.frame(y = rslash(n = 1000, mu = 4, sigma = exp(2)))
fit <- vglm(y ~ 1, slash, data = sdata, trace = TRUE)
coef(fit, matrix = TRUE)
Coeff(fit)
summary(fit)

## End(Not run)
Determine What Mode the Smart Prediction is In

Description
Determine which of three modes the smart prediction is currently in.

Usage
smart.mode.is(mode.arg = NULL)

Arguments
mode.arg a character string, either "read", "write" or "neutral".

Details
Smart functions such as bs and poly need to know what mode smart prediction is in. If it is in "write" mode then the parameters are saved to .smart.prediction using put.smart. If in "read" mode then the parameters are read in using get.smart. If in "neutral" mode then the smart function behaves like an ordinary function.

Value
If mode.arg is given, then either TRUE or FALSE is returned. If mode.arg is not given, then the mode ("neutral", "read" or "write") is returned. Usually, the mode is "neutral".

See Also
put.smart, bs, poly.

Examples
print(sm.min1)
smart.mode.is() # Returns "neutral"
smart.mode.is(smart.mode.is()) # Returns TRUE
Description

Data-dependent parameters in formula terms can cause problems in when predicting. The **smartpred** package saves data-dependent parameters on the object so that the bug is fixed. The `lm` and `glm` functions have been fixed properly. Note that the **VGAM** package by T. W. Yee automatically comes with smart prediction.

Usage

```r
sm.bs(x, df = NULL, knots = NULL, degree = 3, intercept = FALSE,
      Boundary.knots = range(x))
sm.ns(x, df = NULL, knots = NULL, intercept = FALSE,
      Boundary.knots = range(x))
sm.poly(x, ..., degree = 1, coefs = NULL, raw = FALSE)
sm.scale(x, center = TRUE, scale = TRUE)
```

Arguments

- `x` - The `x` argument is actually common to them all.
- `df`, `knots`, `intercept`, `Boundary.knots`
  - See `bs` and/or `ns`.
- `degree`, `...`, `coefs`, `raw`
  - See `poly`.
- `center`, `scale` - See `scale`.

Details

R version 1.6.0 introduced a partial fix for the prediction problem because it does not work all the time, e.g., for terms such as `I(poly(x, 3))`, `poly(c(scale(x), 3), bs(scale(x), 3), scale(scale(x)))`. See the examples below. Smart prediction, however, will always work.

The basic idea is that the functions in the formula are now smart, and the modelling functions make use of these smart functions. Smart prediction works in two ways: using `smart.expression`, or using a combination of `put.smart` and `get.smart`.

Value

The usual value returned by `bs, ns, poly` and `scale`. When used with functions such as `vglm` the data-dependent parameters are saved on one slot component called `smart.prediction`. 
Side Effects

The variables .max.smart, .smart.prediction and .smart.prediction.counter are created while the model is being fitted. They are created in a new environment called smartpredenv. These variables are deleted after the model has been fitted. However, if there is an error in the model fitting function or the fitting model is killed (e.g., by typing control-C) then these variables will be left in smartpredenv. At the beginning of model fitting, these variables are deleted if present in smartpredenv.

During prediction, the variables .smart.prediction and .smart.prediction.counter are re-constructed and read by the smart functions when the model frame is re-evaluated. After prediction, these variables are deleted.

If the modelling function is used with argument smart = FALSE (e.g., vglm(..., smart = FALSE)) then smart prediction will not be used, and the results should match with the original R functions.

WARNING

The functions bs, ns, poly and scale are now left alone (from 2014-05 onwards) and no longer smart. They work via safe prediction. The smart versions of these functions have been renamed and they begin with "sm."

The functions predict.bs and predict.ns are not smart. That is because they operate on objects that contain attributes only and do not have list components or slots. The function predict.poly is not smart.

Author(s)

T. W. Yee and T. J. Hastie

See Also

get.smart.prediction, get.smart.put.smart, smart.expression, smart.mode.is, setup.smart, wrapup.smart. Commonly used data-dependent functions include scale, poly, bs, ns. In R, the functions bs and ns are in the splines package, and this library is automatically loaded in because it contains compiled code that bs and ns call.

The functions vglm, vgam, rrvglm and cqi in T. W. Yee's VGAM package are examples of modelling functions that employ smart prediction.

Examples

# Create some data first
n <- 20
set.seed(86)  # For reproducibility of the random numbers
ldata <- data.frame(x2 = sort(runif(n)), y = sort(runif(n)))
library("splines")  # To get ns() in R

# This will work for R 1.6.0 and later
fit <- lm(y ~ ns(x2, df = 5), data = ldata)
## Not run:
plot(y ~ x2, data = ldata)
lines(fitted(fit) ~ x2, data = ldata)
sratio <- data.frame(x2 = seq(0, 1, len = n))
points(predict(fit, new.ldata) - x2, new.ldata, type = "b", col = 2, err = -1)

## End(Not run)

# The following fails for R 1.6.x and later. It can be
# made to work with smart prediction provided
# ns is changed to sm.ns and scale is changed to sm.scale:
fit1 <- lm(y ~ ns(scale(x2), df = 5), data = ldata)
## Not run:
plot(y ~ x2, data = ldata, main = "Safe prediction fails")
lines(fitted(fit1) ~ x2, data = ldata)
points(predict(fit1, new.ldata) ~ x2, new.ldata, type = "b", col = 2, err = -1)

## End(Not run)

# Fit the above using smart prediction
## Not run:
library("VGAM")  # The following requires the VGAM package to be loaded
fit2 <- vglm(y ~ sm.ns(sm.scale(x2), df = 5), uninormal, data = ldata)
fit2@smart.prediction
plot(y ~ x2, data = ldata, main = "Smart prediction")
lines(fitted(fit2) ~ x2, data = ldata)
points(predict(fit2, new.ldata, type = "response") ~ x2, data = new.ldata,
       type = "b", col = 2, err = -1)

## End(Not run)

sratio

**Ordinal Regression with Stopping Ratios**

### Description

Fits a stopping ratio logit/probit/cloglog/cauchit/... regression model to an ordered (preferably) factor response.

### Usage

sratio(link = "logit", parallel = FALSE, reverse = FALSE, zero = NULL, whitespace = FALSE)

### Arguments

- **link**
  - Link function applied to the $M$ stopping ratio probabilities. See **Links** for more choices.
- **parallel**
  - A logical, or formula specifying which terms have equal/unequal coefficients.
- **reverse**
  - Logical. By default, the stopping ratios used are $\eta_j = \text{logit}(P[Y = j | Y \geq j])$ for $j = 1, \ldots, M$. If reverse is TRUE, then $\eta_j = \text{logit}(P[Y = j+1 | Y \leq j+1])$ will be used.
zero An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \{1, 2, \ldots, M\}. The default value means none are modelled as intercept-only terms.

whitespace See CommonVGAMffArguments for information.

Details

In this help file the response $Y$ is assumed to be a factor with ordered values $1, 2, \ldots, M + 1$, so that $M$ is the number of linear/additive predictors $\eta_j$.

There are a number of definitions for the continuation ratio in the literature. To make life easier, in the VGAM package, we use continuation ratios (see cratio) and stopping ratios. Continuation ratios deal with quantities such as $\logit(P[Y>j|Y=j])$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

No check is made to verify that the response is ordinal if the response is a matrix; see ordered.

Note

The response should be either a matrix of counts (with row sums that are all positive), or a factor. In both cases, the $y$ slot returned by vglm/vgam/rrvglm is the matrix of counts.

For a nominal (unordered) factor response, the multinomial logit model (multinomial) is more appropriate.

Here is an example of the usage of the parallel argument. If there are covariates $x_1$, $x_2$ and $x_3$, then parallel = TRUE $\sim x_1 + x_2 -1$ and parallel = FALSE $\sim x_3$ are equivalent. This would constrain the regression coefficients for $x_1$ and $x_2$ to be equal; those of the intercepts and $x_3$ would be different.

Author(s)

Thomas W. Yee

References


See Also

`cratio`, `acat`, `cumulative`, `multinomial`, `pneumo`, `logit`, `probit`, `cloglog`, `cauchit`.

Examples

```r
pneumo <- transform(pneumo, let = log(exposure.time))
(fit <- vglm(cbind(normal, mild, severe) ~ let,
    sratio(parallel = TRUE), data = pneumo))
coef(fit, matrix = TRUE)
constraints(fit)
predict(fit)
predict(fit, untransform = TRUE)
```

---

studentt  

**Student t Distribution**

Description

Estimating the parameters of a Student t distribution.

Usage

```r
studentt (ldf = "loglog", idf = NULL, tol1 = 0.1, imethod = 1)
studentt2(df = Inf, ilocation = "identitylink", lscale = "loge",
    ilocation = NULL, iscale = NULL, imethod = 1, zero = -2)
studentt3(llocation = "identitylink", lscale = "loge", ldf = "loglog",
    ilocation = NULL, iscale = NULL, idf = NULL,
    imethod = 1, zero = -(2:3))
```

Arguments

- `ldf`, `lscale`, `ldf`
  Parameter link functions for each parameter, e.g., for degrees of freedom $\nu$. See `links` for more choices. The default ensures the parameters are in range. A `loglog` link keeps the degrees of freedom greater than unity; see below.

- `ilocation`, `iscale`, `idf`
  Optional initial values. If given, the values must be in range. The default is to compute an initial value internally.

- `tol1`
  A positive value, the tolerance for testing whether an initial value is 1. Best to leave this argument alone.

- `df`
  Numeric, user-specified degrees of freedom. It may be of length equal to the number of columns of a response matrix.

- `imethod`, `zero`
  See `CommonVGAMffArguments`.

Details

The Student t density function is

\[ f(y; \nu) = \frac{\Gamma((\nu + 1)/2)}{\sqrt{\nu\pi}\Gamma(\nu/2)} \left(1 + \frac{y^2}{\nu}\right)^{-(\nu+1)/2} \]

for all real y. Then \( E(Y) = 0 \) if \( \nu > 1 \) (returned as the fitted values), and \( Var(Y) = \nu/(\nu - 2) \) for \( \nu > 2 \). When \( \nu = 1 \) then the Student t-distribution corresponds to the standard Cauchy distribution, \texttt{cauchy1}. When \( \nu = 2 \) with a scale parameter of \( \sqrt{2} \) then the Student t-distribution corresponds to the standard (Koenker) distribution, \texttt{sc.studentt2}. The degrees of freedom can be treated as a parameter to be estimated, and as a real and not an integer. The Student t distribution is used for a variety of reasons in statistics, including robust regression.

Let \( Y = (T - \mu)/\sigma \) where \( \mu \) and \( \sigma \) are the location and scale parameters respectively. Then \texttt{studentt3} estimates the location, scale and degrees of freedom parameters. And \texttt{studentt2} estimates the location, scale parameters for a user-specified degrees of freedom, \( df \). And \texttt{studentt} estimates the degrees of freedom parameter only. The fitted values are the location parameters. By default the linear/additive predictors are \((\mu, \log(\sigma), \log(\log(\nu))^T\) or subsets thereof.

In general convergence can be slow, especially when there are covariates.

Value

An object of class “\texttt{vglmff}” (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, and \texttt{vgam}.

Note

\texttt{studentt3()} and \texttt{studentt2()} can handle multiple responses.

Practical experience has shown reasonably good initial values are required. If convergence failure occurs try using arguments such as \texttt{idf}. Local solutions are also possible, especially when the degrees of freedom is close to unity or the scale parameter is close to zero.

A standard normal distribution corresponds to a t distribution with infinite degrees of freedom. Consequently, if the data is close to normal, there may be convergence problems; best to use \texttt{uninormal} instead.

Author(s)

T. W. Yee

References


See Also

\texttt{uninormal, cauchy1, logistic, huber2, sc.studentt2, TDist, simulate.vlm}
Examples

tdata <- data.frame(x2 = runif(nn <- 1000))
tdata <- transform(tdata, y1 = rt(nn, df = exp(exp(0.5 - x2))),
                  y2 = rt(nn, df = exp(exp(0.5 - x2))))
fit1 <- vglm(y1 ~ x2, studentt, data = tdata, trace = TRUE)
coef(fit1, matrix = TRUE)

fit2 <- vglm(y1 ~ x2, studentt2(df = exp(exp(0.5))), data = tdata)
coef(fit2, matrix = TRUE)  # df inputted into studentt2() not quite right

fit3 <- vglm(cbind(y1, y2) ~ x2, studentt3, data = tdata, trace = TRUE)
coef(fit3, matrix = TRUE)

---

**SURff**

Seemingly Unrelated Regressions

Description

Fits a system of seemingly unrelated regressions.

Usage

SURff(mle.normal = FALSE,
       divisor = c("n", "n-max(pj,pk)", "sqrt((n-pj)*(n-pk))"),
       parallel = FALSE, Varcov = NULL, matrix.arg = FALSE)

Arguments

- **mle.normal** Logical. If TRUE then the MLE, assuming multivariate normal errors, is computed; the effect is just to add a loglikelihood slot to the returned object. Then it results in the maximum likelihood estimator.

- **divisor** Character, partial matching allowed and the first choice is the default. The divisor for the estimate of the covariances. If "n" then the estimate will be biased. If the others then the estimate will be unbiased for some elements. If mle.normal = TRUE and this argument is not "n" then a warning or an error will result.

- **parallel** See CommonVGAMffArguments. If parallel = TRUE then the constraint applies to the intercept too.

- **Varcov** Numeric. This may be assigned a variance-covariance of the errors. If matrix.arg then this is a $M \times M$ matrix. If !matrix.arg then this is a $M \times M$ matrix in matrix-band format (a vector with at least $M$ and at most $M*(M+1)/2$ elements).

- **matrix.arg** Logical. Of single length.
Details

Proposed by Zellner (1962), the basic seemingly unrelated regressions (SUR) model is a set of LMs ($M > 1$ of them) tied together at the error term level. Each LM’s model matrix may potentially have its own set of predictor variables.

Zellner’s efficient (ZEF) estimator (also known as Zellner’s two-stage Aitken estimator) can be obtained by setting \texttt{maxit = 1} (and possibly \texttt{divisor = “sqrt”} or \texttt{divisor = “n-max”}).

The default value of \texttt{maxit} (in \texttt{vglm.control}) probably means iterative GLS (IGLS) estimator is computed because IRLS will probably iterate to convergence. IGLS means, at each iteration, the residuals are used to estimate the error variance-covariance matrix, and then the matrix is used in the GLS. The IGLS estimator is also known as Zellner’s iterative Aitken estimator, or IZEF.

Value

An object of class "vglmmff" (see \texttt{vglmmff-class}). The object is used by modelling functions such as \texttt{vglm} and \texttt{vgam}.

Warning

The default convergence criterion may be a little loose. Try setting \texttt{epsilon = 1e-11}, especially with \texttt{mle.normal = TRUE}.

Note

The fitted object has slot \texttt{@extra$ncols.X.lm} which is a $M$ vector with the number of parameters for each LM. Also, \texttt{@misc$values.divisor} is the $M$-vector of divisor values.

Constraint matrices are needed in order to specify which response variables that each term on the RHS of the formula is a regressor for. See the \texttt{constraints} argument of \texttt{vglm} for more information.

Author(s)

T. W. Yee.

References


See Also

\texttt{uninormal, gew}. 
Examples

```r
# Obtain some of the results of p.1199 of Kmenta and Gilbert (1968)
clist <- list("(Intercept)" = diag(2),
               "capital.g" = rbind(1, 0),
               "value.g" = rbind(1, 0),
               "capital.w" = rbind(0, 1),
               "value.w" = rbind(0, 1))
zef1 <- vglm(cbind(invest.g, invest.w) ~
capital.g + value.g + capital.w + value.w,
SURff(divisor = "sqrt"), maxit = 1,
data = gew, trace = TRUE, constraints = clist)

round(coef(zef1, matrix = TRUE), digits = 4) # ZEF
zef1@extra$ncoels.X.lm
zef1@misc$divisor
round(sqrt(diag(vcov(zef1))), digits = 4) # SEs
nobs(zef1, type = "lm")
df.residual(zef1, type = "lm")

mle1 <- vglm(cbind(invest.g, invest.w) ~
capital.g + value.g + capital.w + value.w,
SURff(mle.normal = TRUE),
epsilon = 1e-11,
data = gew, trace = TRUE, constraints = clist)
round(coef(mle1, matrix = TRUE), digits = 4) # MLE
round(sqrt(diag(vcov(mle1))), digits = 4) # SEs
```

---

**SurvS4**

Create a Survival Object

---

**Description**

Create a survival object, usually used as a response variable in a model formula.

**Usage**

```r
SurvS4(time, time2, event, type =, origin = 0)
is.SurvS4(x)
```

**Arguments**

- `time` for right censored data, this is the follow up time. For interval data, the first argument is the starting time for the interval.
- `x` any R object.
The status indicator, normally 0=alive, 1=dead. Other choices are TRUE/FALSE (TRUE = death) or 1/2 (2=death). For interval censored data, the status indicator is 0=right censored, 1=event at time, 2=left censored, 3=interval censored. Although unusual, the event indicator can be omitted, in which case all subjects are assumed to have an event.

Ending time of the interval for interval censored or counting process data only. Intervals are assumed to be open on the left and closed on the right, [start, end]. For counting process data, event indicates whether an event occurred at the end of the interval.

Character string specifying the type of censoring. Possible values are "right", "left", "counting", "interval", or "interval2". The default is "right" or "counting" depending on whether the time2 argument is absent or present, respectively.

For counting process data, the hazard function origin. This is most often used in conjunction with a model containing time dependent strata in order to align the subjects properly when they cross over from one strata to another.

Typical usages are

SurvS4(time, event)
SurvS4(time, time2, event, type=, origin=0)

In theory it is possible to represent interval censored data without a third column containing the explicit status. Exact, right censored, left censored and interval censored observation would be represented as intervals of (a,a), (a, infinity), (-infinity,b), and (a,b) respectively; each specifying the interval within which the event is known to have occurred.

If type = "interval2" then the representation given above is assumed, with NA taking the place of infinity. If 'type="interval"' event must be given. If event is 0, 1, or 2, the relevant information is assumed to be contained in time, the value in time2 is ignored, and the second column of the result will contain a placeholder.

Presently, the only methods allowing interval censored data are the parametric models computed by survreg, so the distinction between open and closed intervals is unimportant. The distinction is important for counting process data and the Cox model.

The function tries to distinguish between the use of 0/1 and 1/2 coding for left and right censored data using if (max(status)==2). If 1/2 coding is used and all the subjects are censored, it will guess wrong. Use 0/1 coding in this case.

An object of class SurvS4 (formerly Surv). There are methods for print, is.na, and subscripting survival objects. SurvS4 objects are implemented as a matrix of 2 or 3 columns.

In the case of is.SurvS4, a logical value TRUE if x inherits from class "SurvS4", otherwise a FALSE.
Note

The purpose of having SurvS4 in VGAM is so that the same input can be fed into vglm as functions in survival such as survreg. The class name has been changed from "Surv" to "SurvS4"; see SurvS4-class.

The format J+ is interpreted in VGAM as $\geq J$. If type="interval" then these should not be used in VGAM: ($L, U]$ or ($L, U+$).

Author(s)

The code and documentation comes from survival. Slight modifications have been made for conversion to S4 by T. W. Yee. Also, for "interval" data, as.character.SurvS4() has been modified to print intervals of the form (start, end] and not [start, end] as previously. (This makes a difference for discrete data, such as for cens.poisson). All VGAM family functions beginning with "cen" require the packaging function Surv to format the input.

See Also

SurvS4-class, cens.poisson, survreg, leukemia.

Examples

```r
with(leukemia, SurvS4(time, status))
class(with(leukemia, SurvS4(time, status)))
```

---

### SurvS4-class

#### Class "SurvS4"

**Description**

S4 version of the Surv class.

**Objects from the Class**

A virtual Class: No objects may be created from it.

**Extends**


**Methods**

`show` signature(object = "SurvS4"): ...
Warning

This code has not been thoroughly tested.

Note

The purpose of having SurvS4 in VGAM is so that the same input can be fed into vglm as functions in survival such as survreg.

Author(s)

T. W. Yee.

References

See survival.

See Also

SurvS4.

Examples

showClass("SurvS4")

Tikuv A Short-tailed Symmetric Distribution

Description

Density, cumulative distribution function, quantile function and random generation for the short-tailed symmetric distribution of Tiku and Vaughan (1999).

Usage

dtikuv(x, d, mean = 0, sigma = 1, log = FALSE)
ptikuv(q, d, mean = 0, sigma = 1, lower.tail = TRUE, log.p = FALSE)
qtikuv(p, d, mean = 0, sigma = 1, lower.tail = TRUE, log.p = FALSE, ...)
rtikuv(n, d, mean = 0, sigma = 1, Smallno = 1.0e-6)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. Same as in runif.
d, mean, sigma arguments for the parameters of the distribution. See tikuv for more details. For rtikuv, arguments mean and sigma must be of length 1.
Smallno Numeric, a small value used by the rejection method for determining the lower and upper limits of the distribution. That is, \( ptikuv(L) < \text{Smallno} \) and \( ptikuv(U) > 1 - \text{Smallno} \) where \( L \) and \( U \) are the lower and upper limits respectively.

... Arguments that can be passed into \code{uniroot}.

\code{log} Logical. If \code{log = TRUE} then the logarithm of the density is returned. \code{lower.tail, log.p}

Same meaning as in \code{pnorm} or \code{qnorm}.

**Details**

See \code{tikuv} for more details.

**Value**

\code{dtikuv} gives the density, \code{ptikuv} gives the cumulative distribution function, \code{qtikuv} gives the quantile function, and \code{rtikuv} generates random deviates.

**Author(s)**

T. W. Yee and Kai Huang

**See Also**

\code{tikuv}.

**Examples**

```r
## Not run: par(mfrow = c(2, 1))
x <- seq(-5, 5, len = 401)
plot(x, dnorm(x), type = "l", col = "black", ylab = "", las = 1,
     main = "Black is standard normal, others are \code{dtikuv}(x, d)")
lines(x, dtikuv(x, d = -10), col = "orange")
lines(x, dtikuv(x, d = 1), col = "blue")
legend("topleft", col = c("orange","blue","green"), lty = rep(1, len = 3),
       legend = paste("d = ", c(-10, 1)))

plot(x, pnorm(x), type = "l", col = "black", ylab = "", las = 1,
     main = "Black is standard normal, others are \code{ptikuv}(x, d)")
lines(x, ptikuv(x, d = -10), col = "orange")
lines(x, ptikuv(x, d = 1), col = "blue")
lines(x, ptikuv(x, d = 1), col = "green")
legend("topleft", col = c("orange","blue","green"), lty = rep(1, len = 3),
       legend = paste("d = ", c(-10, 1)))
## End(Not run)

probs <- seq(0.1, 0.9, by = 0.1)
ptikuv(qtikuv(p = probs, d = 1), d = 1) - probs # Should be all 0
```
Short-tailed Symmetric Distribution Family Function

Description

Fits the short-tailed symmetric distribution of Tiku and Vaughan (1999).

Usage

\texttt{tikuv(d, lmean = "identitylink", lsigma = "loge", isigma = NULL, zero = 2)}

Arguments

- \texttt{d}: The \texttt{d} parameter. It must be a single numeric value less than 2. Then \( h = 2 - d > 0 \) is another parameter.
- \texttt{lmean, lsigma}: Link functions for the mean and standard deviation parameters of the usual univariate normal distribution (see Details below). They are \( \mu \) and \( \sigma \) respectively. See Links for more choices.
- \texttt{isigma}: Optional initial value for \( \sigma \). A NULL means a value is computed internally.
- \texttt{zero}: An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The values must be from the set \{1,2\} corresponding respectively to \( \mu, \sigma \). If \texttt{zero} = NULL then all linear/additive predictors are modelled as a linear combination of the explanatory variables. For many data sets having \texttt{zero} = 2 is a good idea.

Details

The short-tailed symmetric distribution of Tiku and Vaughan (1999) has a probability density function that can be written

\[
f(y) = \frac{K}{\sqrt{2\pi}\sigma} \left[ 1 + \frac{1}{2h} \left( \frac{y - \mu}{\sigma} \right)^2 \right]^2 \exp \left( -\frac{1}{2} \left( \frac{y - \mu}{\sigma} \right)^2 \right)
\]

where \( h = 2 - d > 0 \), \( K \) is a function of \( h \), \(-\infty < y < \infty, \sigma > 0 \). The mean of \( Y \) is \( E(Y) = \mu \) and this is returned as the fitted values.

Value

An object of class "\texttt{vglmff}" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, and \texttt{vgam}.

Warning

Under- or over-flow may occur if the data is ill-conditioned, e.g., when \texttt{d} is very close to 2 or approaches \texttt{-Inf}. 

The density function is the product of a univariate normal density and a polynomial in the response $y$. The distribution is bimodal if $d > 0$, else is unimodal. A normal distribution arises as the limit as $d$ approaches $-\infty$, i.e., as $h$ approaches $\infty$. Fisher scoring is implemented. After fitting the value of $d$ is stored in @misc with component name $d$.

Author(s)

Thomas W. Yee

References


See Also

dtikuv, uninormal.

Examples

```r
m <- 1.0; sigma <- exp(0.5)
tdata <- data.frame(y = rtkuv(n = 1000, d = 1, m = m, s = sigma))
tdata <- transform(tdata, sy = sort(y))
fit <- vglm(y ~ -1, tikuv(d = 1), data = tdata, trace = TRUE)
coef(fit, matrix = TRUE)
(Cfit <- Coef(fit))
with(tdata, mean(y))
## Not run: with(tdata, hist(y, prob = TRUE))
lines(dtikuv(sy, d = 1, m = Cfit[1], s = Cfit[2]) ~ sy, data = tdata, col = "orange")
## End(Not run)
```

Tobit

The Tobit Distribution

Description

Density, distribution function, quantile function and random generation for the Tobit model.

Usage

```r
dtobit(x, mean = 0, sd = 1, Lower = 0, Upper = Inf, log = FALSE)
ptobit(q, mean = 0, sd = 1, Lower = 0, Upper = Inf,
      lower.tail = TRUE, log.p = FALSE)
qtobit(p, mean = 0, sd = 1, Lower = 0, Upper = Inf,
      lower.tail = TRUE, log.p = FALSE)
rtobit(n, mean = 0, sd = 1, Lower = 0, Upper = Inf)
```
Arguments

- \( x, q \) vector of quantiles.
- \( p \) vector of probabilities.
- \( n \) number of observations. If \( \text{length}(n) > 1 \) then the length is taken to be the number required.
- \( \text{Lower, Upper} \) vector of lower and upper thresholds.
- \( \text{mean, sd, lower.tail, log, log.p} \)
  
  see \texttt{rnorm}.

Details

See \texttt{tobit}, the VGAM family function for estimating the parameters, for details. Note that the density at \( \text{Lower} \) and \( \text{Upper} \) is the area to the left and right of those points. Thus there are two spikes (but less in value); see the example below. Consequently, \( \text{dtobit}(\text{Lower}) + \text{dtobit}(\text{Upper}) + \) the area in between equals unity.

Value

- \( \text{dtobit} \) gives the density, \( \text{ptobit} \) gives the distribution function, \( \text{qtobit} \) gives the quantile function, and \( \text{rtobit} \) generates random deviates.

Author(s)

T. W. Yee

See Also

- \texttt{tobit}, \texttt{rnorm}.

Examples

```r
mu <- 0.5; x <- seq(-2, 4, by = 0.01)
Lower <- -1; Upper <- 2.0

integrate(dtobit, lower = Lower, upper = Upper,
  mean = mu, Lower = Lower, Upper = Upper)$value +
 dtobit(Lower, mean = mu, Lower = Lower, Upper = Upper) +
 dtobit(Upper, mean = mu, Lower = Lower, Upper = Upper) # Adds to unity

# Not run:
plot(x, ptobit(x, m = mu, Lower = Lower, Upper = Upper),
  type = "l", ylim = 0:1, las = 1, col = "orange",
  ylab = paste("ptobit(m = ", mu, ", sd = 1, Lower =", Lower,
  ", Upper =", Upper, ")"),
  main = "Orange is cumulative distribution function; blue is density",
  sub = "Purple lines are the 10,20,...,90 percentiles")
abline(h = 0)
lines(x, dtobit(x, m = mu, Lower = Lower, Upper = Upper), col = "blue")```
tobit <- function(x, y, z, a, b, c, d) {
  x <- x + y + z + a + b + c + d
  x
}

Description

Fits a Tobit model.

Usage

tobit(Lower = 0, Upper = Inf, lmu = "identitylink", lsd = "log",
    nsimEIM = 250, imu = NULL, isd = NULL,
    type.fitted = c("uncensored", "censored", "mean.obs"),
    imethod = 1, zero = -2)

Arguments

- **Lower**: Numeric. It is the value $L$ described below. Any value of the linear model $x_i^T \beta$ that is less than this lowerbound is assigned this value. Hence this should be the smallest possible value in the response variable. May be a vector (see below for more information).

- **Upper**: Numeric. It is the value $U$ described below. Any value of the linear model $x_i^T \beta$ that is greater than this upperbound is assigned this value. Hence this should be the largest possible value in the response variable. May be a vector (see below for more information).

- **lmu, lsd**: Parameter link functions for the mean and standard deviation parameters. See `links` for more choices. The standard deviation is a positive quantity, therefore a log link is its default.

- **imu, isd**: See CommonVGAMffArguments for information.

- **type.fitted**: Type of fitted value returned. The first choice is default and is the ordinary uncensored or unbounded linear model. If "censored" then the fitted values in the interval $[L, U]$. If "mean.obs" then the mean of the observations is returned; this is a doubly truncated normal distribution augmented by point masses at the truncation points (see `dtobit`).
Initialization method. Either 1 or 2, this specifies two methods for obtaining initial values for the parameters.

nsimeim Used for the nonstandard Tobit model. See CommonVGAMffArguments for information.

zero An integer vector, containing the value 1 or 2. If so, the mean or standard deviation respectively are modelled as an intercept-only. Setting zero = NULL means both linear/additive predictors are modelled as functions of the explanatory variables. See CommonVGAMffArguments for information.

Details

The Tobit model can be written

$$y_i^* = x_i^T \beta + \varepsilon_i$$

where the $\varepsilon_i \sim N(0, \sigma^2)$ independently and $i = 1, \ldots, n$. However, we measure $y_i = y_i^*$ only if $y_i^* > L$ and $y_i^* < U$ for some cutpoints $L$ and $U$. Otherwise we let $y_i = L$ or $y_i = U$, whatever is closer. The Tobit model is thus a multiple linear regression but with censored responses if it is below or above certain cutpoints.

The defaults for lower and upper and lmu correspond to the standard Tobit model. Then Fisher scoring is used, else simulated Fisher scoring. By default, the mean $x_i^T \beta$ is the first linear/additive predictor, and the log of the standard deviation is the second linear/additive predictor. The Fisher information matrix for uncensored data is diagonal. The fitted values are the estimates of $x_i^T \beta$.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Warning

Convergence is often slow. Setting crit = "coeff" is recommended since premature convergence of the log-likelihood is common. Simulated Fisher scoring is implemented for the nonstandard Tobit model. For this, the working weight matrices for some observations are prone to not being positive-definite; if so then some checking of the final model is recommended and/or try inputting some initial values.

Note

The response can be a matrix. If so, then Lower and Upper are recycled into a matrix with the number of columns equal to the number of responses, and the recycling is done row-wise (byrow = TRUE). For example, these are returned in fit4@misc$Lower and fit4@misc$Upper below.

If there is no censoring then uninormal is recommended instead. Any value of the response less than Lower or greater than Upper will be assigned the value Lower and Upper respectively, and a warning will be issued. The fitted object has components censoredL and censoredU in the extra slot which specifies whether observations are censored in that direction. The function cens.normal is an alternative to tobit().
When obtaining initial values, if the algorithm would otherwise want to fit an underdetermined system of equations, then it uses the entire data set instead. This might result in rather poor quality initial values, and consequently, monitoring convergence is advised.

Author(s)

Thomas W. Yee

References


See Also

rtobit, cens.normal, uninormal, double.cens.normal, posnormal, rnorm.

Examples

```r
# Not run:
# Here, fit1 is a standard Tobit model and fit2 is a nonstandard Tobit model
tdata <- data.frame(x2 = seq(-1, 1, length = (nn <- 100)))
set.seed(1)
Lower <- 1; Upper <- 4  # For the nonstandard Tobit model
tdata <- transform(tdata,
  Lower.vec = rnorm(nn, Lower, 0.5),
  Upper.vec = rnorm(nn, Upper, 0.5))
meanfun1 <- function(x) 0 + 2*x
meanfun2 <- function(x) 2 + 2*x
meanfun3 <- function(x) 2 + 2*x
meanfun4 <- function(x) 3 + 2*x
tdata <- transform(tdata,
  y1 = rtobit(nn, mean = meanfun1(x2)),  # Standard Tobit model
  y2 = rtobit(nn, mean = meanfun2(x2), Lower = Lower, Upper = Upper),
  y3 = rtobit(nn, mean = meanfun3(x2), Lower = Lower.vec, Upper = Upper.vec),
  y4 = rtobit(nn, mean = meanfun3(x2), Lower = Lower.vec, Upper = Upper.vec))
with(tdata, table(y1 == 0))  # How many censored values?
with(tdata, table(y2 == Lower | y2 == Upper))  # How many censored values?
with(tdata, table(attr(y2, "cenL")))
with(tdata, table(attr(y2, "cenU")))

fit1 <- vglm(y1 ~ x2, tobit, data = tdata, trace = TRUE,
  crit = "coeff")  # crit = "coeff" is recommended
coef(fit1, matrix = TRUE)
summary(fit1)

fit2 <- vglm(y2 ~ x2, tobit(Lower = Lower, Upper = Upper, type.f = "cens"),
  data = tdata, crit = "coeff", trace = TRUE)  # ditto
table(fit2$extra$censoredL)
table(fit2$extra$censoredU)
coef(fit2, matrix = TRUE)
```
fit3 <- vglm(y3 ~ x2, 
  tobit(Lower = with(tdata, Lower.vec),
    Upper = with(tdata, Upper.vec), type.f = "cens"),
  data = tdata, crit = "coeff", trace = TRUE) # ditto
table(fit3@extra$censoredL)
table(fit3@extra$censoredU)
coef(fit3, matrix = TRUE)

# fit4 is fit3 but with type.fitted = "uncen".
fit4 <- vglm(cbind(y3, y4) ~ x2, 
  tobit(Lower = rep(with(tdata, Lower.vec), each = 2),
    Upper = rep(with(tdata, Upper.vec), each = 2)),
  data = tdata, crit = "coeff", trace = TRUE) # ditto
head(fit4@extra$censoredL) # A matrix
head(fit4@extra$censoredU) # A matrix
head(fit4@misc$Lower) # A matrix
head(fit4@misc$Upper) # A matrix
coef(fit4, matrix = TRUE)

## End(Not run)

## Not run: # Plot the results
par(mfrow = c(2, 2))
## Plot fit1
plot(y1 ~ x2, tdata, las = 1, main = "Standard Tobit model",
  col = as.numeric(attr(y1, "cenL")) + 3,
  pch = as.numeric(attr(y1, "cenL")) + 1)
legend(x = "topleft", leg = c("censored", "uncensored"),
  pch = c(2, 1), col = c("blue", "green"))
legend(-1.0, 2.5, c("Truth", "Estimate", "Naive"),
  col = c("purple", "orange", "black"), lwd = 2, lty = c(1, 2, 2))
lines(meanfun1(x2) ~ x2, tdata, col = "purple", lwd = 2)
lines(fitted(fit1) ~ x2, tdata, col = "orange", lwd = 2, lty = 2)
lines(fitted(lm(y1 ~ x2, tdata)) ~ x2, tdata, col = "black",
  lty = 2, lwd = 2) # This is simplest but wrong!

## Plot fit2
plot(y2 ~ x2, data = tdata, las = 1, main = "Tobit model",
  col = as.numeric(attr(y2, "cenL")) + 3 +
  as.numeric(attr(y2, "cenU")),
  pch = as.numeric(attr(y2, "cenL")) + 1 +
  as.numeric(attr(y2, "cenU")))
legend(x = "topleft", leg = c("censored", "uncensored"),
  pch = c(2, 1), col = c("blue", "green"))
legend(-1.0, 3.5, c("Truth", "Estimate", "Naive"),
  col = c("purple", "orange", "black"), lwd = 2, lty = c(1, 2, 2))
lines(meanfun2(x2) ~ x2, tdata, col = "purple", lwd = 2)
lines(fitted(fit2) ~ x2, tdata, col = "orange", lwd = 2, lty = 2)
lines(fitted(lm(y2 ~ x2, tdata)) ~ x2, tdata, col = "black",
  lty = 2, lwd = 2) # This is simplest but wrong!

## Plot fit3
plot(y3 ~ x2, data = tdata, las = 1,
main = "Tobit model with nonconstant censor levels",

col = as.numeric(attr(y3, "cenL")) + 3 +
    as.numeric(attr(y3, "cenU")),

pch = as.numeric(attr(y3, "cenL")) + 1 +
    as.numeric(attr(y3, "cenU")))

legend(x = "topleft", leg = c("censored", "uncensored"),
    pch = c(2, 1), col = c("blue", "green"))

legend(-1.0, 3.5, c("Truth", "Estimate", "Naive"),
    col = c("purple", "orange", "black"), lwd = 2, lty = c(1, 2, 2))

lines(meanfun3(x2) ~ x2, tdata, col = "purple", lwd = 2)
lines(fitted(fit3) ~ x2, tdata, col = "orange", lwd = 2, lty = 2)
lines(fitted(lm(y3 ~ x2, tdata)) ~ x2, tdata, col = "black",
    lty = 2, lwd = 2)  # This is simplest but wrong!

# Plot fit4
plot(y3 ~ x2, data = tdata, las = 1,
    main = "Tobit model with nonconstant censor levels",
    col = as.numeric(attr(y3, "cenL")) + 3 +
    as.numeric(attr(y3, "cenU")),

pch = as.numeric(attr(y3, "cenL")) + 1 +
    as.numeric(attr(y3, "cenU")))

legend(x = "topleft", leg = c("censored", "uncensored"),
    pch = c(2, 1), col = c("blue", "green"))

legend(-1.0, 3.5, c("Truth", "Estimate", "Naive"),
    col = c("purple", "orange", "black"), lwd = 2, lty = c(1, 2, 2))

lines(meanfun3(x2) ~ x2, data = tdata, col = "purple", lwd = 2)
lines(fitted(fit4)[, 1] ~ x2, tdata, col = "orange", lwd = 2, lty = 2)
lines(fitted(lm(y3 ~ x2, tdata)) ~ x2, data = tdata, col = "black",
    lty = 2, lwd = 2)  # This is simplest but wrong!

## End(Not run)

---

### Tol

#### Tolerances

**Description**

Generic function for the tolerances of a model.

**Usage**

Tol(object, ...)

**Arguments**

- **object**
  
  An object for which the computation or extraction of a tolerance or tolerances is meaningful.

- **...**
  
  Other arguments fed into the specific methods function of the model. Sometimes they are fed into the methods function for `coef`. 
Details

Different models can define an optimum in different ways. Many models have no such notion or definition.

Tolerances occur in quadratic ordination, i.e., CQO and UQO. They have ecological meaning because a high tolerance for a species means the species can survive over a large environmental range (stenoecous species), whereas a small tolerance means the species’ niche is small (eurycous species). Mathematically, the tolerance is like the variance of a normal distribution.

Value

The value returned depends specifically on the methods function invoked.

Warning

There is a direct inverse relationship between the scaling of the latent variables (site scores) and the tolerances. One normalization is for the latent variables to have unit variance. Another normalization is for all the tolerances to be unit. These two normalization cannot simultaneously hold in general. For rank-\(R>1\) models it becomes more complicated because the latent variables are also uncorrelated. An important argument when fitting quadratic ordination models is whether `eq.tolerances` is `true` or `false`. See Yee (2004) for details.

Note

Tolerances are undefined for ‘linear’ and additive ordination models. They are well-defined for quadratic ordination models.

Author(s)

Thomas W. Yee

References


See Also

`tol.qrrvglm`, `Max, Opt, cqo, rcim` for UQO.

Examples

```r
## Not run:
set.seed(111) # This leads to the global solution
hspider[,1:6] <- scale(hspider[, 1:6]) # Standardized environmental vars
p1 <- cqo(cbind(Aloacce, Alopcune, Alopfabr, Arctlude, Arctperi,
                Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull,
                Trocterr, Zoraspin) ~
               WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + Refllux,
```
Toxoplasmosis Data

Description
Toxoplasmosis data in 34 cities in El Salvador.

Usage
data(toxop)

Format
A data frame with 34 observations on the following 4 variables.

- `rainfall` a numeric vector; the amount of rainfall in each city.
- `ssize` a numeric vector; sample size.
- `cityNo` a numeric vector; the city number.
- `positive` a numeric vector; the number of subjects testing positive for the disease.

Details
See the references for details.

Source
See the references for details.

References


See Also
double.expbinomial.

Examples

```r
## Not run: with(toxop, plot(rainfall, positive / ssize, col = "blue"))
plot(toxop, col = "blue")
## End(Not run)
```
Description

Density, distribution function, quantile function and random generation for the Triangle distribution with parameter theta.

Usage

dtriangle(x, theta, lower = 0, upper = 1, log = FALSE)
ptriangle(q, theta, lower = 0, upper = 1, lower.tail = TRUE, log.p = FALSE)
qtriangle(p, theta, lower = 0, upper = 1, lower.tail = TRUE, log.p = FALSE)
rtriangle(n, theta, lower = 0, upper = 1)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. Same as runif.
theta the theta parameter which lies between lower and upper.
lower, upper lower and upper limits of the distribution. Must be finite.
log Logical. If log = TRUE then the logarithm of the density is returned.
lower.tail, log.p Same meaning as in pnorm or qnorm.

Details

See triangle, the VGAM family function for estimating the parameter \( \theta \) by maximum likelihood estimation.

Value

dtriangle gives the density, ptriangle gives the distribution function, qtriangle gives the quantile function, and rtriangle generates random deviates.

Author(s)

T. W. Yee and Kai Huang

See Also

triangle.
Examples

```r
## Not run:  x <- seq(-0.1, 1.1, by = 0.01); theta <- 0.75
plot(x, dtriangle(x, theta = theta), type = "l", col = "blue", las = 1,
     main = "Blue is density, orange is cumulative distribution function",
     sub = "Purple lines are the 10, 20, ..., 90 percentiles",
     ylim = c(0,2), ylab = "")
abline(h = 0, col = "blue", lty = 2)
lines(x, ptriangle(x, theta = theta), col = "orange")
probs <- seq(0.1, 0.9, by = 0.1)
Q <- qtriangle(probs, theta = theta)
lines(Q, dtriangle(Q, theta = theta), col = "purple", lty = 3, type = "h")
ptriangle(Q, theta = theta) - probs  # Should be all zero
abline(h = probs, col = "purple", lty = 3)
## End(Not run)
```

---

triangle  

Triangle Distribution Family Function

Description

Estimating the parameter of the triangle distribution by maximum likelihood estimation.

Usage

```r
triangle(lower = 0, upper = 1,
         link = extlogit(min = 0, max = 1), itheta = NULL)
```

Arguments

- `lower, upper`: lower and upper limits of the distribution. Must be finite. Called $A$ and $B$ respectively below.
- `link`: Parameter link function applied to the parameter $\theta$, which lies in $(A, B)$. See Links for more choices. The default constrains the estimate to lie in the interval.
- `itheta`: Optional initial value for the parameter. The default is to compute the value internally.

Details

The triangle distribution has a probability density function that consists of two lines joined at $\theta$, which is the location of the mode. The lines intersect the $y = 0$ axis at $A$ and $B$. Here, Fisher scoring is used.

On fitting, the extra slot has components called `lower` and `upper` which contains the values of the above arguments (recycled to the right length). The fitted values are the mean of the distribution, which is $(A + B + \theta)/3$. 
Value

An object of class "vglmff" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm} and \texttt{vgam}.

Warning

The MLE regularity conditions do not seem to hold for this distribution so that misleading inferences may result, e.g., in the summary and \texttt{vcov} of the object. Additionally, convergence to the MLE often appears to fail.

Note

The response must contain values in \((A, B)\). For most data sets (especially small ones) it is very common for half-stepping to occur. Arguments \texttt{lower} and \texttt{upper} and \texttt{link} must match. For example, setting \texttt{lower = 0.2} and \texttt{upper = 4} and \texttt{link = extlogit(min = 0.2, max = 4.1)} will result in an error. Ideally \texttt{link = extlogit(min = lower, max = upper)} ought to work but it does not (yet)! Minimal error checking is done for this deficiency.

Author(s)

T. W. Yee

References


See Also

\texttt{Triangle, simulate.vlm}.

Examples

```r
# Example 1
tdata <- data.frame(y = rtriangle(n = 3000, theta = 3/4))
fit <- vglm(y ~ 1, triangle(link = "identitylink"), data = tdata, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
head(fit@extra$lower)
head(fitted(fit))
with(tdata, mean(y))

# Example 2; Kotz and van Dorp (2004), p.14
rdata <- data.frame(y = c(0.1, 0.25, 0.3, 0.4, 0.45, 0.6, 0.75, 0.8))
fit <- vglm(y ~ 1, triangle(link = "identitylink"), data = rdata, trace = TRUE, 
crit = "coef", maxit = 1000)
Coef(fit) # The MLE is the 3rd order statistic, which is 0.3.
fit <- vglm(y ~ 1, triangle(link = "identitylink"), data = rdata, trace = TRUE, 
crit = "coef", maxit = 1001)
Coef(fit) # The MLE is the 3rd order statistic, which is 0.3.
```
Description

Generic function for a trajectory plot.

Usage

\texttt{trplot(object, ...)}

Arguments

- \texttt{object}: An object for which a trajectory plot is meaningful.
- \texttt{...}: Other arguments fed into the specific methods function of the model. They usually are graphical parameters, and sometimes they are fed into the methods function for \texttt{coef}.

Details

Trajectory plots can be defined in different ways for different models. Many models have no such notion or definition.

For quadratic and additive ordination models they plot the fitted values of two species against each other (more than two is theoretically possible, but not implemented in this software yet).

Value

The value returned depends specifically on the methods function invoked.

Author(s)

Thomas W. Yee

References


See Also

\texttt{trplot.qrrvglm}, \texttt{perspqrqrrvglm}, \texttt{lvplot}. 
Examples

```r
## Not run: set.seed(123)
hs = hspider[, 1:6] <- scale(hspider[, 1:6])  # Standardized environmental vars
plc = cbind(Alophaceae, Alopecurus, Arctium, Arctium, Arctium, Arctium,
            Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull, Trocterr, Zoraspin) ~
            WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + RefLux,
quasipoissonff, data = hspider, CrowPositive = FALSE)
nos = ncol(depvar(plc))
cr < - 1:nos  # OR (1:cols)[1:7] to omit yellow
trplot(plc[, which.species == 1:3], log = "xy",
col = c("blue", "orange", "green"), lwd = 2, label = TRUE) -> ii
legend(0.00005, 0.3, paste(ii$Species[1], ii$Species[2], sep = " and "),
lwd = 2, lty = 1, col = c("blue", "orange", "green"))
abline(a = 0, b = 1, lty = "dashed", col = "grey")
## End(Not run)
```

Description

Produces a trajectory plot for quadratic reduced-rank vector generalized linear models (QRR-VGLMs). It is only applicable for rank-1 models with argument norrr = ~ 1.

Usage

```r
trplot.qrrvglm(object, which.species = NULL, add = FALSE, show.plot = TRUE,
               label.sites = FALSE, sitenames = rownames(object@y),
               axes.equal = TRUE, cex = par$ciex,
               col = 1:nos * (nos - 1)/2, log = "",
lty = rep(par$lty, length.out = nos * (nos - 1)/2),
lwd = rep(par$lwd, length.out = nos * (nos - 1)/2),
tcol = rep(par$tcol, length.out = nos * (nos - 1)/2),
xlab = NULL, ylab = NULL,
main = "", type = "b", check.ok = TRUE, ...)
```

Arguments

- `object`: Object of class "qrrvglm", i.e., a CQO object.
- `which.species`: Integer or character vector specifying the species to be plotted. If integer, these are the columns of the response matrix. If character, these must match exactly with the species' names. The default is to use all species.
- `add`: Logical. Add to an existing plot? If FALSE (default), a new plot is made.
- `show.plot`: Logical. Plot it?
label.sites Logical. If TRUE, the points on the curves/trajectories are labelled with the sitenames.
sitenames Character vector. The names of the sites.
axes.equal Logical. If TRUE, the x- and y-axes will be on the same scale.
cex Character expansion of the labelling of the site names. Used only if label.sites is TRUE. See the cex argument in par.
col Color of the lines. See the col argument in par. Here, n is the number of species.
log Character, specifying which (if any) of the x- and y-axes are to be on a logarithmic scale. See the log argument in par.
lty Line type. See the lty argument of par.
lwd Line width. See the lwd argument of par.
tcol Color of the text for the site names. See the col argument in par. Used only if label.sites is TRUE.
xlab Character caption for the x-axis. By default, a suitable caption is found. See the xlab argument in plot or title.
ylab Character caption for the y-axis. By default, a suitable caption is found. See the xlab argument in plot or title.
main Character, giving the title of the plot. See the main argument in plot or title.
type Character, giving the type of plot. A common option is to use type="l" for lines only. See the type argument of plot.
check.ok Logical. Whether a check is performed to see that noR ~ Q was used. It doesn't make sense to have a trace plot unless this is so.
... Arguments passed into the plot function when setting up the entire plot. Useful arguments here include xlim and ylim.

Details

A trajectory plot plots the fitted values of a ‘second’ species against a ‘first’ species. The argument which.species must therefore contain at least two species. By default, all of the species that were fitted in object are plotted. With more than a few species the resulting plot will be very congested, and so it is recommended that only a few species be selected for plotting.

In the above, M is the number of species selected for plotting, so there will be M(M − 1)/2 curves/trajectories in total.

A trajectory plot will be fitted only if noR ~ 1 because otherwise the trajectory will not be a smooth function of the latent variables.

Value

A list with the following components.
species.names A matrix of characters giving the ‘first’ and ‘second’ species. The number of different combinations of species is given by the number of rows. This is useful for creating a legend.
sitenames A character vector of site names, sorted by the latent variable (from low to high).
Note

Plotting the axes on a log scale is often a good idea. The use of xlim and ylim to control the axis limits is also a good idea, so as to limit the extent of the curves at low abundances or probabilities. Setting label.sites = TRUE is a good idea only if the number of sites is small, otherwise there is too much clutter.

Author(s)

Thomas W. Yee

References


See Also
cqo, par, title.

Examples

```r
## Not run: set.seed(111) # This leads to the global solution
# hspider[,1:6] <- scale(hspider[,1:6]) # Standardize the environmental variables
p1 <- cqo(cbind(Alopacce, Alopcurve, Alopfuel, Arctlute, Arctperi, Auloalbi, Pardlugu, Pardmont, Pardnigr, Pardpull, Trocterr, Zoraspin) ~
WaterCon + BareSand + FallTwig + CoveMoss + CoveHerb + Reflux,
poissonff, data = hspider, trace = FALSE)

trplot(p1, which.species = 1:3, log = "xy", type = "b", lty = 1,
main = "Trajectory plot of three hunting spiders species",
col = c("blue", "red", "green"), lwd = 2, label = TRUE) -> ii
legend(0.00005, 0.3, lwd = 2, lty = 1, col = c("blue", "red", "green"),
with(ii, paste(species.names[1], species.names[2], sep = " and ")))
abline(a = 0, b = 1, lty = "dashed", col = "grey") # Useful reference line

## End(Not run)
```

---

### Truncpareto

*The Truncated Pareto Distribution*

Description

Density, distribution function, quantile function and random generation for the upper truncated Pareto(1) distribution with parameters lower, upper and shape.
Usage

dtruncpareto(x, lower, upper, shape, log = FALSE)
ptruncpareto(q, lower, upper, shape, lower.tail = TRUE, log.p = FALSE)
qtruncpareto(p, lower, upper, shape)
rtruncpareto(n, lower, upper, shape)

Arguments

x, q  vector of quantiles.
p  vector of probabilities.
n, log  Same meaning as runif.
lower, upper, shape  the lower, upper and shape (k) parameters. If necessary, values are recycled.
lower.tail, log.p  Same meaning as in pnorm or qnorm.

Details

See truncpareto, the VGAM family function for estimating the parameter k by maximum likelihood estimation, for the formula of the probability density function and the range restrictions imposed on the parameters.

Value

dtruncpareto gives the density, ptruncpareto gives the distribution function, qtruncpareto gives the quantile function, and rtruncpareto generates random deviates.

Author(s)

T. W. Yee and Kai Huang

References


See Also

truncpareto.

Examples

lower <- 3; upper <- 8; kay <- exp(0.5)
## Not run:  xx <- seq(lower - 0.5, upper + 0.5, len = 401)
plot(xx, dtruncpareto(xx, low = lower, upp = upper, shape = kay),
main = "Truncated Pareto density split into 10 equal areas",
type = "l", ylim = 0:1, xlab = "x")
abline(h = 0, col = "blue", lty = 2)
qq <- qtruncpareto(seq(0.1, 0.9, by = 0.1), low = lower, upp = upper,
shape = kay)
lines(qq, dtruncpareto(qq, low = lower, upp = upper, shape = kay),
col = "purple", lty = 3, type = "h")
lines(xx, ptruncpareto(xx, low = lower, upp = upper, shape = kay),
col = "orange")

## End(not run)
pp <- seq(0.1, 0.9, by = 0.1)
qq <- qtruncpareto(pp, lower = lower, upper = upper, shape = kay)
ptruncpareto(qq, lower = lower, upper = upper, shape = kay)
qtruncpareto(ptruncpareto(qq, lower = lower, upper = upper, shape = kay),
   lower = lower, upper = upper, shape = kay) - qq  # Should be all 0

truncweibull

Truncated Weibull Distribution Family Function

Description

Maximum likelihood estimation of the 2-parameter Weibull distribution with lower truncation. No observations should be censored.

Usage

truncweibull(lower.limit = 1e-5,
   lAlpha = "loge", lBetaa = "loge",
   iAlpha = NULL, iBetaa = NULL,
   nrfs = 1, probs.y = c(0.2, 0.5, 0.8),
   imethod = 1, zero = -2)

Arguments

lower.limit Positive lower truncation limits. Recycled to the same dimension as the response, going across rows first. The default, being close to 0, should mean effectively the same results as weibullR if there are no response values that are smaller.

lAlpha, lBetaa Parameter link functions applied to the (positive) parameters Alpha (called $\alpha$ below) and (positive) Betaa (called $\beta$ below). See Links for more choices.

iAlpha, iBetaa See CommonVGAMffArguments.

imethod, nrfs, zero, probs.y Details at weibullR.

Details

MLE of the two parameters of the Weibull distribution are computed, subject to lower truncation. That is, all response values are greater than lower.limit, element-wise. For a particular observation this is any known positive value. This function is currently based directly on Wingo (1989) and his parameterization is used (it differs from weibullR.) In particular, $\beta = a$ and $\alpha = (1/b)^a$ where $a$ and $b$ are as in weibullR and dweibull.
Upon fitting the extra slot has a component called `lower.limit` which is of the same dimension as the response. The fitted values are the mean, which are computed using `pgamma.deriv` and `pgamma.deriv.unscaled`.

**Value**

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

**Warning**

This function may be converted to the same parameterization as `weibullR` at any time. Yet to do: one element of the EIM may be wrong (due to two interpretations of a formula; but it seems to work). Convergence is slower than usual and this may imply something is wrong; use argument `maxit`. In fact, it’s probably because `pgamma.deriv.unscaled` is inaccurate at $q = 1$ and $q = 2$. Also, convergence should be monitored, especially if the truncation means that a large proportion of the data is lost compared to an ordinary Weibull distribution.

**Note**

More improvements need to be made, e.g., initial values are currently based on no truncation. This `VGAM` family function handles multiple responses.

**Author(s)**

T. W. Yee

**References**


**See Also**

`weibullR`, `dweibull`, `pgamma.deriv`, `pgamma.deriv.unscaled`.

**Examples**

```r
nn <- 5000; prop.lost <- 0.40  # Proportion lost to truncation
wdata <- data.frame(x = runif(nn))  # Complete Weibull data
wdata <- transform(wdata,
  Betaa = exp(1))  # > 2 is okay (satisfies regularity conds)
wdata <- transform(wdata, Alpha = exp(0.5 - 1 * x))
wdata <- transform(wdata, Shape = Betaa,
  #
  aaa = Betaa,
  #
  bbb = 1 / Alpha*(1 / Betaa),
  Scale = 1 / Alpha*(1 / Betaa))
wdata <- transform(wdata, y2 = rweibull(nn, shape = Shape, scale = Scale))
summary(wdata)
lower.limit2 <- with(wdata, quantile(y2, prob = prop.lost))  # Proportion lost
```

University California Berkeley Graduate Admissions

**Description**

University California Berkeley Graduate Admissions: counts cross-classified by acceptance/rejection and gender, for the six largest departments.

**Usage**

data(ucberk)

**Format**

A data frame with 6 departmental groups with the following 5 columns.

- **m.deny** Counts of men denied admission.
- **m.admit** Counts of men admitted.
- **w.deny** Counts of women denied admission.
- **w.admit** Counts of women admitted.
- **dept** Department (the six largest), called A, codeB, ..., codeF.

**Details**

From Bickel et al. (1975), the data consists of applications for admission to graduate study at the University of California, Berkeley, for the fall 1973 quarter. In the admissions cycle for that quarter, the Graduate Division at Berkeley received approximately 15,000 applications, some of which were later withdrawn or transferred to a different proposed entry quarter by the applicants. Of the applications finally remaining for the fall 1973 cycle 12,763 were sufficiently complete to permit a decision. There were about 101 graduate department and interdepartmental graduate majors. There were 8442 male applicants and 4321 female applicants. About 44 percent of the males and about 35 percent of the females were admitted. The data are well-known for illustrating Simpson's paradox.

**References**


uninormal

Examples

summary(ucberk)

<table>
<thead>
<tr>
<th>uninormal</th>
<th>Univariate Normal Distribution</th>
</tr>
</thead>
</table>

Description

Maximum likelihood estimation of the two parameters of a univariate normal distribution.

Usage

```r
uninormal(lmean = "identitylink", lsd = "loge", lvar = "loge",
          var.arg = FALSE, imethod = 1, isd = NULL, parallel = FALSE,
          smallno = 1e-05, zero = -2)
```

Arguments

- `lmean, lsd, lvar` Link functions applied to the mean and standard deviation/variance. See `Links` for more choices. Being positive quantities, a log link is the default for the standard deviation and variance (see `var.arg`).
- `var.arg` Logical. If `TRUE` then the second parameter is the variance and `lsd` and `esd` are ignored, else the standard deviation is used and `lvar` and `evar` are ignored.
- `smallno` Numeric, positive but close to 0. Used specifically for quasi-variances; if the link for the mean is `explink` then any non-positive value of `eta` is replaced by this quantity (hopefully, temporarily and only during early iterations).
- `imethod, parallel, isd, zero` See `CommonVGAMffArguments` for more information. If `lmean = loge` then try `imethod = 2`. If `parallel = TRUE` then the parallelism constraint is not applied to the intercept.

Details

This fits a linear model (LM) as the first linear/additive predictor. So, by default, this is just the mean. By default, the log of the standard deviation is the second linear/additive predictor. The Fisher information matrix is diagonal. This `VGAM` family function can handle multiple responses.

Value

An object of class `"vglmff"` (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Warning

`uninormal()` is the new name; `normalq()` is old and will be decommissioned soon.
Note

Yet to do: allow an argument such as eq. sd that enables the standard deviations to be the same.

Author(s)

T. W. Yee

References


See Also
gaussianff, posnormal, mix2normal, normal.vcm, Qvar, tobit, cens.normal, foldnormal, skewnormal, double.cens.normal, SURff, AR1, huber2, studentt, binormal, dnorm, simulate.vlm.

Examples

```r
udata <- data.frame(x2 = rnorm(nn <- 200))
udata <- transform(udata,
  y1 = rnorm(nn, m = 1 - 3*x2, sd = exp(1 + 0.2*x2)),
  y2a = rnorm(nn, m = 1 + 2*x2, sd = exp(1 + 2.0*x2)^0.5),
  y2b = rnorm(nn, m = 1 + 2*x2, sd = exp(1 + 2.0*x2)^0.5))
fit1 <- vglm(y1 ~ x2, uninormal(zero = NULL), data = udata, trace = TRUE)
coef(fit1, matrix = TRUE)
fit2 <- vglm(cbind(y2a, y2b) ~ x2, data = udata, trace = TRUE,
  uninormal(var = TRUE, parallel = TRUE ~ x2,
  zero = NULL))
coef(fit2, matrix = TRUE)

# Generate data from N(mu = theta = 10, sigma = theta) and estimate theta.
theta <- 10
udata <- data.frame(y3 = rnorm(100, m = theta, sd = theta))
fit3a <- vglm(y3 ~ 1, uninormal(lsd = "identitylink"), data = udata,
  constraints = list("Intercept" = rbind(1, 1)))
fit3b <- vglm(y3 ~ 1, uninormal(lsd = "identitylink", parallel = TRUE ~ 1,
  zero = NULL), data = udata)
coef(fit3a, matrix = TRUE)
coef(fit3b, matrix = TRUE) # Same as fit3a
```

**V1 Flying-Bombs Hits in London**

Description

A small count data set. During WWII V1 flying-bombs were fired from sites in France (Pas-de-Calais) and Dutch coasts towards London. The number of hits per square grid around London were recorded.
Usage

```r
data(V1)
```

Format

A data frame with the following variables.

- **hits** Values between 0 and 4, and 7. Actually, the 7 is really imputed from the paper (it was recorded as "5 and over").
- **ofreq** Observed frequency, i.e., the number of grids with that many hits.

Details

The data concerns 576 square grids each of 0.25 square kms about south London. The area was selected comprising 144 square kms over which the basic probability function of the distribution was very nearly constant. V1s, which were one type of flying-bomb, were a “Vergeltungswaffen” or vengeance weapon fired during the summer of 1944 at London. The V1s were informally called Buzz Bombs or Doodlebugs, and they were pulse-jet-powered with a warhead of 850 kg of explosives. Over 9500 were launched at London, and many were shot down by artillery and the RAF. Over the period considered the total number of bombs within the area was 537.

It was asserted that the bombs tended to be grouped in clusters. However, a basic Poisson analysis shows this is not the case. Their guidance system being rather primitive, the data is consistent with a Poisson distribution (random).

Source


References


See Also

- poissonff.

Examples

```r
V1
mean(with(V1, rep(hits, times = ofreq)))
var(with(V1, rep(hits, times = ofreq)))
sum(with(V1, rep(hits, times = ofreq)))
```

```r
## Not run: barplot(with(V1, ofreq),
## names.arg = as.character(with(V1, hits)),
## main = "London V1 buzz bomb hits",
## col = "lightblue", las = 1,
## ylab = "Frequency", xlab = "Hits")
## End(Not run)
```
Venice Maximum Sea Levels Data

Description

Some sea levels data sets recorded at Venice, Italy.

Usage

data(venice)
data(venice90)

Format

venice is a data frame with 51 observations on the following 11 variables. It concerns the maximum heights of sea levels between 1931 and 1981.

- **year** a numeric vector.
- **r1, r2, r3, r4, r5, r6, r7, r8, r9, r10** numeric vectors; r1 is the highest recorded value, r2 is the second highest recorded value, etc.

venice90 is a data frame with 455 observations on the following 7 variables.

- **year, month, day, hour** numeric vectors; actual time of the recording.
- **sealevel** numeric; sea level.
- **ohour** numeric; number of hours since the midnight of 31 Dec 1939 and 1 Jan 1940.
- **Year** numeric vector; approximate year as a real number. The formula is start.year + ohour / (365.26 * 24) where start.year is 1940. One can treat Year as continuous whereas year can be treated as both continuous and discrete.

Details

Sea levels are in cm. For venice90, the value 0 corresponds to a fixed reference point (e.g., the mean sea level in 1897 at an old palace of Venice). Clearly since the relative (perceived) mean sea level has been increasing in trend over time (more than an overall 0.4 m increase by 2010), therefore the value 0 is (now) a very low and unusual measurement.

For venice, in 1935 only the top six values were recorded.

For venice90, this is a subset of a data set provided by Paolo Pirazzoli consisting of hourly sea levels from 1940 to 2009. Values greater than 90 cm were extracted, and then declustered (each cluster provides no more than one value, and each value is at least 24 hours apart). Thus the values are more likely to be independent. Of the original (2009-1940+1)*365.26*24 values about 7 percent of these comprise venice90.

Yet to do: check for consistency between the data sets. Some external data sets elsewhere have some extremes recorded at times not exactly on the hour.
Source
Thanks to Paolo Pirazzoli and Alberto Tomasin for the venice90 data.

References

See Also
guplot, gev, gpd.

Examples
```r
## Not run:
matplot(venice["year"], venice[, -1], xlab = "Year",
ylab = "Sea level (cm)", type = "l")

ymat <- as.matrix(venice[, paste("r", 1:10, sep = "")])
fit1 <- vgam(ymat ~ s(year, df = 3), gumbel(R = 365, mpv = TRUE),
  data = venice, trace = TRUE, na.action = na.pass)
head(fitted(fit1))

par(mfrow = c(2, 1), xpd = TRUE)
plot(fit1, se = TRUE, lcol = "blue", llwd = 2, slty = "dashed")

par(mfrow = c(1,1), bty = "l", xpd = TRUE, las = 1)
qtplot(fit1, mpv = TRUE, lcol = c(1, 2, 5), tcol = c(1, 2, 5),
  llwd = 2, pcol = "blue", tadj = 0.1)

plot(sealevel ~ Year, data = venice90, type = "h", col = "blue")
summary(venice90)
dim(venice90)
round(100 * nrow(venice90) / ((2009 - 1940 + 1) * 365.26 * 24), digits = 3)
```

vgam  

**Fitting Vector Generalized Additive Models**

**Description**

Fit a vector generalized additive model (VGAM). This is a large class of models that includes generalized additive models (GAMs) and vector generalized linear models (VGLMs) as special cases.
Usage

vgam(formula, family, data = list(), weights = NULL, subset = NULL,
na.action = na.fail, etastart = NULL, mustart = NULL,
coefstart = NULL, control = vgam.control(...), offset = NULL,
method = "vgam.fit", model = FALSE, x.arg = TRUE, y.arg = TRUE,
contrasts = NULL, constraints = NULL,
extra = list(), form2 = NULL, qr.arg = FALSE, smart = TRUE, ...)  

Arguments

formula a symbolic description of the model to be fit. The RHS of the formula is applied to each linear/additive predictor, and usually includes at least one s term. Different variables in each linear/additive predictor can be chosen by specifying constraint matrices.

family Same as for vglm.

data an optional data frame containing the variables in the model. By default the variables are taken from environment(formula), typically the environment from which vgam is called.

weights, subset, na.action Same as for vglm. Note that subset may be unreliable and to get around this problem it is best to use subset to create a new smaller data frame and feed in the smaller data frame. See below for an example. This is a bug that needs fixing.

etastart, mustart, coefstart Same as for vglm.

control a list of parameters for controlling the fitting process. See vgam.control for details.

method the method to be used in fitting the model. The default (and presently only) method vgam.fit uses iteratively reweighted least squares (IRLS).

constraints, model, offset Same as for vglm.

x.arg, y.arg logical values indicating whether the model matrix and response vector/matrix used in the fitting process should be assigned in the x and y slots. Note the model matrix is the LM model matrix; to get the VGAM model matrix type model.matrix(vgamfit) where vgamfit is a vgam object.

contrasts, extra, form2, qr.arg, smart Same as for vglm.

... further arguments passed into vgam.control.

Details

A vector generalized additive model (VGAM) is loosely defined as a statistical model that is a function of \( M \) additive predictors. The central formula is given by

\[
\eta_j = \sum_{k=1}^{p} f_{(j)k}(x_k)
\]
where $x_k$ is the $k$th explanatory variable (almost always $x_1 = 1$ for the intercept term), and $f(j)k$ are smooth functions of $x_k$ that are estimated by smoothers. The first term in the summation is just the intercept. Currently only one type of smoother is implemented and this is called a vector (cubic smoothing spline) smoother. Here, $j = 1, \ldots, M$ where $M$ is finite. If all the functions are constrained to be linear then the resulting model is a vector generalized linear model (VGLM). VGLMs are best fitted with vglm.

Vector (cubic smoothing spline) smoothers are represented by $s()$ (see $s$). Local regression via $lo()$ is not supported. The results of vgam will differ from the gam() (in the gam) because vgam() uses a different knot selection algorithm. In general, fewer knots are chosen because the computation becomes expensive when the number of additive predictors $M$ is large.

The underlying algorithm of VGAMs is iteratively reweighted least squares (IRLS) and modified vector backfitting using vector splines. B-splines are used as the basis functions for the vector (smoothing) splines. vgam.fit() is the function that actually does the work. The smoothing code is based on F. O’Sullivan’s BART code.

A closely related methodology based on VGAMs called constrained additive ordination (CAO) first forms a linear combination of the explanatory variables (called latent variables) and then fits a GAM to these. This is implemented in the function cao for a very limited choice of family functions.

Value

An object of class "vgam" (see vgam-class for further information).

WARNING

Currently vgam can only handle constraint matrices cmat, say, such that crossprod(cmat) is diagonal. This is a bug that I will try to fix up soon; see is.buggy.

See warnings in vglm.control.

Note

This function can fit a wide variety of statistical models. Some of these are harder to fit than others because of inherent numerical difficulties associated with some of them. Successful model fitting benefits from cumulative experience. Varying the values of arguments in the VGAM family function itself is a good first step if difficulties arise, especially if initial values can be inputted. A second, more general step, is to vary the values of arguments in vgam.control. A third step is to make use of arguments such as etastart, coefstart and mustart.

Some VGAM family functions end in "ff" to avoid interference with other functions, e.g., binomialff, poissonff, gaussianff, gammaff. This is because VGAM family functions are incompatible with glm (and also gam in the gam library and gam in the mgcv library).

The smart prediction (smartpred) library is packed with the VGAM library.

The theory behind the scaling parameter is currently being made more rigorous, but it should give the same value as the scale parameter for GLMs.

Author(s)

Thomas W. Yee
References


See Also

`is.buggy`, `vgam.control`, `vgam-class`, `vglmff-class`, `plotvgam`, `vglm`, `s`, `vsmooth.spline`, `cao`.

Examples

# Nonparametric proportional odds model
pneumo <- transform(pneumo, let = log(exposure.time))
vgam(cbind(normal, mild, severe) ~ s(let),
     cumulative(parallel = TRUE), data = pneumo, trace = TRUE)

# Nonparametric logistic regression
fit <- vgam(agaaua ~ s(altitude, df = 2), binomialff, data = hunua)
## Not run: plot(fit, se = TRUE)
pfit <- predict(fit, type = "terms", raw = TRUE, se = TRUE)
names(pfit)
head(pfit$fitted)
head(pfit$se.fit)
pfit$df
pfit$sigma

# Fit two species simultaneously
fit2 <- vgam(cbind(agaaua, kniexc) ~ s(altitude, df = c(2, 3)),
             binomialff(multiple.responses = TRUE), data = hunua)
coef(fit2, matrix = TRUE) # Not really interpretable
## Not run: plot(fit2, se = TRUE, overlay = TRUE, lcol = 1:2, scol = 1:2)

ooo <- with(hunua, order(altitude))
with(hunua, matplot(altitude[ooo], fitted(fit2)[ooo[, ylim = c(0, 0.8),
    xlab = "Altitude (m)", ylab = "Probability of presence", las = 1,
    main = "Two plant species' response curves", type = "l", lwd = 2))
with(hunua, rug(altitude))
## End(Not run)

# The subset argument does not work here. Use subset() instead.
set.seed(1)
zdata <- data.frame(x2 = runif(nn <- 100))
zdata <- transform(zdata, y = rbinom(nn, 1, 0.5))
zdata <- transform(zdata, subS = runif(nn) < 0.7)
sub.zdata <- subset(zdata, subS) # Use this instead
if (FALSE)
  fit4a <- vgam(cbind(y, y) ~ s(x2, df = 2),
                binomialff(multiple.responses = TRUE),
                data = zdata, subset = subS) # This fails!!!
fit4b <- vgam(cbind(y, y) ~ s(x2, df = 2),
                data = zdata, subset = subS)
Description

Vector generalized additive models.

Objects from the Class

Objects can be created by calls of the form vgam(...).

Slots

nl.chisq: Object of class "numeric". Nonlinear chi-squared values.
nl.df: Object of class "numeric". Nonlinear chi-squared degrees of freedom values.
spar: Object of class "numeric" containing the (scaled) smoothing parameters.
s.xargument: Object of class "character" holding the variable name of any s() terms.
var: Object of class "matrix" holding approximate pointwise standard error information.
b spline: Object of class "list" holding the scaled (internal and boundary) knots, and the fitted B-spline coefficients. These are used for prediction.
ex tra: Object of class "list"; the extra argument on entry to vglm. This contains any extra information that might be needed by the family function.
family: Object of class "vglmff". The family function.
iter: Object of class "numeric". The number of IRLS iterations used.
predictors: Object of class "matrix" with M columns which holds the M linear predictors.
assign: Object of class "list", from class "vlm". This named list gives information matching the columns and the (LM) model matrix terms.
call: Object of class "call", from class "vlm". The matched call.
coefficients: Object of class "numeric", from class "vlm". A named vector of coefficients.
constraints: Object of class "list", from class "vlm". A named list of constraint matrices used in the fitting.
contrasts: Object of class "list", from class "vlm". The contrasts used (if any).
control: Object of class "list", from class "vlm". A list of parameters for controlling the fitting process. See vglm.control for details.
criterion: Object of class "list", from class "vlm". List of convergence criterion evaluated at the final IRLS iteration.
df.residual: Object of class "numeric", from class "vlm". The residual degrees of freedom.
df.total: Object of class "numeric", from class "vlm". The total degrees of freedom.
dispersion: Object of class "numeric", from class "vlm". The scaling parameter.
effects: Object of class "numeric", from class "vlm". The effects.
fitted.values: Object of class "matrix", from class "vlm". The fitted values. This is usually
the mean but may be quantiles, or the location parameter, e.g., in the Cauchy model.
misc: Object of class "list", from class "vlm". A named list to hold miscellaneous parameters.
model: Object of class "data.frame", from class "vlm". The model frame.
n.a. action: Object of class "list", from class "vlm". A list holding information about missing
values.
offset: Object of class "matrix", from class "vlm". If non-zero, a \(M\)-column matrix of offsets.
post: Object of class "list", from class "vlm" where post-analysis results may be put.
preplot: Object of class "list", from class "vlm" used by plotvgam; the plotting parameters
may be put here.
prior.weights: Object of class "matrix", from class "vlm" holding the initially supplied weights.
qr: Object of class "list", from class "vlm". QR decomposition at the final iteration.
R: Object of class "matrix", from class "vlm". The \(R\) matrix in the QR decomposition used in
the fitting.
rank: Object of class "integer", from class "vlm". Numerical rank of the fitted model.
residuals: Object of class "matrix", from class "vlm". The \textit{working} residuals at the final IRLS
iteration.
ResSS: Object of class "numeric", from class "vlm". Residual sum of squares at the final IRLS
iteration with the adjusted dependent vectors and weight matrices.
smart.prediction: Object of class "list", from class "vlm". A list of data-dependent parameters
(if any) that are used by smart prediction.
terms: Object of class "list", from class "vlm". The \textit{terms} object used.
weights: Object of class "matrix", from class "vlm". The weight matrices at the final IRLS
iteration. This is in matrix-band form.
x: Object of class "matrix", from class "vlm". The model matrix (LM, not VGLM).
xlevels: Object of class "list", from class "vlm". The levels of the factors, if any, used in
fitting.
y: Object of class "matrix", from class "vlm". The response, in matrix form.
xm2: Object of class "matrix", from class "vlm". See \texttt{vglm-class}).
ynom2: Object of class "matrix", from class "vlm". See \texttt{vglm-class}).
callXm2: Object of class "call", from class "vlm". The matched call for argument \texttt{form2}.

Extends

Class "vglm", directly. Class "vlm", by class "vglm".
Methods

**cdf** signature(object = "vglm"): cumulative distribution function. Useful for quantile regression and extreme value data models.

**deplot** signature(object = "vglm"): density plot. Useful for quantile regression models.

**deviance** signature(object = "vglm"): deviance of the model (where applicable).

**plot** signature(x = "vglm"): diagnostic plots.

**predict** signature(object = "vglm"): extract the additive predictors or predict the additive predictors at a new data frame.

**print** signature(x = "vglm"): short summary of the object.

**qtplot** signature(object = "vglm"): quantile plot (only applicable to some models).

**resid** signature(object = "vglm"): residuals. There are various types of these.

**residuals** signature(object = "vglm"): residuals. Shorthand for resid.

**rlplot** signature(object = "vglm"): return level plot. Useful for extreme value data models.

**summary** signature(object = "vglm"): a more detailed summary of the object.

Note

VGAMs have all the slots that **vglm** objects have (**vglm-class**), plus the first few slots described in the section above.

Author(s)

Thomas W. Yee

References


See Also

**vgam.control, vglm, s, vglm-class, vglmff-class**.

Examples

```r
# Fit a nonparametric proportional odds model
pneumo <- transform(pneumo, let = log(exposure.time))
vgam(cbind(normal, mild, severe) ~ s(let),
    cumulative(parallel = TRUE), data = pneumo)
```
vgam.control

Control function for vgam

Description

Algorithmic constants and parameters for running vgam are set using this function.

Usage

vgam.control(all.knots = FALSE, bf.epsilon = 1e-07, bf.maxit = 30,
checkwz = TRUE, criterion = names(.min.criterion.VGAM),
epsilon = 1e-07, maxit = 30, na.action = na.fail,
nk = NULL, save.weights = FALSE, se.fit = TRUE,
trace = FALSE, wzepsilon = .Machine$double.eps^0.75,
...)

Arguments

  all.knots logical indicating if all distinct points of the smoothing variables are to be used as knots. By default, all.knots=TRUE for n \leq 40, and for n > 40, the number of knots is approximately 40 + (n - 40)^{0.25}. This increases very slowly with n so that the number of knots is approximately between 50 and 60 for large n.

  bf.epsilon tolerance used by the modified vector backfitting algorithm for testing convergence. Must be a positive number.

  bf.maxit maximum number of iterations allowed in the modified vector backfitting algorithm. Must be a positive integer.

  checkwz logical indicating whether the diagonal elements of the working weight matrices should be checked whether they are sufficiently positive, i.e., greater than wzepsilon. If not, any values less than wzepsilon are replaced with this value.

  criterion character variable describing what criterion is to be used to test for convergence. The possibilities are listed in .min.criterion.VGAM, but most family functions only implement a few of these.

  epsilon positive convergence tolerance epsilon. Roughly speaking, the Newton-Raphson/Fisher-scoring/local-scoring iterations are assumed to have converged when two successive criterion values are within epsilon of each other.

  maxit maximum number of Newton-Raphson/Fisher-scoring/local-scoring iterations allowed.

  na.action how to handle missing values. Unlike the SPLUS gam function, vgam cannot handle NAs when smoothing.

  nk vector of length d containing positive integers. where d be the number of s terms in the formula. Recycling is used if necessary. The ith value is the number of B-spline coefficients to be estimated for each component function of the ith s() term. nk differs from the number of knots by some constant. If specified, nk overrides the automatic knot selection procedure.
save.weights logical indicating whether the weights slot of a "glm" object will be saved on the object. If not, it will be reconstructed when needed, e.g., `summary`.

date logical indicating whether approximate pointwise standard errors are to be saved on the object. If `TRUE`, then these can be plotted with `plot(..., se = TRUE)`.

trace logical indicating if output should be produced for each iteration.

wzepsilon Small positive number used to test whether the diagonals of the working weight matrices are sufficiently positive.

... other parameters that may be picked up from control functions that are specific to the VGAM family function.

Details

Most of the control parameters are used within `vgam.fit` and you will have to look at that to understand the full details. Many of the control parameters are used in a similar manner by `vglm.fit` (`vglm`) because the algorithm (IRLS) is very similar.

Setting `save.weights=FALSE` is useful for some models because the weights slot of the object is often the largest and so less memory is used to store the object. However, for some VGAM family function, it is necessary to set `save.weights=TRUE` because the weights slot cannot be reconstructed later.

Value

A list with components matching the input names. A little error checking is done, but not much. The list is assigned to the control slot of vgam objects.

Warning

See `vglm.control`.

Note

vgam does not implement half-stepsizing, therefore parametric models should be fitted with `vglm`. Also, vgam is slower than vglm too.

Author(s)

Thomas W. Yee

References


See Also

vgam, vglm.control, vsmooth.spline, vglm.
Examples

pneumo <- transform(pneumo, let = log(exposure.time))
vgam(cbind(normal, mild, severe) ~ s(let, df = 2), multinomial, 
data = pneumo, trace = TRUE, eps = 1e-4, maxit = 10)

Description

_vglm_ is used to fit vector generalized linear models (VGLMs). This is a very large class of models
that includes generalized linear models (GLMs) as a special case.

Usage

\[
\text{vglm(formula, family, data = list(), weights = NULL, subset = NULL,} \\
\text{na.action = na.fail, etastart = NULL, mustart = NULL,} \\
\text{coefstart = NULL, control = vglm.control(...), offset = NULL,} \\
\text{method = "vglm.fit", model = FALSE, x.arg = TRUE, y.arg = TRUE,} \\
\text{contrasts = NULL, constraints = NULL, extra = list(),} \\
\text{form2 = NULL, qr.arg = TRUE, smart = TRUE, ...}
\]

Arguments

- **formula**: a symbolic description of the model to be fit. The RHS of the formula is ap-
p lied to each linear predictor. Different variables in each linear predictor can be
 chosen by specifying constraint matrices.
- **family**: a function of class "_vglmff_" (see _vglmff-class_) describing what statistical
model is to be fitted. This is called a "VGAM family function". See _CommonVGAMffArguments_ for
general information about many types of arguments found in this type of
function. The argument name "family" is used loosely and for the ease of ex-
sting _glm_ users; there is no concept of a formal “error distribution” for VGLMs.
Possibly the argument name should be better "model" but unfortunately
that name has already been taken.
- **data**: an optional data frame containing the variables in the model. By default the vari-
 ables are taken from environment(formula), typically the environment from
which _vglm_ is called.
- **weights**: an optional vector or matrix of (prior fixed and known) weights to be used in
the fitting process. If the _VGAM_ family function handles multiple responses
(Q > 1 of them, say) then weights can be a matrix with Q columns. Each
column matches the respective response. If it is a vector (the usually case) then
it is recycled into a matrix with Q columns. The values of weights must be
positive; try setting a very small value such as 1.0e-8 to effectively delete an
observation.
Currently the weights argument does not support sampling weights from com-
plex sampling designs. And currently sandwich estimators are not computed in
vglm

any shape or form. The present weights are multiplied by the corresponding
log-likelihood contributions.

subset an optional logical vector specifying a subset of observations to be used in the
fitting process.

na.action a function which indicates what should happen when the data contain NAs. The
default is set by the na.action setting of options, and is na.fail if that is
unset. The ”factory-fresh” default is na.omit.

etastart starting values for the linear predictors. It is a \( M \)-column matrix with the same
number of rows as the response. If \( M = 1 \) then it may be a vector. Note that
etastart and the output of predict(fit) should be comparable. Here, fit is
the fitted object.

mustart starting values for the fitted values. It can be a vector or a matrix; if a matrix,
then it has the same number of rows as the response. Usually mustart and the
output of fitted(fit) should be comparable. Some family functions do not
make use of this argument.

disjstart starting values for the coefficient vector. The length and order must match that
of coef(fit).

control a list of parameters for controlling the fitting process. See vglm.control for
details.

offset a vector or \( M \)-column matrix of offset values. These are \textit{a priori} known and are
added to the linear/additive predictors during fitting.

method the method to be used in fitting the model. The default (and presently only)
method vglm.fit() uses iteratively reweighted least squares (IRLS).

model a logical value indicating whether the model frame should be assigned in the
model slot.

x.arg, y.arg logical values indicating whether the model matrix and response vector/matrix
used in the fitting process should be assigned in the x and y slots. Note the
model matrix is the LM model matrix; to get the VGLM model matrix type
model.matrix(vglmfit) where vglmfit is a vglm object.

contrasts an optional list. See the contrasts.arg of model.matrix.default.

constraints an optional list of constraint matrices. The components of the list must be named
with the term it corresponds to (and it must match in character format exactly).
There are two types of input: "lm"-type and "vlm"-type. The former is a subset
of the latter. The former has a matrix for each term of the LM matrix. The latter
has a matrix for each column of the VLM matrix. After fitting, the constraints
extractor function may be applied; it returns the "vlm"-type list of constraint
matrices by default. If "lm"-type are returned by constraints then these can
be fed into this argument and it should give the same model as before.
Each constraint matrix must have \( M \) rows, and be of full-column rank. By def-

default, constraint matrices are the \( M \) by \( M \) identity matrix unless arguments in
the family function itself override these values, e.g., parallel (see CommonVGAMffArguments).
If constraints is used it must contain all the terms; an incomplete list is not
accepted.

extra an optional list with any extra information that might be needed by the VGAM
family function.
form2  The second (optional) formula. If argument xij is used (see \texttt{vglm.control}) then form2 needs to have \textit{all} terms in the model. Also, some \texttt{VGAM} family functions such as \texttt{micmen} use this argument to input the regressor variable. If given, the slots \@x2 and \@y2 may be assigned. Note that smart prediction applies to terms in form2 too.

\texttt{qr.arg}  logical value indicating whether the slot \texttt{qr}, which returns the QR decomposition of the VLM model matrix, is returned on the object.

\texttt{smart}  logical value indicating whether smart prediction (\texttt{smartpred}) will be used.

\ldots further arguments passed into \texttt{vglm.control}.

\subsection*{Details}

A vector generalized linear model (VGLM) is loosely defined as a statistical model that is a function of \( M \) linear predictors. The central formula is given by

\[ \eta_j = \beta_j^T x \]

where \( x \) is a vector of explanatory variables (sometimes just a 1 for an intercept), and \( \beta_j \) is a vector of regression coefficients to be estimated. Here, \( j = 1, \ldots, M \), where \( M \) is finite. Then one can write \( \eta = (\eta_1, \ldots, \eta_M)^T \) as a vector of linear predictors.

Most users will find \texttt{vglm} similar in flavour to \texttt{glm}. The function \texttt{vglm.fit} actually does the work.

\subsection*{Value}

An object of class "\texttt{vglm}"", which has the following slots. Some of these may not be assigned to save space, and will be recreated if necessary later:

\texttt{extra}  the list \texttt{extra} at the end of fitting.

\texttt{family}  the family function (of class "\texttt{glmff}").

\texttt{iter}  the number of IRLS iterations used.

\texttt{predictors}  a \( M \)-column matrix of linear predictors.

\texttt{assign}  a named list which matches the columns and the (LM) model matrix terms.

\texttt{call}  the matched call.

\texttt{coefficients}  a named vector of coefficients.

\texttt{constraints}  a named list of constraint matrices used in the fitting.

\texttt{contrasts}  the contrasts used (if any).

\texttt{control}  list of control parameter used in the fitting.

\texttt{criterion}  list of convergence criterion evaluated at the final IRLS iteration.

\texttt{df.residual}  the residual degrees of freedom.

\texttt{df.total}  the total degrees of freedom.

\texttt{dispersion}  the scaling parameter.

\texttt{effects}  the effects.

\texttt{fitted.values}  the fitted values, as a matrix. This is often the mean but may be quantiles, or the location parameter, e.g., in the Cauchy model.
<table>
<thead>
<tr>
<th>Slot</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>misc</td>
<td>a list to hold miscellaneous parameters.</td>
</tr>
<tr>
<td>model</td>
<td>the model frame.</td>
</tr>
<tr>
<td>na.action</td>
<td>a list holding information about missing values.</td>
</tr>
<tr>
<td>offset</td>
<td>if non-zero, a $M$-column matrix of offsets.</td>
</tr>
<tr>
<td>post</td>
<td>a list where post-analysis results may be put.</td>
</tr>
<tr>
<td>preplot</td>
<td>used by <code>plotvgam</code>, the plotting parameters may be put here.</td>
</tr>
<tr>
<td>prior.weights</td>
<td>initially supplied weights (the weights argument). Also see <code>weightsvglm</code>.</td>
</tr>
<tr>
<td>qr</td>
<td>the QR decomposition used in the fitting.</td>
</tr>
<tr>
<td>R</td>
<td>the R matrix in the QR decomposition used in the fitting.</td>
</tr>
<tr>
<td>rank</td>
<td>numerical rank of the fitted model.</td>
</tr>
<tr>
<td>residuals</td>
<td>the working residuals at the final IRLS iteration.</td>
</tr>
<tr>
<td>ResSS</td>
<td>residual sum of squares at the final IRLS iteration with the adjusted dependent vectors and weight matrices.</td>
</tr>
<tr>
<td>smart.pred</td>
<td>a list of data-dependent parameters (if any) that are used by smart prediction.</td>
</tr>
<tr>
<td>terms</td>
<td>the <code>terms</code> object used.</td>
</tr>
<tr>
<td>weights</td>
<td>the working weight matrices at the final IRLS iteration. This is in matrix-band form.</td>
</tr>
<tr>
<td>x</td>
<td>the model matrix (linear model LM, not VGLM).</td>
</tr>
<tr>
<td>xlevels</td>
<td>the levels of the factors, if any, used in fitting.</td>
</tr>
<tr>
<td>y</td>
<td>the response, in matrix form.</td>
</tr>
</tbody>
</table>

This slot information is repeated at `vglm-class`.

**WARNING**

See warnings in `vglm.control`. Also, see warnings under `weights` above regarding sampling weights from complex sampling designs.

**Note**

This function can fit a wide variety of statistical models. Some of these are harder to fit than others because of inherent numerical difficulties associated with some of them. Successful model fitting benefits from cumulative experience. Varying the values of arguments in the VGAM family function itself is a good first step if difficulties arise, especially if initial values can be inputted. A second, more general step, is to vary the values of arguments in `vglm.control`. A third step is to make use of arguments such as `etastart`, `coefstart` and `mustart`.

Some VGAM family functions end in "ff" to avoid interference with other functions, e.g., `binomialff`, `poissonff`, `gaussianff`, `gammaff`. This is because VGAM family functions are incompatible with `glm` (and also `gam` in the `gam` library and `gam` in the `mgcv` library).

The smart prediction (`smartpred`) library is incorporated within the VGAM library.

The theory behind the scaling parameter is currently being made more rigorous, but it should give the same value as the scale parameter for GLMs.
In Example 5 below, the $x_{ij}$ argument to illustrate covariates that are specific to a linear predictor. Here, $lop/rop$ are the ocular pressures of the left/right eye (artificial data). Variables $leye$ and $reye$ might be the presence/absence of a particular disease on the LHS/RHS eye respectively. See \texttt{vglm.control} and \texttt{fill} for more details and examples.

**Author(s)**

Thomas W. Yee

**References**


**See Also**

\texttt{vglm.control}, \texttt{vglm-class}, \texttt{vglmff-class}, \texttt{smartpred}, \texttt{vglm.fit}, \texttt{fill}, \texttt{rrvglm}, \texttt{vgam}. Methods functions include \texttt{coef.vlm}, \texttt{constraints.vlm}, \texttt{hatvalues.vlm}, \texttt{linkfun.vglm}, \texttt{predict.vglm}, \texttt{summary.vglm}, \texttt{AIC.vglm}, \texttt{lrtest_vglm}, etc.

**Examples**

```
# Example 1. See help(glm)
print(d.AD <- data.frame(treatment = gl(3, 3),
                         outcome = gl(3, 1, 9),
                         counts = c(18,17,15,20,10,20,25,13,12)))
vglm.D93 <- vglm(counts ~ outcome + treatment, family = poissonff,
               data = d.AD, trace = TRUE)
summary(vglm.D93)

# Example 2. Multinomial logit model
pneumo <- transform(pneumo, let = log(exposure.time))
vglm(cbind(normal, mild, severe) ~ let, multinomial, data = pneumo)

# Example 3. Proportional odds model
fit3 <- vglm(cbind(normal, mild, severe) ~ let, propodds, data = pneumo)
coef(fit3, matrix = TRUE)
constraints(fit3)
model.matrix(fit3, type = "lm")  # LM model matrix
model.matrix(fit3)  # Larger VGLM (or VLM) model matrix

# Example 4. Bivariate logistic model
fit4 <- vglm(cbind(nBnW, nBW, BnW, BW) ~ age, binom2.or, coalminers)
coef(fit4, matrix = TRUE)
depvar(fit4)  # Response are proportions
```
weights(fit4, type = "prior")

# Example 5. The use of the xij argument (simple case).
# The constraint matrix for 'op' has one column.
nn <- 1000
eyesdat <- round(data.frame(lop = runif(nn),
                           rop = runif(nn),
                           op = runif(nn)), digits = 2)
eyesdat <- transform(eyesdat, eta1 = -1 + 2 * lop,
                      eta2 = -1 + 2 * lop)
eyesdat <- transform(eyesdat,
                     leye = rbinom(nn, size = 1, prob = logit(eta1, inverse = TRUE)),
                     reye = rbinom(nn, size = 1, prob = logit(eta2, inverse = TRUE)))
head(eyesdat)
fit5 <- vglm(cbind(leye, reye) ~ op,
             binom2.or(exchangeable = TRUE, zero = 3),
             data = eyesdat, trace = TRUE,
             xij = list(op ~ lop + rop + fill(lop)),
             form2 = ~ op + lop + rop + fill(lop))
coef(fit5)
coef(fit5, matrix = TRUE)
constraints(fit5)

---

**vglm-class**  
*Class “vglm”*

**Description**

Vector generalized linear models.

**Objects from the Class**

Objects can be created by calls of the form `vglm(...)`.

**Slots**

In the following, $M$ is the number of linear predictors.

- `extra`: Object of class "list"; the extra argument on entry to `vglm`. This contains any extra information that might be needed by the family function.
- `family`: Object of class "vglmff". The family function.
- `iter`: Object of class "numeric". The number of IRLS iterations used.
- `predictors`: Object of class "matrix" with $M$ columns which holds the $M$ linear predictors.
- `assign`: Object of class "list", from class “vlm”. This named list gives information matching the columns and the (LM) model matrix terms.
- `call`: Object of class "call", from class “vlm”. The matched call.
coefficients: Object of class "numeric", from class "vlm". A named vector of coefficients.
constraints: Object of class "list", from class "vlm". A named list of constraint matrices
used in the fitting.
contrasts: Object of class "list", from class "vlm". The contrasts used (if any).
control: Object of class "list", from class "vlm". A list of parameters for controlling the fitting
process. See vglm.control for details.
criterion: Object of class "list", from class "vlm". List of convergence criterion evaluated at
the final IRLS iteration.
df.residual: Object of class "numeric", from class "vlm". The residual degrees of freedom.
df.total: Object of class "numeric", from class "vlm". The total degrees of freedom.
dispersion: Object of class "numeric", from class "vlm". The scaling parameter.
effects: Object of class "numeric", from class "vlm". The effects.
fitted.values: Object of class "matrix", from class "vlm". The fitted values.
misc: Object of class "list", from class "vlm". A named list to hold miscellaneous parameters.
model: Object of class "data.frame", from class "vlm". The model frame.
na.action: Object of class "list", from class "vlm". A list holding information about missing
values.
offset: Object of class "matrix", from class "vlm". If non-zero, a $M$-column matrix of offsets.
post: Object of class "list", from class "vlm" where post-analysis results may be put.
preplot: Object of class "list", from class "vlm" used by plotvgam; the plotting parameters
may be put here.
prior.weights: Object of class "matrix", from class "vlm" holding the initially supplied weights.
qr: Object of class "list", from class "vlm". QR decomposition at the final iteration.
R: Object of class "matrix", from class "vlm". The $R$ matrix in the QR decomposition used in
the fitting.
rank: Object of class "integer", from class "vlm". Numerical rank of the fitted model.
residuals: Object of class "matrix", from class "vlm". The working residuals at the final IRLS
iteration.
ReS5S: Object of class "numeric", from class "vlm". Residual sum of squares at the final IRLS
iteration with the adjusted dependent vectors and weight matrices.
smart.prediction: Object of class "list", from class "vlm". A list of data-dependent parameters
(if any) that are used by smart prediction.
terms: Object of class "list", from class "vlm". The terms object used.
weights: Object of class "matrix", from class "vlm". The weight matrices at the final IRLS
iteration. This is in matrix-band form.
x: Object of class "matrix", from class "vlm". The model matrix (LM, not VGLM).
xlevels: Object of class "list", from class "vlm". The levels of the factors, if any, used in
fitting.
y: Object of class "matrix", from class "vlm". The response, in matrix form.
Xm2: Object of class "matrix", from class "vlm". See vglm-class).
Ym2: Object of class "matrix", from class "vlm". See vglm-class).
callXm2: Object of class "call", from class "vlm". The matched call for argument form2.
Extends

Class "v1m", directly.

Methods

cdf signature(object = "vglm"): cumulative distribution function. Applicable to, e.g., quantile regression and extreme value data models.
depplot signature(object = "vglm"): Applicable to, e.g., quantile regression.
deviance signature(object = "vglm"): deviance of the model (where applicable).
plot signature(x = "vglm"): diagnostic plots.
predict signature(object = "vglm"): extract the linear predictors or predict the linear predictors at a new data frame.
print signature(x = "vglm"): short summary of the object.
qtplot signature(object = "vglm"): quantile plot (only applicable to some models).
resid signature(object = "vglm"): residuals. There are various types of these.
residuals signature(object = "vglm"): residuals. Shorthand for resid.
rlplot signature(object = "vglm"): return level plot. Useful for extreme value data models.
summary signature(object = "vglm"): a more detailed summary of the object.

Author(s)

Thomas W. Yee

References


See Also

vglm, vglmff-class, vgam-class.

Examples

# Multinomial logit model
pneumo <- transform(pneumo, let = log(exposure.time))
vglm(cbind(normal, mild, severe) ~ let, multinomial, data = pneumo)
vglm.control  Control function for vglm

Description
Algorithmic constants and parameters for running vglm are set using this function.

Usage
vglm.control(checkwz = TRUE, Check.rank = TRUE, Check.cm.rank = TRUE,
criterion = names(.min.criterion.VGAM),
epsilon = 1e-07, half.stepsizing = TRUE,
maxit = 30, nowarning = FALSE,
stepsize = 1, save.weights = FALSE,
trace = FALSE, wzepsilon = .Machine$double.eps^0.75,
xij = NULL, ...)

Arguments
checkwz  logical indicating whether the diagonal elements of the working weight matrices should be checked whether they are sufficiently positive, i.e., greater than wzepsilon. If not, any values less than wzepsilon are replaced with this value.
Check.rank  logical indicating whether the rank of the VLM matrix should be checked. If this is not of full column rank then the results are not to be trusted. The default is to give an error message if the VLM matrix is not of full column rank.
Check.cm.rank  logical indicating whether the rank of each constraint matrix should be checked. If this is not of full column rank then an error will occur. Under no circumstances should any constraint matrix have a rank less than the number of columns.
criterion  character variable describing what criterion is to be used to test for convergence. The possibilities are listed in .min.criterion.VGAM, but most family functions only implement a few of these.
epsilon  positive convergence tolerance epsilon. Roughly speaking, the Newton-Raphson/Fisher-scoring iterations are assumed to have converged when two successive criterion values are within epsilon of each other.
half.stepsizing  logical indicating if half-stepsizing is allowed. For example, in maximizing a log-likelihood, if the next iteration has a log-likelihood that is less than the current value of the log-likelihood, then a half step will be taken. If the log-likelihood is still less than at the current position, a quarter-step will be taken etc. Eventually a step will be taken so that an improvement is made to the convergence criterion. half.stepsizing is ignored if criterion == "coefficients".
maxit  maximum number of (usually Fisher-scoring) iterations allowed. Sometimes Newton-Raphson is used.
nowarning  logical indicating whether to suppress a warning if convergence is not obtained within maxit iterations. This is ignored if maxit = 1 is set.
**stepsizes**

usual step size to be taken between each Newton-Raphson/Fisher-scoring iteration. It should be a value between 0 and 1, where a value of unity corresponds to an ordinary step. A value of 0.5 means half-steps are taken. Setting a value near zero will cause convergence to be generally slow but may help increase the chances of successful convergence for some family functions.

**save.weights**

logical indicating whether the weights slot of a "vglm" object will be saved on the object. If not, it will be reconstructed when needed, e.g., summary. Some family functions have save.weights = TRUE and others have save.weights = FALSE in their control functions.

**trace**

logical indicating if output should be produced for each iteration. Setting trace = TRUE is recommended in general because VGAM fits a very broad variety of models and distributions, and for some of them, convergence is intrinsically more difficult. Monitoring convergence can help check that the solution is reasonable or that a problem has occurred. It may suggest better initial values are needed, the making of invalid assumptions, or that the model is inappropriate for the data, etc.

**wzepsilon**

small positive number used to test whether the diagonals of the working weight matrices are sufficiently positive.

**xij**

A formula or a list of formulas. Each formula has a RHS giving $M$ terms making up a covariate-dependent term (whose name is the response). That is, it creates a variable that takes on different values for each linear/additive predictor, e.g., the ocular pressure of each eye. The $M$ terms must be unique; use \texttt{fill1}, \texttt{fill2}, \texttt{fill3}, etc. if necessary. Each formula should have a response which is taken as the name of that variable, and the $M$ terms are enumerated in sequential order. Each of the $M$ terms multiply each successive row of the constraint matrix. When xij is used, the use of form2 is also required to give every term used by the model.

The function \texttt{Select} can be used to select variables beginning with the same character string.

... other parameters that may be picked up from control functions that are specific to the VGAM family function.

**Details**

Most of the control parameters are used within vglm.fit and you will have to look at that to understand the full details.

Setting save.weights = FALSE is useful for some models because the weights slot of the object is the largest and so less memory is used to store the object. However, for some VGAM family function, it is necessary to set save.weights = TRUE because the weights slot cannot be reconstructed later.

**Value**

A list with components matching the input names. A little error checking is done, but not much. The list is assigned to the control slot of vglm objects.
Warning

For some applications the default convergence criterion should be tightened. Setting something like `criterion = "coef", epsilon = 1e-09` is one way to achieve this, and also add `trace = TRUE` to monitor the convergence. Setting `maxit` to some higher number is usually not needed, and needing to do so suggests something is wrong, e.g., an ill-conditioned model, over-fitting or under-fitting.

Note

Reiterating from above, setting `trace = TRUE` is recommended in general.

In Example 2 below there are two covariates that have linear/additive predictor specific values. These are handled using the `xij` argument.

Author(s)

Thomas W. Yee

References


See Also

`vglm`, `fillQ`. The author’s homepage has further documentation about the `xij` argument; see also `select`.

Examples

```r
# Example 1.
pneumo <- transform(pneumo, let = log(exposure.time))
vglm(cbind(normal, mild, severe) ~ let, multinomial, data = pneumo,
    crit = "coef", step = 0.5, trace = TRUE, epsilon = 1e-8, maxit = 40)
```

```r
# Example 2. The use of the xij argument (simple case).
ymat <- rdiric(n <- 1000, shape = rep(exp(2), len = 4))
mydat <- data.frame(x1 = runif(n), x2 = runif(n), x3 = runif(n), x4 = runif(n),
    z1 = runif(n), z2 = runif(n), z3 = runif(n), z4 = runif(n))
mydat <- transform(mydat, X = x1, Z = z1)
mydat <- round(mydat, digits = 2)
fit2 <- vglm(ymat ~ X + Z,
    dirichlet(parallel = TRUE), data = mydat, trace = TRUE,
    xij = list(Z ~ z1 + z2 + z3 + z4,
        X ~ x1 + x2 + x3 + x4),
    form2 = ~ Z + z1 + z2 + z3 + z4 +
        X + x1 + x2 + x3 + x4)
head(model.matrix(fit2, type = "lm"))  # LM model matrix
head(model.matrix(fit2, type = "vlm"))  # Big VLM model matrix
coef(fit2)
coef(fit2, matrix = TRUE)
max(abs(predict(fit2)-predict(fit2, new = mydat)))  # Predicts correctly
```
Example 3. The use of the xij argument (complex case).

```r
set.seed(123)
coalminers <- transform(coalminers,
  Age = (age - 42) / 5,
  dum1 = round(runif(nrow(coalminers)), digits = 2),
  dum2 = round(runif(nrow(coalminers)), digits = 2),
  dum3 = round(runif(nrow(coalminers)), digits = 2),
  dummy = round(runif(nrow(coalminers)), digits = 2))
BS <- function(x, ..., df = 3) sm.bs(c(x,...), df = df)[1:length(x),drop = FALSE]
NS <- function(x, ..., df = 3) sm.ns(c(x,...), df = df)[1:length(x),drop = FALSE]

# Equivalently...
BS <- function(x, ..., df = 3)
  head(sm.bs(c(x,...), df = df), length(x), drop = FALSE)
NS <- function(x, ..., df = 3)
  head(sm.ns(c(x,...), df = df), length(x), drop = FALSE)

fit3 <- vglm(cbind(nBnW,nBw,BnW,Bw) ~ Age + NS(dum1, dum2),
  fam = binom2.or(exchangeable = TRUE, zero = 3),
  xij = list(NS(dum1, dum2) ~ NS(dum1, dum2) +
  NS(dum2, dum1) +
  fill(NS( dum1))),
  form2 = ~ NS(dum1, dum2) + NS(dum2, dum1) + fill(NS(dum1)) +
  dum1 + dum2 + dum3 + Age + age + dummy,
  data = coalminers, trace = TRUE)
head(model.matrix(fit3, type = "lm"))  # LM model matrix
head(model.matrix(fit3, type = "vlm"))  # Big VLM model matrix
coef(fit3)
## Not run: plotvgam(fit3, se = TRUE, lcol = "red", sclor = "blue", xlab = "dum1")
```

Description

Family functions for the VGAM package

Objects from the Class

Objects can be created by calls of the form new("vglmff", ...).
Slots

In the following, \( M \) is the number of linear/additive predictors.

blurb: Object of class "character" giving a small description of the model. Important arguments such as parameter link functions can be expressed here.

consstraints: Object of class "expression" which sets up any constraint matrices defined by arguments in the family function. A zero argument is always fed into `cm.zero.vgam`, whereas other constraints are fed into `cm.vgam`.

deviance: Object of class "function" returning the deviance of the model. This slot is optional. If present, the function must have arguments `function(mu, y, w, residuals = FALSE, eta, extra = NULL)`. Deviance residuals are returned if `residuals = TRUE`.

fini: Object of class "expression" to insert code at a special position in `vglm.fit` or `vgam.fit`. This code is evaluated immediately after the fitting.

first: Object of class "expression" to insert code at a special position in `vglm` or `vgam`.

infos: Object of class "function" which returns a list with components such as `M1`. At present only a very few VGAM family functions have this feature implemented. Those that do not require specifying the `M1` argument when used with `rcim`.

initialize: Object of class "expression" used to perform error checking (especially for the variable \( y \)) and obtain starting values for the model. In general, `etastart` or `mustart` are assigned values based on the variables \( y, x \) and \( w \).

linkinv: Object of class "function" which returns the fitted values, given the linear/additive predictors. The function must have arguments `function(eta, extra = NULL)`.

last: Object of class "expression" to insert code at a special position (at the very end) of `vglm.fit()` or `vgam.fit()`. This code is evaluated after the fitting. The list `misc` is often assigned components in this slot, which becomes the `misc` slot on the fitted object.

linkfun: Object of class "function" which, given the fitted values, returns the linear/additive predictors. If present, the function must have arguments `function(mu, extra = NULL)`. Most VGAM family functions do not have a `linkfun` function. They largely are for classical exponential families, i.e., GLMs.

loglikelihood: Object of class "function" returning the log-likelihood of the model. This slot is optional. If present, the function must have arguments `function(mu, y, w, residuals = FALSE, eta, extra = NULL)`.

The argument `residuals` can be ignored because log-likelihood residuals aren’t defined.

middle: Object of class "expression" to insert code at a special position in `vglm.fit` or `vgam.fit`.

middle2: Object of class "expression" to insert code at a special position in `vglm.fit` or `vgam.fit`.

simslot: Object of class "function" to allow `simulate` to work.

summary.dispersion: Object of class "logical" indicating whether the general VGLM formula (based on a residual sum of squares) can be used for computing the scaling/dispersion parameter. It is `TRUE` for most models except for nonlinear regression models.

vfamily: Object of class "character" giving class information about the family function. Although not developed at this stage, more flexible classes are planned in the future. For example, family functions `sratio`, `cratio`, `cumulative`, and `acat` all operate on categorical data, therefore will have a special class called "VGAMcat", say. Then if `fit` was a `vglm` object, then `coef(fit)` would print out the `vglm` coefficients plus "VGAMcat" information as well.
deriv: Object of class "expression" which returns a $M$-column matrix of first derivatives of the log-likelihood function with respect to the linear/additive predictors, i.e., the score vector. In Yee and Wild (1996) this is the $d_i$ vector. Thus each row of the matrix returned by this slot is such a vector.

weight: Object of class "expression" which returns the second derivatives of the log-likelihood function with respect to the linear/additive predictors. This can be either the observed or expected information matrix, i.e., Newton-Raphson or Fisher-scoring respectively. In Yee and Wild (1996) this is the $W_i$ matrix. Thus each row of the matrix returned by this slot is such a matrix. Like the weights slot of vglm/vgam, it is stored in matrix-band form, whereby the first $M$ columns of the matrix are the diagonals, followed by the upper-diagonal band, followed by the band above that, etc. In this case, there can be up to $M(M + 1)$ columns, with the last column corresponding to the $(1,M)$ elements of the weight matrices.

Methods

print signature(x = "vglmff"): short summary of the family function.

Warning

VGAM family functions are not compatible with glm, nor gam (from either gam or mgev packages).

Note

With link functions etc., one must use substitute to embed the options into the code. There are two different forms: eval(substitute(expression(...), list(...))) for expressions, and eval(substitute( function(...) { ... }, list(...) )) for functions.

The extra argument in linkinv, linkfun, deviance, loglikelihood, etc. matches with the argument extra in vglm, vgam and rrvglm. This allows input to be fed into all slots of a VGAM family function.

The expression derivative is evaluated immediately prior to weight, so there is provision for re-use of variables etc. Programmers must be careful to choose variable names that do not interfere with vglm.fit, vgam.fit() etc.

Programmers of VGAM family functions are encouraged to keep to previous conventions regarding the naming of arguments, e.g., link is the argument for parameter link functions, zero for allowing some of the linear/additive predictors to be an intercept term only, etc.

In general, Fisher-scoring is recommended over Newton-Raphson where tractable. Although usually slightly slower in convergence, the weight matrices from using the expected information are positive-definite over a larger parameter space.

Author(s)

Thomas W. Yee

References

See Also

vglm, vgam, rrvglm, rcim.

Examples

```r
cratio()
cratio(link = "cloglog")
cratio(link = "cloglog", reverse = TRUE)
```

---

**vonmises**

*von Mises Distribution Family Function*

**Description**

Estimates the location and scale parameters of the von Mises distribution by maximum likelihood estimation.

**Usage**

```r
vonmises(llocation = extlogit(min = 0, max = 2 * pi), lscale = "loge", ilocation = NULL, iscale = NULL, imethod = 1, zero = NULL)
```

**Arguments**

- `llocation, lscale`  
  Parameter link functions applied to the location $\alpha$ parameter and scale parameter $k$, respectively. See Links for more choices. For $k$, a log link is the default because the parameter is positive.

- `ilocation`  
  Initial value for the location $\alpha$ parameter. By default, an initial value is chosen internally using `imethod`. Assigning a value will override the argument `imethod`.

- `iscale`  
  Initial value for the scale $k$ parameter. By default, an initial value is chosen internally using `imethod`. Assigning a value will override the argument `imethod`.

- `imethod`  
  An integer with value 1 or 2 which specifies the initialization method. If failure to converge occurs try the other value, or else specify a value for `ilocation` and `iscale`.

- `zero`  
  An integer-valued vector specifying which linear/additive predictors are modelled as intercepts only. The default is none of them. If used, choose one value from the set $\{1,2\}$. 
The (two-parameter) von Mises is the most commonly used distribution in practice for circular data. It has a density that can be written as

\[
f(y; a, k) = \frac{\exp[k \cos(y - a)]}{2\pi I_0(k)}
\]

where \(0 \leq y < 2\pi\), \(k > 0\) is the scale parameter, \(a\) is the location parameter, and \(I_0(k)\) is the modified Bessel function of order 0 evaluated at \(k\). The mean of \(Y\) (which is the fitted value) is \(a\) and the circular variance is \(1 - I_1(k)/I_0(k)\) where \(I_1(k)\) is the modified Bessel function of order 1. By default, \(\eta_1 = \log(a/(2\pi - a))\) and \(\eta_2 = \log(k)\) for this family function.

The response and the fitted values are scaled so that \(0 \leq y < 2\pi\). The linear/additive predictors are left alone. Fisher scoring is used.

The response and the fitted values are scaled so that \(0 \leq y < 2\pi\). The linear/additive predictors are left alone. Fisher scoring is used.

Author(s)

T. W. Yee

References


See Also

Bessel, cardioid.

CircStats and circular currently have a lot more R functions for circular data than the VGAM package.

Examples

```r
vdata <- data.frame(x2 = runif(nn <- 1000))
vdata <- transform(vdata, y = rnorm(nn, m = 2+x2, sd = exp(0.2))) # Bad data!!
fit <- vglm(y ~ x2, vonmises(zero = 2), data = vdata, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
```
**vsmooth.spline**

**Vector cubic smoothing spline**

**Description**

Fits a vector cubic smoothing spline.

**Usage**

```r
vsmooth.spline(x, y, w = NULL, df = rep(5, M), spar = NULL,
    i.constraint = diag(M),
    x.constraint = diag(M),
    constraints = list("(Intercepts)" = i.constraint,
        x = x.constraint),
    all.knots = FALSE, var.arg = FALSE, scale.w = TRUE,
    nk = NULL, control.spar = list())
```

**Arguments**

- **x**
  - A vector, matrix or a list. If a list, the x component is used. If a matrix, the first column is used. x may also be a complex vector, in which case the real part is used, and the imaginary part is used for the response. In this help file, n is the number of unique values of x.

- **y**
  - A vector, matrix or a list. If a list, the y component is used. If a matrix, all but the first column is used. In this help file, M is the number of columns of y if there are no constraints on the functions.

- **w**
  - The weight matrices or the number of observations. If the weight matrices, then this must be a n-row matrix with the elements in matrix-band form (see `iam`). If a vector, then these are the number of observations. By default, w is the M by M identity matrix, denoted by \( \text{matrix}(1, n, M) \).

- **df**
  - Numerical vector containing the degrees of freedom for each component function (smooth). If necessary, the vector is recycled to have length equal to the number of component functions to be estimated (M if there are no constraints), which equals the number of columns of the x-constraint matrix. A value of 2 means a linear fit, and each element of df should lie between 2 and n. The larger the values of df the more wiggly the smooths.

- **spar**
  - Numerical vector containing the non-negative smoothing parameters for each component function (smooth). If necessary, the vector is recycled to have length equal to the number of component functions to be estimated (M if there are no constraints), which equals the number of columns of the x-constraint matrix. A value of zero means the smooth goes through the data and hence is wiggly. A value of Inf may be assigned, meaning the smooth will be linear. By default, the NULL value of spar means df is used to determine the smoothing parameters.
all.knots Logical. If TRUE then each distinct value of x will be a knot. By default, only a subset of the unique values of x are used; typically, the number of knots is \(O(n^{0.25})\) for \(n\) large, but if \(n \leq 40\) then all the unique values of x are used.

e.constraint A \(m\)-row constraint matrix for the intercepts. It must be of full column rank. By default, the constraint matrix for the intercepts is the \(M\) by \(M\) identity matrix, meaning no constraints.

x.constraint A \(m\)-row constraint matrix for \(x\). It must be of full column rank. By default, the constraint matrix for the intercepts is the \(M\) by \(M\) identity matrix, meaning no constraints.

constraints An alternative to specifying e.constraint and x.constraint, this is a list with two components corresponding to the intercept and \(x\) respectively. They must both be a \(m\)-row constraint matrix with full column rank.

var.arg Logical: return the pointwise variances of the fit? Currently, this corresponds only to the nonlinear part of the fit, and may be wrong.

scale.w Logical. By default, the weights \(w\) are scaled so that the diagonal elements have mean 1.

nk Number of knots. If used, this argument overrides all.knots, and must lie between 6 and \(n+2\) inclusive.

control.spar See smooth.spline.

Details

The algorithm implemented is detailed in Yee (2000). It involves decomposing the component functions into a linear and nonlinear part, and using B-splines. The cost of the computation is \(O(n M^3)\).

The argument spar contains scaled smoothing parameters.

Value

An object of class "vsmooth.spline" (see vsmooth.spline-class).

WARNING

See vgam for information about an important bug.

Note

This function is quite similar to smooth.spline but offers less functionality. For example, cross validation is not implemented here. For \(M = 1\), the results will be generally different, mainly due to the different way the knots are selected.

The vector cubic smoothing spline which \(s()\) represents is computationally demanding for large \(M\). The cost is approximately \(O(n M^3)\) where \(n\) is the number of unique abscissae.

Yet to be done: return the unscaled smoothing parameters.

Author(s)

Thomas W. Yee
References


See Also

vsmooth.spline-class, plot.vsmooth.spline, predict.vsmooth.spline, lam, s, smooth.spline.

Examples

```r
nn <- 20; x <- 2 + 5*(nn:1)/nn
y1 <- sin(x) + rnorm(nn, sd = 0.13)
y2 <- cos(x) + rnorm(nn, sd = 0.13)
y3 <- 1 + sin(x) + rnorm(nn, sd = 0.13)  # Run this for constraints
y <- cbind(y1, y2, y3)
ww <- cbind(rep(3, nn), 4, (1:nn)/nn)

(fit <- vsmooth.spline(x, y, w = ww, df = 5))
## Not run:
plot(fit)  # The 1st and 3rd functions do not differ by a constant

## End(Not run)

mat <- matrix(c(1,0,1, 0,1,0), 3, 2)
(fit2 <- vsmooth.spline(x, y, w = ww, df = 5, i.constr = mat, x.constr = mat))
# The 1st and 3rd functions do differ by a constant:
mycols <- c("orange", "blue", "orange")
## Not run: plot(fit2, lcol = mycols, pcol = mycols, las = 1)

p <- predict(fit, x = model.matrix(fit, type = "lm"), deriv = 0)
max(abs(depvar(fit) - with(p, y)))  # Should be 0; and fit@y is not good

par(mfrow = c(3, 1))
ux <- seq(1, 8, len = 100)
for (dd in 1:3) {
  pp <- predict(fit, x = ux, deriv = dd)
  ## Not run: with(pp, matplot(x, y, type = "1", main = paste("deriv = ", dd), lwd = 2, ylab = "", cex.axis = 1.5, cex.lab = 1.5, cex.main = 1.5))

  ## End(Not run)
}
```

waitakere  
Waitakere Ranges Data


**Description**

The waitakere data frame has 579 rows and 18 columns. Altitude is explanatory, and there are binary responses (presence/absence = 1/0 respectively) for 17 plant species.

**Usage**

```r
data(waitakere)
```

**Format**

This data frame contains the following columns:

- **agaaus** Agathis australis, or Kauri
- **beitaw** Beilschmiedia tawa, or Tawa
- **corlæ** Corynocarpus laevigatus
- **cyadea** Cyathea dealbata
- **cyamed** Cyathea medullaris
- **daccup** Dacrydium cupressinum
- **dacdac** Dacrycarpus dacrydioides
- **eladen** Elaeocarpus dentatus
- **hedarb** Hedycarya arborea
- **hohpop** Species name unknown
- **kniexc** Knightia excelsa, or Rewarewa
- **kuneri** Kunzea ericoides
- **lepsco** Leptospermum scoparium
- **metrob** Metrosideros robusta
- **neslan** Nestegis lanceolata
- **rhosap** Rhopalostylis sapida
- **vitluc** Vitex lucens, or Puriri
- **altitude** meters above sea level

**Details**

These were collected from the Waitakere Ranges, a small forest in northern Auckland, New Zealand. At 579 sites in the forest, the presence/absence of 17 plant species was recorded, as well as the altitude. Each site was of area size 200m$^2$.

**Source**

Dr Neil Mitchell, University of Auckland.

**See Also**

- `hunua`
Examples

```r
fit <- vgam(agaus ~ s(altitude, df = 2), binomialff, waitakere)
head(predict(fit, waitakere, type = "response"))
## Not run: plot(fit, se = TRUE, lcol = "orange", scol = "blue")
```

### waldff

#### Wald Distribution Family Function

#### Description

Estimates the parameter of the standard Wald distribution by maximum likelihood estimation.

#### Usage

```r
waldff(link.lamb = "loge", init.lamb = NULL)
```

#### Arguments

- `link.lamb`: Parameter link function for the \( \lambda \) parameter. See [Links](#) for more choices and general information.
- `init.lamb`: Initial value for the \( \lambda \) parameter. The default means an initial value is chosen internally.

#### Details

The standard Wald distribution is a special case of the inverse Gaussian distribution with \( \mu = 1 \). It has a density that can be written as

\[
f(y; \lambda) = \frac{\lambda}{(2\pi y^3)} \exp \left(-\lambda(y - 1)^2/(2y)\right)
\]

where \( y > 0 \) and \( \lambda > 0 \). The mean of \( Y \) is 1 (returned as the fitted values) and its variance is \( 1/\lambda \). By default, \( \eta = \log(\lambda) \).

#### Value

An object of class "vglmff" (see [vglmff-class](#)). The object is used by modelling functions such as `vglm`, and `vgam`.

#### Note

The VGAM family function `inv.gaussianff` estimates the location parameter \( \mu \) too.

#### Author(s)

T. W. Yee
weibullR

References


See Also

`inv.gaussianff, rinv.gaussian`.

Examples

```r
wdata <- data.frame(y = rinv.gaussian(n = 1000, mu = 1, lambda = exp(1)))
fit <- vglm(y ~ 1, waldff(init = 0.2), data = wdata, trace = TRUE)
coef(fit, matrix = TRUE)
Coef(fit)
summary(fit)
```

weibullR  Weibull Distribution Family Function

Description

Maximum likelihood estimation of the 2-parameter Weibull distribution. No observations should be censored.

Usage

```r
weibullR(lshape = "loge", lscale = "loge",
        ishape = NULL, iscale = NULL, lss = TRUE, nrfs = 1,
        probs.y = c(0.2, 0.5, 0.8), imethod = 1, zero = ifelse(lss, -2, -1))
```

Arguments

- `lshape, lscale`: Parameter link functions applied to the (positive) shape parameter (called \(a\) below) and (positive) scale parameter (called \(b\) below). See `Links` for more choices.
- `ishape, iscale`: Optional initial values for the shape and scale parameters.
- `nrfs`: Currently this argument is ignored. Numeric, of length one, with value in \([0, 1]\). Weighting factor between Newton-Raphson and Fisher scoring. The value 0 means pure Newton-Raphson, while 1 means pure Fisher scoring. The default value uses a mixture of the two algorithms, and retaining positive-definite working weights.
- `imethod`: Initialization method used if there are censored observations. Currently only the values 1 and 2 are allowed.
- `zero, probs.y, lss`: Details at `CommonVGAMffArguments`.


Details

The Weibull density for a response \( Y \) is

\[
f(y; a, b) = ay^{a-1} \exp[-(y/b)^a]/(b^a)
\]

for \( a > 0, b > 0, y > 0 \). The cumulative distribution function is

\[
F(y; a, b) = 1 - \exp[-(y/b)^a].
\]

The mean of \( Y \) is \( b \Gamma(1 + 1/a) \) (returned as the fitted values), and the mode is at \( b(1 - 1/a)^{1/a} \) when \( a > 1 \). The density is unbounded for \( a < 1 \). The \( k \)th moment about the origin is \( E(Y^k) = b^k \Gamma(1 + k/a) \). The hazard function is \( at^{a-1}/b^a \).

This VGAM family function currently does not handle censored data. Fisher scoring is used to estimate the two parameters. Although the expected information matrices used here are valid in all regions of the parameter space, the regularity conditions for maximum likelihood estimation are satisfied only if \( a > 2 \) (according to Kleiber and Kotz (2003)). If this is violated then a warning message is issued. One can enforce \( a > 2 \) by choosing `1shape = logoff(offset = -2)`. Common values of the shape parameter lie between 0.5 and 3.5.

Summarized in Harper et al. (2011), for inference, there are 4 cases to consider. If \( a \leq 1 \) then the MLEs are not consistent (and the smallest observation becomes a hyperefficient solution for the location parameter in the 3-parameter case). If \( 1 < a < 2 \) then MLEs exist but are not asymptotically normal. If \( a = 2 \) then the MLEs exist and are normal and asymptotically efficient but with a slower convergence rate than when \( a > 2 \). If \( a > 2 \) then MLEs have classical asymptotic properties.

The 3-parameter (location is the third parameter) Weibull can be estimated by maximizing a profile log-likelihood (see, e.g., Harper et al. (2011) and Lawless (2003)), else try `gev` which is a better parameterization.

Value

An object of class "vglmff" (see `vglmff-class`). The object is used by modelling functions such as `vglm`, and `vgam`.

Warning

This function is under development to handle other censoring situations. The version of this function which will handle censored data will be called `cenweibull()`. It is currently being written and will use `SurvS4` as input. It should be released in later versions of VGAM.

If the shape parameter is less than two then misleading inference may result, e.g., in the `summary` and `vcov` of the object.

Note

Successful convergence depends on having reasonably good initial values. If the initial values chosen by this function are not good, make use the two initial value arguments.

This VGAM family function handles multiple responses.

The Weibull distribution is often an alternative to the lognormal distribution. The inverse Weibull distribution, which is that of \( 1/Y \) where \( Y \) has a Weibull(\( a, b \)) distribution, is known as the log-Gompertz distribution.
There are problems implementing the three-parameter Weibull distribution. These are because the classical regularity conditions for the asymptotic properties of the MLEs are not satisfied because the support of the distribution depends on one of the parameters.

Other related distributions are the Maxwell and Rayleigh distributions.

Author(s)

T. W. Yee

References


See Also
dweibull, truncweibull, gev, lognormal, expexpff, maxwell, rayleigh, gumbelII.

Examples

```r
wdata <- data.frame(x2 = runif(nn <- 1000)) # Complete data
wdata <- transform(wdata,
   y1 = rweibull(nn, shape = exp(1), scale = exp(-2 + x2)),
   y2 = rweibull(nn, shape = exp(2), scale = exp(1 - x2))
fit <- vglm(cbind(y1, y2) ~ x2, weibullR, data = wdata, trace = TRUE)
coef(fit, matrix = TRUE)
vcov(fit)
summary(fit)
```
weightsvglm  Prior and Working Weights of a VGLM fit

Description

Returns either the prior weights or working weights of a VGLM object.

Usage

weightsvglm(object, type = c("prior", "working"),
matrix.arg = TRUE, ignore.slot = FALSE,
deriv.arg = FALSE, ...)

Arguments

object  a model object from the VGAM R package that inherits from a vector generalized linear model (VGLM), e.g., a model of class "vglm".
type  Character, which type of weight is to be returned? The default is the first one.
matrix.arg  Logical, whether the answer is returned as a matrix. If not, it will be a vector.
ignore.slot  Logical. If TRUE then object@weights is ignored even if it has been assigned, and the long calculation for object@weights is repeated. This may give a slightly different answer because of the final IRLS step at convergence may or may not assign the latest value of quantities such as the mean and weights.
deriv.arg  Logical. If TRUE then a list with components deriv and weights is returned. See below for more details.
...  Currently ignored.

details

Prior weights are usually inputted with the weights argument in functions such as vglm and vgam. It may refer to frequencies of the individual data or be weight matrices specified beforehand.

Working weights are used by the IRLS algorithm. They correspond to the second derivatives of the log-likelihood function with respect to the linear predictors. The working weights correspond to positive-definite weight matrices and are returned in matrix-band form, e.g., the first $M$ columns correspond to the diagonals, etc.

Value

If type = "working" and deriv = TRUE then a list is returned with the two components described below. Otherwise the prior or working weights are returned depending on the value of type.

deriv  Typically the first derivative of the log-likelihood with respect to the linear predictors. For example, this is the variable deriv.mu in glm.fit(), or equivalently, the matrix returned in the "deriv" slot of a VGAM family function.

weights  The working weights.
Note

This function is intended to be similar to weights.glm (see glm).

Author(s)

Thomas W. Yee

References


See Also

glm, vglmff-class, vglm.

Examples

```r
pneumo <- transform(pneumo, let = log(exposure.time))
(fit <- vglm(cbind(normal, mild, severe) ~ let,
            cumulative(parallel = TRUE, reverse = TRUE), data = pneumo))
depvar(fit)  # These are sample proportions
weights(fit, type = "prior", matrix = FALSE)  # Number of observations

# Look at the working residuals
nn <- nrow(model.matrix(fit, type = "lm"))
M <- ncol(predict(fit))

wwt <- weights(fit, type = "working", deriv = TRUE)  # In matrix-band format
wz <- m2a(wwt$weights, M = M)  # In array format
wzinv <- array(apply(wz, 3, solve), c(M, M, nn))
wresid <- matrix(NA, nn, M)  # Working residuals
for (ii in 1:nn)
  wresid[ii, ] <- wzinv[, ii, drop = TRUE] %*% wz$deriv[ii, ]
max(abs(c(resid(fit, type = "work")) - c(wresid)))  # Should be 0

(zedd <- predict(fit) + wresid)  # Adjusted dependent vector
```

**wine**

*Bitterness in Wine Data*

Description

This oenological data frame concerns the amount of bitterness in 78 bottles of white wine.

Usage

data(wine)
Format

A data frame with 4 rows on the following 7 variables.

- **temp** temperature, with levels cold and warm.
- **contact** whether contact of the juice with the skin was allowed or avoided, for a specified period.
  - Two levels: no or yes.
- **bitter1, bitter2, bitter3, bitter4, bitter5** numeric vectors, the counts. The order is none to most intense.

Details

The data set comes from Randall (1989) and concerns a factorial experiment for investigating factors that affect the bitterness of white wines. There are two factors in the experiment: temperature at the time of crushing the grapes and contact of the juice with the skin. Two bottles of wine were fermented for each of the treatment combinations. A panel of 9 judges were selected and trained for the ability to detect bitterness. Thus there were 72 bottles in total. Originally, the bitterness of the wine were taken on a continuous scale in the interval from 0 (none) to 100 (intense) but later they were grouped using equal lengths into five ordered categories 1, 2, 3, 4 and 5.

Source


Examples

```r
wine
summary(wine)
```

```
wrapup.smart

Cleans Up After Smart Prediction

Description

**wrapup.smart** deletes any variables used by smart prediction. Needed by both the modelling function and the prediction function.

Usage

**wrapup.smart()**
The variables to be deleted are `.smart.prediction`, `.smart.prediction.counter`, and `.smart.prediction.mode`. The function `wrapup.smart` is useful in R because these variables are held in `smartpredenv`.

**See Also**

`setup.smart`.

**Examples**

```r
## Not run: # Place this inside modelling functions such as lm, glm, vglm.
wrapup.smart()  # Put at the end of lm

## End(Not run)
```

### yeo.johnson

**Yeo-Johnson Transformation**

**Description**

Computes the Yeo-Johnson transformation, which is a normalizing transformation.

**Usage**

```r
yeo.johnson(y, lambda, derivative = 0,
             epsilon = sqrt(.Machine$double.eps), inverse = FALSE)
```

**Arguments**

- `y`: Numeric, a vector or matrix.
- `lambda`: Numeric. It is recycled to the same length as `y` if necessary.
- `derivative`: Non-negative integer. The default is the ordinary function evaluation, otherwise the derivative with respect to `lambda`.
- `epsilon`: Numeric and positive value. The tolerance given to values of `lambda` when comparing it to 0 or 2.
- `inverse`: Logical. Return the inverse transformation?

**Details**

The Yeo-Johnson transformation can be thought of as an extension of the Box-Cox transformation. It handles both positive and negative values, whereas the Box-Cox transformation only handles positive values. Both can be used to transform the data so as to improve normality. They can be used to perform LMS quantile regression.

**Value**

The Yeo-Johnson transformation or its inverse, or its derivatives with respect to `lambda`, of `y`. 
Note

If inverse = TRUE then the argument derivative = 0 is required.

Author(s)

Thomas W. Yee

References


See Also

`lms.yjn`, `boxcox`.

Examples

```r
y <- seq(-4, 4, len = (nn <- 200))
lltry <- c(0, 0.5, 1, 1.5, 2)  # Try these values of lambda
lltry <- length(lltry)
psi <- matrix(as.numeric(NA), nn, lltry)
for (ii in 1:lltry)
  psi[, ii] <- yeo.johnson(y, lambda = lltry[ii])

# Not run:
matplot(y, psi, type = "l", ylim = c(-4, 4), lwd = 2, lty = 1:lltry,
  ylab = "Yeo-Johnson transformation", col = 1:lltry, las = 1,
  main = "Yeo-Johnson transformation with some values of lambda")
abline(v = 0, h = 0)
legend(x = 1, y = -0.5, lty = 1:lltry, legend = as.character(lltry),
  lwd = 2, col = 1:lltry)
```

## yip88

### Zero-Inflated Poisson Distribution (Yip (1988) algorithm)

Description

Fits a zero-inflated Poisson distribution based on Yip (1988).

Usage

```r
yip88(link = "loge", n.arg = NULL)
```
Arguments

- **link**: Link function for the usual \( \lambda \) parameter. See [Links](#) for more choices.
- **n.arg**: The total number of observations in the data set. Needed when the response variable has all the zeros deleted from it, so that the number of zeros can be determined.

Details

The method implemented here, Yip (1988), maximizes a *conditional* likelihood. Consequently, the methodology used here deletes the zeros from the data set, and is thus related to the positive Poisson distribution (where \( P(Y = 0) = 0 \)).

The probability function of \( Y \) is 0 with probability \( \phi \), and Poisson(\( \lambda \)) with probability \( 1 - \phi \). Thus

\[
P(Y = 0) = \phi + (1 - \phi)P(W = 0)
\]

where \( W \) is Poisson(\( \lambda \)). The mean, \( (1 - \phi)\lambda \), can be obtained by the extractor function [fitted](#) applied to the object.

This family function treats \( \phi \) as a scalar. If you want to model both \( \phi \) and \( \lambda \) as a function of covariates, try [zipoisson](#).

Value

An object of class "vglmff" (see [vglmff-class](#)). The object is used by modelling functions such as [vglm](#), [rrvglm](#) and [vgam](#).

Warning

Under- or over-flow may occur if the data is ill-conditioned. Yip (1988) only considered \( \phi \) being a scalar and not modelled as a function of covariates. To get around this limitation, try [zipoisson](#).

Inference obtained from [summary.vglm](#) and [summary.vgam](#) may or may not be correct. In particular, the p-values, standard errors and degrees of freedom may need adjustment. Use simulation on artificial data to check that these are reasonable.

Note

The data may be inputted in two ways. The first is when the response is a vector of positive values, with the argument \( n \) in [yip88](#) specifying the total number of observations. The second is simply include all the data in the response. In this case, the zeros are trimmed off during the computation, and the \( x \) and \( y \) slots of the object, if assigned, will reflect this.

The estimate of \( \phi \) is placed in the \( \text{misc} \) slot as \( \text{@misc}$pstr@\). However, this estimate is computed only for intercept models, i.e., the formula is of the form \( y \sim 1 \).

Author(s)

Thomas W. Yee
References


See Also

zipoisson, Zipois, zapoisson, pospoisson, poissonff, dzipois.

Examples

```r
phi <- 0.35; lambda <- 2  # Generate some artificial data
y <- rzipois(n <- 1000, lambda, phi)
table(y)

# Two equivalent ways of fitting the same model
fit1 <- vglm(y ~ 1, yip88(n = length(y)), subset = y > 0)
fit2 <- vglm(y ~ 1, yip88, trace = TRUE, crit = "coef")
(true.mean <- (1-phi) * lambda)
mean(y)
head(fitted(fit1))
fit1@misc$ptr0  # The estimate of phi

# Compare the ZIP with the positive Poisson distribution
pp <- vglm(y ~ 1, pospoisson, subset = y > 0, crit = "c")
coef(pp)
coef(fit1) - coef(pp)  # Same
head(fitted(fit1) - fitted(pp))  # Different

# Another example (Angers and Biswas, 2003) ------------------------
abdata <- data.frame(y = 0:7, w = c(182, 41, 12, 2, 2, 0, 0, 1))
abdata <- subset(abdata, w > 0)

yy <- with(abdata, rep(y, w))
fit3 <- vglm(yy ~ 1, yip88(n = length(yy)), subset = yy > 0)
fit3@misc$ptr0  # Estimate of phi (they get 0.5154 with SE 0.0707)
coef(fit3, matrix = TRUE)
Coef(fit3)  # Estimate of lambda (they get 0.6997 with SE 0.1520)
head(fitted(fit3))
mean(yy)  # Compare this with fitted(fit3)
```
Yules  

Yule-Simon Distribution

Description

Density, distribution function, quantile function and random generation for the Yule-Simon distribution.

Usage

dyules(x, rho, log = FALSE)
pyules(q, rho, log.p = FALSE)
ryules(n, rho)

Arguments

x, q  Vector of quantiles. For the density, it should be a vector with positive integer values in order for the probabilities to be positive.
n  number of observations. Same as in runif.
rho  See yulesimon.
log  logical; if TRUE, the logarithm is returned.
log.p  Same meaning as in pnorm or qnorm.

Details

See yulesimon, the VGAM family function for estimating the parameter, for the formula of the probability density function and other details.

Value

dyules gives the density, pyules gives the distribution function, and ryules generates random deviates.

Author(s)

T. W. Yee

See Also

yulesimon.
Examples

dyules(1:20, 2.1)
ryules(20, 2.1)

round(1000 * dyules(1:8, 2))
table(ryules(1000, 2))

## Not run: x <- 0:6
plot(x, dyules(x, rho = 2.2), type = "h", las = 1, col = "blue")
## End(Not run)

yulesimon

Yule-Simon Family Function

Description

Estimating the parameter of the Yule-Simon distribution.

Usage

yulesimon(link = "loge", irho = NULL, nsimeim = 200, zero = NULL)

Arguments

link Link function for the \( \rho \) parameter. See Links for more choices and for general information.

irho Optional initial value for the (positive) parameter. See CommonVGAMffArguments for more information. The default is to obtain an initial value internally. Use this argument if the default fails.

nsimeim, zero See CommonVGAMffArguments for more information.

Details

The probability function is

\[
  f(y; \rho) = \rho \ast \text{beta}(y, \rho + 1),
\]

where the parameter \( \rho > 0 \), \text{beta} is the \text{beta} function, and \( y = 1, 2, \ldots \). The function \text{dyules} computes this probability function. The mean of \( Y \), which is returned as fitted values, is \( \rho/(\rho - 1) \) provided \( \rho > 1 \). The variance of \( Y \) is \( \rho^2/((\rho - 1)^2(\rho - 2)) \) provided \( \rho > 2 \).

The distribution was named after Udny Yule and Herbert A. Simon. Simon originally called it the Yule distribution. This family function can handle multiple responses.

Value

An object of class "vglmff" (see \text{vglmff-class}). The object is used by modelling functions such as \text{vglm} and \text{vgam}.
Zabinom

Author(s)

T. W. Yee

References


See Also

ryules, simulate.vlm.

Examples

```r
ydata <- data.frame(x2 = runif(nn <- 1000))
ydata <- transform(ydata, y = ryules(nn, rho = exp(1.5 - x2)))
with(ydata, table(y))
fit <- vglm(y ~ x2, yulesimon, data = ydata, trace = TRUE)
coef(fit, matrix = TRUE)
summary(fit)
```

Zabinom

**Zero-Altered Binomial Distribution**

Description

Density, distribution function, quantile function and random generation for the zero-altered binomial distribution with parameter `pobs`.

Usage

```r
dzabinom(x, size, prob, pobs = 0, log = FALSE)
pzabinom(q, size, prob, pobs = 0)
qzabinom(p, size, prob, pobs = 0)
rzabinom(n, size, prob, pobs = 0)
```

Arguments

- `x, q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If `length(n) > 1` then the length is taken to be the number required.
- `size, prob, log` Parameters from the ordinary binomial distribution (see `dbinom`).
- `pobs` Probability of (an observed) zero, called `pobs`. The default value of `pobs = 0` corresponds to the response having a positive binomial distribution.
Details

The probability function of $Y$ is 0 with probability $p_{obs0}$, else a positive binomial(size, prob) distribution.

Value

dzabinom gives the density and pzabinom gives the distribution function, qzabinom gives the quantile function, and rzabinom generates random deviates.

Note

The argument $p_{obs0}$ is recycled to the required length, and must have values which lie in the interval $[0, 1]$.

Author(s)

T. W. Yee

See Also

zabinomial, rposbinom.

Examples

```r
size <- 10; prob <- 0.15; pobs0 <- 0.05; x <- (-1):7
dzabinom(x, size = size, prob = prob, pobs0 = pobs0)
table(rzabinom(100, size = size, prob = prob, pobs0 = pobs0))

## Not run: x <- 0:10
barplot(rbind(dzabinom(x, size = size, prob = prob, pobs0 = pobs0),
              dbinom(x, size = size, prob = prob)),
        beside = TRUE, col = c("blue", "orange"), cex.main = 0.7, las = 1,
        ylab = "Probability", names.arg = as.character(x),
        main = paste("ZAB(size = ", size, ", prob = ", prob, ", pobs0 = ", pobs0,
                      ") [blue] vs", " Binom(size = ", size, ", prob = ", prob,
                      ") [orange] densities", sep = ")")

## End(Not run)
```

Zero-Altered Binomial Distribution

Description

Fits a zero-altered binomial distribution based on a conditional model involving a Bernoulli distribution and a positive-binomial distribution.
Usage

```r
zabinomial(lpobs0 = "logit", lprob = "logit",
   type.fitted = c("mean", "pobs0"),
   ipobs0 = NULL, iprob = NULL, imethod = 1, zero = NULL)
zabinomialff(lprob = "logit", lonempobs0 = "logit",
   type.fitted = c("mean", "pobs0", "onempobs0"),
   iprob = NULL, lonempobs0 = NULL, imethod = 1, zero = 2)
```

Arguments

- `lprob`: Parameter link function applied to the probability parameter of the binomial distribution. See Links for more choices.
- `lpobs0`: Link function for the parameter \( p_0 \), called \( pobs0 \) here. See Links for more choices.
- `type.fitted`: See CommonVGAMffArguments and fittedvlm for information.
- `iprob`, `ipobs0`: See CommonVGAMffArguments.
- `lonempobs0`, `lonempobs0`: Corresponding argument for the other parameterization. See details below.
- `imethod`, `zero`: See CommonVGAMffArguments.

Details

The response \( Y \) is zero with probability \( p_0 \), else \( Y \) has a positive-binomial distribution with probability \( 1 - p_0 \). Thus \( 0 < p_0 < 1 \), which may be modelled as a function of the covariates. The zero-altered binomial distribution differs from the zero-inflated binomial distribution in that the former has zeros coming from one source, whereas the latter has zeros coming from the binomial distribution too. The zero-inflated binomial distribution is implemented in zabinomial. Some people call the zero-altered binomial a hurdle model.

The input is currently a vector or one-column matrix. By default, the two linear/additive predictors for zabinomial() are \((\text{logit}(p_0), \log(p))\)^T

The VGAM family function zabinomialff() has a few changes compared to zabinomial(). These are: (i) the order of the linear/additive predictors is switched so the binomial probability comes first; (ii) argument onempobs0 is now 1 minus the probability of an observed 0, i.e., the probability of the positive binomial distribution, i.e., onempobs0 is \( 1 - pobs0 \); (iii) argument zero has a new default so that the onempobs0 is intercept-only by default. Now zabinomialff() is generally recommended over zabinomial(). Both functions implement Fisher scoring and neither can handle multiple responses.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

The fitted.values slot of the fitted object, which should be extracted by the generic function fitted, returns the mean \( \mu \) (default) which is given by

\[
\mu = (1 - p_0) \mu_b / [1 - (1 - \mu_b)^N]
\]

where \( \mu_b \) is the usual binomial mean. If `type.fitted = "pobs0"` then \( p_0 \) is returned.
Note
The response should be a two-column matrix of counts, with first column giving the number of successes.

Note this family function allows $p_0$ to be modelled as functions of the covariates by having zero = NULL. It is a conditional model, not a mixture model.

These family functions effectively combine posbinomial and binomialff into one family function.

Author(s)
T. W. Yee

See Also
dzabinom, zibinomial, posbinomial, binomialff, dbinom, CommonVGAMffArguments.

Examples
```r
zdata <- data.frame(x2 = runif(nn <- 1000))
zdata <- transform(zdata, size = 10, prob = logit(-2 + 3*x2, inverse = TRUE), pobs0 = logit(-1 + 2*x2, inverse = TRUE))
zdata <- transform(zdata, y1 = rzabinom(nn, size = size, prob = prob, pobs0 = pobs0))
with(zdata, table(y1))

fit <- vglm(cbind(y1, size - y1) ~ x2, zabinomial(zero = NULL), data = zdata, trace = TRUE)
coef(fit, matrix = TRUE)
head(fitted(fit))
head(predict(fit))
summary(fit)
```

---

Zageom

Zero-Altered Geometric Distribution

Description
Density, distribution function, quantile function and random generation for the zero-altered geometric distribution with parameter pobs0.

Usage
```r
dzageom(x, prob, pobs0 = 0, log = FALSE)
pzageom(q, prob, pobs0 = 0)
qzageom(p, prob, pobs0 = 0)
rzageom(n, prob, pobs0 = 0)
```
Arguments

\( x, q \)  \hspace{1cm} \text{vector of quantiles.}

\( p \)  \hspace{1cm} \text{vector of probabilities.}

\( n \)  \hspace{1cm} \text{number of observations. If length}(n) > 1 \text{ then the length is taken to be the number required.}

\( \text{prob, log} \)  \hspace{1cm} \text{Parameters from the ordinary geometric distribution (see dgeom).}

\( \text{pobs} \)  \hspace{1cm} \text{Probability of (an observed) zero, called pobs}. \text{ The default value of pobs} = 0 \text{ corresponds to the response having a positive geometric distribution.}

Details

The probability function of \( Y \) is 0 with probability \( \text{pobs} \), else a positive geometric(prob) distribution.

Value

dzageom gives the density and pzageom gives the distribution function, qzageom gives the quantile function, and rzageom generates random deviates.

Note

The argument \( pobs \) is recycled to the required length, and must have values which lie in the interval \([0, 1]\).

Author(s)

T. W. Yee

See Also

zageometric, zigeometric, rposgeom.

Examples

\[
\begin{aligned}
\text{prob} & \leftarrow 0.35; \text{pobs} \leftarrow 0.05; x \leftarrow (-1):7 \\
\text{dzageom}(x, \text{prob} = \text{prob}, \text{pobs} = \text{pobs}) \\
\text{table}(\text{rzageom}(100, \text{prob} = \text{prob}, \text{pobs} = \text{pobs}))
\end{aligned}
\]

\[
\begin{aligned}
\text{## Not run:} & \quad x \leftarrow 0:10 \\
\text{barplot}(\text{rbind}(\text{dzageom}(x, \text{prob} = \text{prob}, \text{pobs} = \text{pobs}), \\
\text{dgeom}(x, \text{prob} = \text{prob})), \\
\text{beside = TRUE, col = c(“blue”, “orange”), cex.main = 0.7, las = 1,} \\
\text{ylab = ”Probability”, names.arg = as.character(x),} \\
\text{main = paste(”ZAG(prob = ”, \text{prob}, “, pobs = ”, \text{pobs}, “) [blue] vs”, “ Geometric(prob = ”, \text{prob}, “) [orange] densities”, sep = “”))}
\end{aligned}
\]

\[
\text{## End(Not run)}
\]


Description

Fits a zero-altered geometric distribution based on a conditional model involving a Bernoulli distribution and a positive-geometric distribution.

Usage

```r
zageometric(lpobs0 = "logit", lprob = "logit",
  type.fitted = c("mean", "pobs0", "onempobs0"),
  imethod = 1, ipobs0 = NULL, iprob = NULL, zero = NULL)
zageometricff(lprob = "logit", lonempobs0 = "logit",
  type.fitted = c("mean", "pobs0", "onempobs0"),
  imethod = 1, iprob = NULL, ionempobs0 = NULL, zero = -2)
```

Arguments

- `lpobs0`: Link function for the parameter $p_0$ or $\phi$, called `pobs0` or `phi` here. See Links for more choices.
- `lprob`: Parameter link function applied to the probability of success, called `prob` or $p$. See Links for more choices.
- `type.fitted`: See CommonVGAMffArguments and fittedvlm for information.
- `ipobs0`, `iprob`: Optional initial values for the parameters. If given, they must be in range. For multi-column responses, these are recycled sideways.
- `lonempobs0`, `ionempobs0`: Corresponding argument for the other parameterization. See details below.
- `zero`, `imethod`: See CommonVGAMffArguments.

Details

The response $Y$ is zero with probability $p_0$, or $Y$ has a positive-geometric distribution with probability $1 - p_0$. Thus $0 < p_0 < 1$, which is modelled as a function of the covariates. The zero-altered geometric distribution differs from the zero-inflated geometric distribution in that the former has zeros coming from one source, whereas the latter has zeros coming from the geometric distribution too. The zero-inflated geometric distribution is implemented in the VGAM package. Some people call the zero-altered geometric a hurdle model.

The input can be a matrix (multiple responses). By default, the two linear/additive predictors of `zageometric` are $(\logit(\phi), \logit(p))^T$.

The VGAM family function `zageometricff()` has a few changes compared to `zageometric()`. These are: (i) the order of the linear/additive predictors is switched so the geometric probability comes first; (ii) argument `onempobs0` is now 1 minus the probability of an observed 0, i.e., the probability of the positive geometric distribution, i.e., `onempobs0` is $1 - pobs0$; (iii) argument `zero`
has a new default so that \( \text{pobs0} \) is intercept-only by default. Now \texttt{zageometric}() is generally recommended over \texttt{zageometric}(). Both functions implement Fisher scoring and can handle multiple responses.

**Value**

An object of class "\texttt{vglmff}" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, and \texttt{vgam}.

The \texttt{fitted.values} slot of the fitted object, which should be extracted by the generic function \texttt{fitted}, returns the mean \( \mu \) (default) which is given by

\[
\mu = (1 - \phi)/p.
\]

If \texttt{type.fitted = "pobs0"} then \( p_0 \) is returned.

**Warning**

Convergence for this \texttt{VGAM} family function seems to depend quite strongly on providing good initial values.

Inference obtained from \texttt{summary.vglm} and \texttt{summary.vgam} may or may not be correct. In particular, the \( p \)-values, standard errors and degrees of freedom may need adjustment. Use simulation on artificial data to check that these are reasonable.

**Note**

Note this family function allows \( p_0 \) to be modelled as functions of the covariates. It is a conditional model, not a mixture model.

This family function effectively combines \texttt{binomialff} and \texttt{posgeometric}() and \texttt{geometric} into one family function. However, \texttt{posgeometric}() is not written because it is trivially related to \texttt{geometric}.

**Author(s)**

T. W. Yee

**See Also**

\texttt{dzageom, geometric, zigeometric, dgeom, CommonVGAMffArguments, simulate.vlm}.

**Examples**

```r
zdata <- data.frame(x2 = runif(nn <- 1000))
zdata <- transform(zdata, pobs0 = logit(-1 + 2*x2, inverse = TRUE),
                    prob = logit(-2 + 3*x2, inverse = TRUE))
zdata <- transform(zdata, y1 = rzageom(nn, prob = prob, pobs0 = pobs0),
                    y2 = rzageom(nn, prob = prob, pobs0 = pobs0))
with(zdata, table(y1))

fit <- vglm(cbind(y1, y2) ~ x2, zageometric, data = zdata, trace = TRUE)
coef(fit, matrix = TRUE)
```
Zanegbin  

Zero-Altered Negative Binomial Distribution

Description

Density, distribution function, quantile function and random generation for the zero-altered negative binomial distribution with parameter pobs0.

Usage

dzanegbin(x, size, prob = NULL, munb = NULL, pobs0 = 0, log = FALSE)
pzanegbin(q, size, prob = NULL, munb = NULL, pobs0 = 0)
qzanegbin(p, size, prob = NULL, munb = NULL, pobs0 = 0)
rzanegbin(n, size, prob = NULL, munb = NULL, pobs0 = 0)

Arguments

x, q  vector of quantiles.
p  vector of probabilities.
n  number of observations. If length(n) > 1 then the length is taken to be the number required.
size, prob, munb, log  Parameters from the ordinary negative binomial distribution (see dnbinom). Some arguments have been renamed slightly.
pobs0  Probability of zero, called pobs0. The default value of pobs0 = 0 corresponds to the response having a positive negative binomial distribution.

Details

The probability function of Y is 0 with probability pobs0, else a positive negative binomial(µnb, size) distribution.

Value

dzanegbin gives the density and pzanegbin gives the distribution function, qzanegbin gives the quantile function, and rzanegbin generates random deviates.

Note

The argument pobs0 is recycled to the required length, and must have values which lie in the interval [0, 1].
Zero-Altered Negative Binomial Distribution

Description

Fits a zero-altered negative binomial distribution based on a conditional model involving a binomial distribution and a positive-negative binomial distribution.

Usage

\[
\text{zanegbinomial}(l\text{pobs0} = \text{"logit"}, l\text{munb} = \text{"loge"}, l\text{size} = \text{"loge"}, \\
\text{type.fitted} = \text{c("mean", "pobs0")}, \\
\text{ipobs0} = \text{NULL}, isize = \text{NULL}, zero = -3, \text{imethod} = 1, \\
\text{nsimEIM} = 250, \text{ishrinkage} = 0.95)
\]

\[
\text{zanegbinomialff}(l\text{munb} = \text{"loge"}, l\text{size} = \text{"loge"}, l\text{onempobs0} = \text{"logit"}, \\
\text{type.fitted} = \text{c("mean", "pobs0", "onempobs0")}, \\
isize = \text{NULL}, l\text{onempobs0} = \text{NULL}, zero = \text{c(-2, -3)}, \\
imethod = 1, \text{nsimEIM} = 250, \text{ishrinkage} = 0.95)
\]

Arguments

\text{l\text{pobs0}} \quad \text{Link function for the parameter } p_0, \text{called pobs0 here. See LINKS for more choices.}
lmunb  Link function applied to the munb parameter, which is the mean $\mu_{nb}$ of an ordinary negative binomial distribution. See \texttt{Links} for more choices.

lsize  Parameter link function applied to the reciprocal of the dispersion parameter, called $k$. That is, as $k$ increases, the variance of the response decreases. See \texttt{Links} for more choices.

type.fitted  See \texttt{CommonVGAMffArguments} and \texttt{fittedvlm} for information.

lonempobs0, ionempobs0  Corresponding argument for the other parameterization. See details below.

ipobs0, isize  Optional initial values for $p_0$ and $k$. If given then it is okay to give one value for each response/species by inputting a vector whose length is the number of columns of the response matrix.

zero  Specifies which of the three linear predictors are modelled as an intercept only. All parameters can be modelled as a function of the explanatory variables by setting zero = \texttt{NULL} (not recommended). A negative value means that the value is recycled, e.g., setting $-3$ means all $k$ are intercept-only for zanegbinomial.

nsimEIM, imethod  See \texttt{CommonVGAMffArguments}.

ishrinkage  See \texttt{negbinomial} and \texttt{CommonVGAMffArguments}.

\textbf{Details}

The response $Y$ is zero with probability $p_0$, or $Y$ has a positive-negative binomial distribution with probability $1 - p_0$. Thus $0 < p_0 < 1$, which is modelled as a function of the covariates. The zero-altered negative binomial distribution differs from the zero-inflated negative binomial distribution in that the former has zeros coming from one source, whereas the latter has zeros coming from the negative binomial distribution too. The zero-inflated negative binomial distribution is implemented in the \texttt{VGAM} package. Some people call the zero-altered negative binomial a \textit{hurdle} model.

For one response/species, by default, the three linear/additive predictors for \texttt{zanegbinomial()} are $(\text{logit}(p_0), \log(\mu_{nb}), \log(k))^T$. This vector is recycled for multiple species.

The \texttt{VGAM} family function \texttt{zanegbinomialff()} has a few changes compared to \texttt{zanegbinomial()}. These are: (i) the order of the linear/additive predictors is switched so the negative binomial mean comes first; (ii) argument \texttt{onempobs0} is now 1 minus the probability of an observed 0, i.e., the probability of the positive negative binomial distribution, i.e., \texttt{onempobs0} is $1 - \text{pobs0}$; (iii) argument \texttt{zero} has a new default so that the \texttt{pobs0} is intercept-only by default. Now \texttt{zanegbinomialff()} is generally recommended over \texttt{zanegbinomial()}.

Both functions implement Fisher scoring and can handle multiple responses.

\textbf{Value}

An object of class "\texttt{vglmff}" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm}, and \texttt{vgam}.

The \texttt{fitted.values} slot of the fitted object, which should be extracted by the generic function \texttt{fitted}, returns the mean $\mu$ (default) which is given by

$$\mu = (1 - p_0)\mu_{nb}/[1 - (k/(k + \mu_{nb}))^k].$$

If \texttt{type.fitted = \texttt{"pobs0"}} then $p_0$ is returned.
Warning

Convergence for this VGAM family function seems to depend quite strongly on providing good initial values.

This VGAM family function is computationally expensive and usually runs slowly; setting trace = TRUE is useful for monitoring convergence.

Inference obtained from summary.vglm and summary.vgam may or may not be correct. In particular, the p-values, standard errors and degrees of freedom may need adjustment. Use simulation on artificial data to check that these are reasonable.

Note

Note this family function allows \( p_0 \) to be modelled as functions of the covariates provided zero is set correctly. It is a conditional model, not a mixture model. Simulated Fisher scoring is the algorithm.

This family function effectively combines posnegbinomial and binomialff into one family function.

This family function can handle a multivariate response, e.g., more than one species.

Author(s)

T. W. Yee

References


See Also

dzanegbin, posnegbinomial, negbinomial, binomialff, rposnegbin, zinegbinomial, zipoisson, dnbinom, CommonVGAMffArguments, simulate.vlm.

Examples

```r
## Not run:
zdata <- data.frame(x2 = runif(nn <- 2000))
zdata <- transform(zdata, pobs0 = logit(-1 + 2*x2, inverse = TRUE))
zdata <- transform(zdata,
  y1 = rzanegbin(nn, munb = exp(0+2*x2), size = exp(1), pobs0 = pobs0),
  y2 = rzanegbin(nn, munb = exp(1+2*x2), size = exp(1), pobs0 = pobs0))
with(zdata, table(y1))
with(zdata, table(y2))

fit <- vglm(cbind(y1, y2) ~ x2, zanegbinomial, data = zdata, trace = TRUE)
coef(fit, matrix = TRUE)
```
Zero-Altered Poisson Distribution

Description

Density, distribution function, quantile function and random generation for the zero-altered Poisson distribution with parameter \( p\text{obs} \).

Usage

- \( \text{dzapois}(x, \lambda, p\text{obs} = 0, \text{log} = \text{FALSE}) \)
- \( \text{pzapois}(q, \lambda, p\text{obs} = 0) \)
- \( \text{qzapois}(p, \lambda, p\text{obs} = 0) \)
- \( \text{rzapois}(n, \lambda, p\text{obs} = 0) \)

Arguments

- \( x, q \): vector of quantiles.
- \( p \): vector of probabilities.
- \( n \): number of observations. If \( \text{length}(n) > 1 \) then the length is taken to be the number required.
- \( \lambda \): Vector of positive means.
- \( p\text{obs} \): Probability of zero, called \( p\text{obs} \). The default value of \( p\text{obs} = 0 \) corresponds to the response having a positive Poisson distribution.
- \( \text{log} \): Logical. Return the logarithm of the answer?

Details

The probability function of \( Y \) is 0 with probability \( p\text{obs} \), else a positive \( \text{Poisson}(\lambda) \).

Value

- \( \text{dzapois} \) gives the density, \( \text{pzapois} \) gives the distribution function, \( \text{qzapois} \) gives the quantile function, and \( \text{rzapois} \) generates random deviates.

Note

The argument \( p\text{obs} \) is recycled to the required length, and must have values which lie in the interval \([0, 1]\).
zapoison

Zero-Altered Poisson Distribution

Description

Fits a zero-altered Poisson distribution based on a conditional model involving a Bernoulli distribution and a positive-Poisson distribution.

Usage

zapoison(lpobs0 = "logit", llambda = "loge", type.fitted = c("mean", "pobs0", "onempobs0"), zero = NULL)
zapoisononff(llambda = "loge", lonempobs0 = "logit", type.fitted = c("mean", "pobs0", "onempobs0"), zero = -2)

Arguments

lpobs0 Link function for the parameter $p_0$, called `pobs0` here. See Links for more choices.
llambda Link function for the usual $\lambda$ parameter. See Links for more choices.
type.fitted See CommonVGAMffArguments and fittedvlm for information.
lonempobs0 Corresponding argument for the other parameterization. See details below.
zero See CommonVGAMffArguments for more information.
Details

The response $Y$ is zero with probability $p_0$, else $Y$ has a positive-Poisson($\lambda$) distribution with probability $1 - p_0$. Thus $0 < p_0 < 1$, which is modelled as a function of the covariates. The zero-altered Poisson distribution differs from the zero-inflated Poisson distribution in that the former has zeros coming from one source, whereas the latter has zeros coming from the Poisson distribution too. Some people call the zero-altered Poisson a hurdle model.

For one response/species, by default, the two linear/additive predictors for zapoisson() are $(\logit(p_0), \log(\lambda))^T$.

The VGAM family function zapoissonff() has a few changes compared to zapoisson(). These are: (i) the order of the linear/additive predictors is switched so the Poisson mean comes first; (ii) argument onempobs0 is now 1 minus the probability of an observed 0, i.e., the probability of the positive Poisson distribution, i.e., onempobs0 is $1 - \text{pobs0}$; (iii) argument zero has a new default so that the onempobs0 is intercept-only by default. Now zapoissonff() is generally recommended over zapoisson(). Both functions implement Fisher scoring and can handle multiple responses.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

The fitted.values slot of the fitted object, which should be extracted by the generic function fitted, returns the mean $\mu$ (default) which is given by

$$\mu = (1 - p_0)\lambda/[1 - \exp(-\lambda)].$$

If type.fitted = "pobs0" then $p_0$ is returned.

Note

There are subtle differences between this family function and zipoisson and yip88. In particular, zipoisson is a mixture model whereas zapoisson() and yip88 are conditional models.

Note this family function allows $p_0$ to be modelled as functions of the covariates.

This family function effectively combines pospoisson and binomialff into one family function. This family function can handle a multivariate response, e.g., more than one species.

Author(s)

T. W. Yee

References


See Also
rzapois, zipoisson, pospoisson, posnegbinomial, binomialff, rpospois, CommonVGAMffArguments, simulate.vlm.

Examples

zdata <- data.frame(x2 = runif(nn <- 1000))
zdata <- transform(zdata, pobs0 = logit(-1 + 1*x2, inverse = TRUE),
lambda = loge(-0.5 + 2*x2, inverse = TRUE))
zdata <- transform(zdata, y = rzapois(nn, lambda, pobs0 = pobs0))

with(zdata, table(y))
fit <- vglm(y ~ x2, zapoisson, data = zdata, trace = TRUE)
fit <- vglm(y ~ x2, zapoisson, data = zdata, trace = TRUE, crit = "coef")
head(fitted(fit))
head(predict(fit))
coef(fit, matrix = TRUE)
summary(fit)

# Another example -------------------------------
# Data from Angers and Biswas (2003)
abdata <- data.frame(y = 0:7, w = c(182, 41, 12, 2, 2, 0, 0, 1))
abdata <- subset(abdata, w > 0)
Abdata <- data.frame(yy = with(abdata, rep(y, w)))
fit3 <- vglm(yy ~ 1, zapoisson, data = Abdata, trace = TRUE, crit = "coef")
coef(fit3, matrix = TRUE)
Coef(fit3) # Estimate lambda (they get 0.6997 with SE 0.1520)
head(fitted(fit3), 1)
with(Abdata, mean(yy)) # Compare this with fitted(fit3)

zero

The zero Argument in VGAM Family Functions

Description

The zero argument allows users to conveniently model certain linear/additive predictors as intercepts only.

Details

Often a certain parameter needs to be modelled simply while other parameters in the model may be more complex, for example, the $\lambda$ parameter in LMS-Box-Cox quantile regression should be modelled more simply compared to its $\mu$ parameter. Another example is the $\xi$ parameter in a GEV distribution which should be modelled simpler than its $\mu$ parameter. Using the zero argument allows this to be fitted conveniently without having to input all the constraint matrices explicitly.

The zero argument should be assigned an integer vector from the set \{1:M\} where M is the number of linear/additive predictors. Full details about constraint matrices can be found in the references.
Value

Nothing is returned. It is simply a convenient argument for constraining certain linear/additive predictors to be an intercept only.

Warning

The use of other arguments may conflict with the zero argument. For example, using constraints to input constraint matrices may conflict with the zero argument. Another example is the argument parallel. In general users should not assume any particular order of precedence when there is potential conflict of definition. Currently no checking for consistency is made.

The argument zero may be renamed in the future to something better.

Side Effects

The argument creates the appropriate constraint matrices internally.

Note

In all VGAM family functions zero = NULL means none of the linear/additive predictors are modelled as intercepts-only. Almost all VGAM family functions have zero = NULL as the default, but there are some exceptions, e.g., binom2.or.

Typing something like coef(fit, matrix = TRUE) is a useful way to ensure that the zero argument has worked as expected.

Author(s)

T. W. Yee

References


See Also

constraints.

Examples

args(multinomial)
args(binom2.or)
args(gpd)

#LMS quantile regression example
fit <- vglm(BMI ~ sm.bs(age, df = 4), lms.bcg(zero = c(1,3)),
            data = bmi.nz, trace = TRUE)
coef(fit, matrix = TRUE)
Description

Density for the zeta distribution.

Usage

dzeta(x, p, log = FALSE)

Arguments

x  Numerical vector/matrix to evaluate the density.
p  The parameter \( p \). This must be positive.
log Logical. If \( \log = \text{TRUE} \) then the logarithm of the density is returned.

Details

The density function of the zeta distribution is given by

\[ y^{-p-1}/\zeta(p + 1) \]

where \( p > 0 \), \( y = 1, 2, \ldots \), and \( \zeta \) is Riemann’s zeta function.

Value

Returns the density evaluated at \( x \).

Warning

This function has not been fully tested.

Note

The VGAM family function \texttt{zetaff} estimates the parameter \( p \).

Author(s)

T. W. Yee

References


See Also

\texttt{zeta}, \texttt{zetaff}. 

Examples

dzeta(1:20, p = 2)
## Not run: plot(1:6, dzeta(1:6, p = 4), type = "h", las = 1, ylab = "Probability",
   main = "zeta probability function; black: p = 4; blue: p = 2")
points(0.10 + 1:6, dzeta(1:6, p = 2), type = "h", col = "blue")
## End(Not run)

zeta

Riemann’s Zeta Function

Description

Computes Riemann’s zeta function and its first two derivatives.

Usage

zeta(x, deriv = 0)

Arguments

x A complex-valued vector/matrix whose real values must be ≥ 1. Otherwise, if
   x may be real. If deriv is 1 or 2 then x must be real and positive.

deriv An integer equalling 0 or 1 or 2, which is the order of the derivative. The default
   means it is computed ordinarily.

Details

While the usual definition involves an infinite series, more efficient methods have been devised to
compute the value. In particular, this function uses Euler-Maclaurin summation. Theoretically, the
zeta function can be computed over the whole complex plane because of analytic continuation.

The formula used here for analytic continuation is

$$\zeta(s) = 2^s\pi^{s-1}\sin(\pi s/2)\Gamma(1-s)\zeta(1-s).$$

This is actually one of several formulas, but this one was discovered by Riemann himself and is
called the functional equation.

Value

A vector/matrix of computed values.

Warning

This function has not been fully tested, especially the derivatives. In particular, analytic continuation
does not work here for complex x with \Re(x)<1 because currently the gamma function does not handle complex arguments.
Note

Estimation of the parameter of the zeta distribution can be achieved with `zetaff`.

Author(s)

T. W. Yee, with the help of Garry J. Tee.

References


See Also

`zetaff, lerch, gamma`.

Examples

```r
zeta(2:10)

## Not run:
curve(zeta, -13, 0.8, xlim = c(-12, 10), ylim = c(-1, 4), col = "orange",
     las = 1, main = expression(zeta(x)))
curve(zeta, 1.2, 12, add = TRUE, col = "orange")
abline(v = 0, h = c(0, 1), lty = "dashed", col = "gray")

# Close up plot:
curve(zeta, -14, -0.4, col = "orange", main = expression(zeta(x)))
abline(v = 0, h = 0, lty = "dashed", col = "gray")

x <- seq(0.04, 0.8, len = 100) # Plot of the first derivative
plot(x, zeta(x, deriv = 1), type = "l", las = 1, col = "blue",
     xlim = c(0.04, 3), ylim = c(-6, 0), main = "zeta'(x)"

x <- seq(1.2, 3, len = 100)
lines(x, zeta(x, deriv = 1), col = "blue")
abline(v = 0, h = 0, lty = "dashed", col = "gray")

## End(Not run)

zeta(2) - pi^2 / 6 # Should be zero
zeta(4) - pi^4 / 90 # Should be zero
zeta(6) - pi^6 / 945 # Should be 0
zeta(8) - pi^8 / 9450 # Should be 0
# zeta(0, deriv = 1) + 0.5 * log(2*pi) # Should be 0
```
Description

Estimates the parameter of the zeta distribution.

Usage

zetaff(link = "log", init.p = NULL, zero = NULL)

Arguments

link, init.p, zero

See CommonVGAMffArguments for more information. These arguments apply to the (positive) parameter \( p \). See Links for more choices. Choosing loglog constrains \( p > 1 \), but may fail if the maximum likelihood estimate is less than one.

Details

In this long tailed distribution the response must be a positive integer. The probability function for a response \( Y \) is

\[
P(Y = y) = 1/[y^{p+1}\zeta(p + 1)], \quad p > 0, \quad y = 1, 2, ...
\]

where \( \zeta \) is Riemann’s zeta function. The parameter \( p \) is positive, therefore a log link is the default. The mean of \( Y \) is \( \mu = \zeta(p)/\zeta(p + 1) \) (provided \( p > 1 \)) and these are the fitted values. The variance of \( Y \) is \( \zeta(p - 1)/\zeta(p + 1) - \mu^2 \) provided \( p > 2 \).

It appears that good initial values are needed for successful convergence. If convergence is not obtained, try several values ranging from values near 0 to values about 10 or more.

Multiple responses are handled.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.

Note

The zeta function may be used to compute values of the zeta function.

Author(s)

T. W. Yee
References


See Also

*zeta*, *dzeta*, *hzeta*, *zipf*.

Examples

```r
data <- data.frame(y = 1:5, w = c(63, 14, 5, 1, 2))  # Knight, p.304
fit <- vglm(y ~ 1, zetaff, data = zdata, trace = TRUE, weight = w, crit = "coef")
(phat <- Coef(fit))  # 1.682557
with(data, cbind(round(dzeta(y, phat) * sum(w), 1), w))

with(data, weighted.mean(y, w))
fitted(fit, matrix = FALSE)
predict(fit)

# The following should be zero at the MLE:
with(data, mean(log(rep(y, w))) + zeta(1+phat, deriv = 1) / (1+phat))
```

---

**Zibinom**

*Zero-Inflated Binomial Distribution*

**Description**

Density, distribution function, quantile function and random generation for the zero-inflated binomial distribution with parameter `pstr`.

**Usage**

```r
dzibinom(x, size, prob, pstr0 = 0, log = FALSE)
pzibinom(q, size, prob, pstr0 = 0, lower.tail = TRUE, log.p = FALSE)
qzibinom(p, size, prob, pstr0 = 0, lower.tail = TRUE, log.p = FALSE)
rzibinom(n, size, prob, pstr0 = 0)
```

**Arguments**

- `x, q` vector of quantiles.
- `p` vector of probabilities.
- `size` number of trials. It is the $N$ symbol in the formula given in `zibinomial`.
- `prob` probability of success on each trial.
- `n` Same as in `runif`.
- `log, log.p, lower.tail` Arguments that are passed on to `pbinom`.
pstr0  Probability of a structural zero (i.e., ignoring the binomial distribution), called φ. The default value of φ = 0 corresponds to the response having an ordinary binomial distribution.

Details
The probability function of Y is 0 with probability φ, and Binomial(size, prob) with probability 1 − φ. Thus

\[ P(Y = 0) = φ + (1 − φ)P(W = 0) \]

where W is distributed Binomial(size, prob).

Value
dzibinom gives the density, pzibinom gives the distribution function, qzibinom gives the quantile function, and rzibinom generates random deviates.

Note
The argument pstr0 is recycled to the required length, and must have values which lie in the interval [0, 1].
These functions actually allow for zero-deflation. That is, the resulting probability of a zero count is less than the nominal value of the parent distribution. See Zipois for more information.

Author(s)
T. W. Yee

See Also
zibinomial, dbinom.

Examples
```r
prob <- 0.2; size <- 10; pstr0 <- 0.5
(ii <- dzibinom(0:size, size, prob, pstr0 = pstr0))
max(abs(cumsum(ii) - pzibinom(0:size, size, prob, pstr0 = pstr0))) # Should be 0
table(rzibinom(100, size, prob, pstr0 = pstr0))

table(qzibinom(runif(100), size, prob, pstr0 = pstr0))
round(dzibinom(0:10, size, prob, pstr0 = pstr0) * 100) # Should be similar
```

## Not run: x <- 0:size
```r
barplot(rbind(dzibinom(x, size, prob, pstr0 = pstr0),
          dbinom(x, size, prob)),
        beside = TRUE, col = c("blue", "green"), ylab = "Probability",
        main = paste("ZIB0", size, ", ", prob, ", pstr0 = ", pstr0, ") (blue) vs",
        " Binomial(", size, ", ", prob, ") (green)", sep=""),
        names.arg = as.character(x), las = 1, lwd = 2)
```
## End(Not run)
**zibinomial**

**Zero-Inflated Binomial Distribution Family Function**

**Description**

Fits a zero-inflated binomial distribution by maximum likelihood estimation.

**Usage**

```r
zibinomial(lpstr0 = "logit", lprob = "logit",
    type.fitted = c("mean", "pobs0", "pstr0", "onempstr0"),
    ipstr0 = NULL, zero = NULL, multiple.responses = FALSE, imethod = 1)
```

```r
zibinomialff(lprob = "logit", lonempstr0 = "logit",
    type.fitted = c("mean", "pobs0", "pstr0", "onempstr0"),
    lonempstr0 = NULL, zero = 2, multiple.responses = FALSE,
    imethod = 1)
```

**Arguments**

- `lpstr0`, `lprob` Link functions for the parameter $\phi$ and the usual binomial probability $\mu$ parameter. See Links for more choices. For the zero-deflated model see below.
- `type.fitted` See CommonVGAMffArguments and fittedvlm.
- `ipstr0` Optional initial values for $\phi$, whose values must lie between 0 and 1. The default is to compute an initial value internally. If a vector then recycling is used.
- `lonempstr0`, `ionempstr0` Corresponding arguments for the other parameterization. See details below.
- `multiple.responses` Logical. Currently it must be FALSE to mean the function does not handle multivariate responses. This is to remain compatible with the same argument in binomialff.
- `zero`, `imethod` See CommonVGAMffArguments for information. Argument zero has changed its default value for version 0.9-2.

**Details**

These functions are based on

$$P(Y = 0) = \phi + (1 - \phi)(1 - \mu)^N,$$

defining $P(Y = y)$ for $y = 0$, and

$$P(Y = y) = (1 - \phi) \binom{N}{Ny} \mu^{Ny}(1 - \mu)^{N(1-y)}.$$

for $y = 1/N, 2/N, \ldots, 1$. That is, the response is a sample proportion out of $N$ trials, and the argument size in rzibinom is $N$ here. The parameter $\phi$ is the probability of a structural zero, and it satisfies $0 < \phi < 1$. The mean of $Y$ is $E(Y) = (1 - \phi)\mu$ and these are returned as
the fitted values by default. By default, the two linear/additive predictors for \texttt{zibinomial()} are \((\logit(\phi), \logit(\mu))^T\).

The \texttt{VGAM} family function \texttt{zibinomialff()} has a few changes compared to \texttt{zibinomial()}. These are: (i) the order of the linear/additive predictors is switched so the binomial probability comes first; (ii) argument \texttt{onempstrP} is now 1 minus the probability of a structural zero, i.e., the probability of the parent (binomial) component, i.e., \texttt{onempstrP} is 1-\texttt{pstrP}; (iii) argument \texttt{zero} has a new default so that the \texttt{onempstrP} is intercept-only by default. Now \texttt{zibinomialff()} is generally recommended over \texttt{zibinomial()}. Both functions implement Fisher scoring.

\textbf{Value}

An object of class "\texttt{vglmff}" (see \texttt{vglmff-class}). The object is used by modelling functions such as \texttt{vglm} and \texttt{vgam}.

\textbf{Warning}

Numerical problems can occur. Half-stepping is not uncommon. If failure to converge occurs, make use of the argument \texttt{ipstrP} or \texttt{ionempstrP}, or \texttt{imethod}.

\textbf{Note}

The response variable must have one of the formats described by \texttt{binomialff}, e.g., a factor or two column matrix or a vector of sample proportions with the \texttt{weights} argument specifying the values of \(N\).

To work well, one needs large values of \(N\) and \(\mu > 0\), i.e., the larger \(N\) and \(\mu\) are, the better. If \(N = 1\) then the model is unidentifiable since the number of parameters is excessive.

Setting \texttt{stepsize = 0.5}, say, may aid convergence.

Estimated probabilities of a structural zero and an observed zero are returned, as in \texttt{zipoisson}.

The zero-deflated binomial distribution might be fitted by setting \texttt{lpstrP = identitylink}, albeit, not entirely reliably. See \texttt{zipoisson} for information that can be applied here. Else try the zero-altered binomial distribution (see \texttt{zabinomial}).

\textbf{Author(s)}

T. W. Yee

\textbf{References}


\textbf{See Also}

\texttt{rzibinom, binomialff, posbinomial, rbinom}. 

Examples

```r
size <- 10  # Number of trials; N in the notation above
nn <- 200
zdata <- data.frame(pstr0 = logit(0, inverse = TRUE), # 0.50
                     mubin = logit(-1, inverse = TRUE), # Mean of usual binomial
                     sv = rep(size, length = nn))
zdata <- transform(zdata,
                   y = rbinom(nn, size = sv, prob = mubin, pstr0 = pstr0))
with(zdata, table(y))
fit <- vglm(cbind(y, sv - y) ~ 1, zinomialff, data = zdata, trace = TRUE)
fit <- vglm(cbind(y, sv - y) ~ 1, zinomialff, data = zdata, trace = TRUE, stepsize = 0.5)

coef(fit, matrix = TRUE)
Coef(fit) # Useful for intercept-only models
fitted(fit, type = "pobs0") # Estimate of P(Y = 0)
head(fitted(fit))
with(zdata, mean(y)) # Compare this with fitted(fit)
summary(fit)
```

---

### Zigeom

**Zero-Inflated Geometric Distribution**

**Description**

Density, and random generation for the zero-inflated geometric distribution with parameter `pstr0`.

**Usage**

- `dzigeom(x, prob, pstr0 = 0, log = FALSE)`
- `pzigeom(q, prob, pstr0 = 0)`
- `qzigeom(p, prob, pstr0 = 0)`
- `rzigeom(n, prob, pstr0 = 0)`

**Arguments**

- `x`, `q` vector of quantiles.
- `p` vector of probabilities.
- `prob` see `dgeom`.
- `n` Same as in `runif`.
- `pstr0` Probability of structural zero (ignoring the geometric distribution), called φ. The default value corresponds to the response having an ordinary geometric distribution.
- `log` Logical. Return the logarithm of the answer?
Details

The probability function of $Y$ is 0 with probability $\phi$, and geometric($\text{prob}$) with probability $1 - \phi$. Thus

$$P(Y = 0) = \phi + (1 - \phi)P(W = 0)$$

where $W$ is distributed geometric($\text{prob}$).

Value

dzigeom gives the density, pzigeom gives the distribution function, qzigeom gives the quantile function, and rzigeom generates random deviates.

Note

The argument $\text{pstr}\theta$ is recycled to the required length, and must have values which lie in the interval $[0, 1]$.

These functions actually allow for zero-deflation. That is, the resulting probability of a zero count is less than the nominal value of the parent distribution. See $\text{zipois}$ for more information.

Author(s)

T. W. Yee

See Also

$\text{zigeometric, dgeom}$.

Examples

```r
prob <- 0.5; pstr0 <- 0.2; x <- (-1):20
(ii <- dzigeom(x, prob, pstr0))
max(abs(cumsum(ii) - pzigeom(x, prob, pstr0))) # Should be 0
table(rzigeom(1000, prob, pstr0))
```

```r
## Not run: x <- 0:10
barplot(rbind(dzigeom(x, prob, pstr0), dgeom(x, prob)),
beside = TRUE, col = c("blue","orange"),
ylab = "P[Y = y]", xlab = "y", las = 1,
main = paste("zigeometric(" , prob, ", pstr0 = ", pstr0,
") (blue) vs",
" geometric(" , prob, ") (orange)", sep = ""),
names.arg = as.character(x))
```

## End(Not run)
Description

Fits a zero-inflated geometric distribution by maximum likelihood estimation.

Usage

\[
\begin{align*}
\text{zigeometric} & (lpstr0 = \text{"logit"}, lprob = \text{"logit"},) \\
& \quad \text{type.fitted} = \text{c("mean", "pobs0", "pstr0", "onempstr0")}, \\
& \quad \text{ipstr0} = \text{NULL}, \text{iprob} = \text{NULL}, \\
& \quad \text{imethod} = 1, \text{bias.red} = 0.5, \text{zero} = \text{NULL}) \\
\text{zigeometricff} & (lpstr0 = \text{"logit"}, \text{iempstr0} = \text{"logit"},) \\
& \quad \text{type.fitted} = \text{c("mean", "pobs0", "pstr0", "onempstr0")}, \\
& \quad \text{iprob} = \text{NULL}, \text{iempstr0} = \text{NULL}, \\
& \quad \text{imethod} = 1, \text{bias.red} = 0.5, \text{zero} = -2)
\end{align*}
\]

Arguments

- \text{lpstr0}, \text{lprob}: Link functions for the parameters \(\phi\) and \(p\) (prob). The usual geometric probability parameter is the latter. The probability of a structural zero is the former. See Links for more choices. For the zero-deflated model see below.
- \text{iempstr0}, \text{ionempstr0}: Corresponding arguments for the other parameterization. See details below.
- \text{bias.red}: A constant used in the initialization process of \(\text{pstr0}\). It should lie between 0 and 1, with 1 having no effect.
- \text{type.fitted}: See CommonVGAMffArguments and fittedvlm for information.
- \text{ipstr0}, \text{iprob}: See CommonVGAMffArguments for information.
- \text{zero}, \text{imethod}: See CommonVGAMffArguments for information.

Details

Function \text{zigeometric}() is based on

\[
P(Y = 0) = \phi + (1 - \phi)p,
\]

for \(y = 0\), and

\[
P(Y = y) = (1 - \phi)p(1 - p)^y.
\]

for \(y = 1, 2, \ldots\). The parameter \(\phi\) satisfies \(0 < \phi < 1\). The mean of \(Y\) is \(E(Y) = (1 - \phi)p/(1 - p)\) and these are returned as the fitted values by default. By default, the two linear/additive predictors are \((\text{logit}(\phi), \text{logit}(p))^T\). Multiple responses are handled.

Estimated probabilities of a structural zero and an observed zero can be returned, as in \text{zipoisson}; see fittedvlm for information.
The `VGAM` family function `zigeometricff()` has a few changes compared to `zigeometric()`. These are: (i) the order of the linear/additive predictors is switched so the geometric probability comes first; (ii) argument `onempstrP` is now 1 minus the probability of a structural zero, i.e., the probability of the parent (geometric) component, i.e., `onempstrP` is 1-pstr0; (iii) argument zero has a new default so that the `onempstrP` is intercept-only by default. Now `zigeometricff()` is generally recommended over `zigeometric()`. Both functions implement Fisher scoring and can handle multiple responses.

**Value**

An object of class "`vglmff`" (see `vglmff-class`). The object is used by modelling functions such as `vglm` and `vgam`.

**Note**

The zero-deflated geometric distribution might be fitted by setting `lpstr0 = identitylink`, albeit, not entirely reliably. See `zipoisson` for information that can be applied here. Else try the zero-altered geometric distribution (see `zageometric`).

**Author(s)**

T. W. Yee

**See Also**

`rzigeom`, `geometric`, `zageometric`, `rgeom`, `simulate.vlm`.

**Examples**

```r
# gdata <- data.frame(x2 = runif(nn <- 1000) - 0.5)
gdata <- transform(gdata, x3 = runif(nn) - 0.5,
                    x4 = runif(nn) - 0.5)
gdata <- transform(gdata, eta1 = 1.0 - 1.0 * x2 + 2.0 * x3,
                    eta2 = -1.0,
                    eta3 = 0.5)
gdata <- transform(gdata, prob1 = logit(eta1, inverse = TRUE),
                    prob2 = logit(eta2, inverse = TRUE),
                    prob3 = logit(eta3, inverse = TRUE))
gdata <- transform(gdata, y1 = rzigeom(nn, prob1, pstr0 = prob3),
                    y2 = rzigeom(nn, prob2, pstr0 = prob3),
                    y3 = rzigeom(nn, prob2, pstr0 = prob3))

with(gdata, table(y1))
with(gdata, table(y2))
with(gdata, table(y3))
head(gdata)

fit1 <- vglm(y1 ~ x2 + x3 + x4, zigeometric(zero = 1), data = gdata, trace = TRUE)
coef(fit1, matrix = TRUE)
head(fitted(fit1, type = "pstr0"))

fit2 <- vglm(cbind(y2, y3) ~ 1, zigeometric(zero = 1), data = gdata, trace = TRUE)
```
Zinegbin

Zero-Inflated Negative Binomial Distribution

Description
Density, distribution function, quantile function and random generation for the zero-inflated negative binomial distribution with parameter pstr philosophers.

Usage

dzinegbin(x, size, prob = NULL, munb = NULL, pstr0 = 0, log = FALSE)
pzinegbin(q, size, prob = NULL, munb = NULL, pstr0 = 0)
qzinegbin(p, size, prob = NULL, munb = NULL, pstr0 = 0)
rzinegbin(n, size, prob = NULL, munb = NULL, pstr0 = 0)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n Same as in runif.
size, prob, munb, log Arguments matching dnbinom. The argument munb corresponds to mu in dnbinom and has been renamed to emphasize the fact that it is the mean of the negative binomial component.
pstr0 Probability of structural zero (i.e., ignoring the negative binomial distribution), called φ.

Details
The probability function of Y is 0 with probability φ, and a negative binomial distribution with probability 1 − φ. Thus

\[ P(Y = 0) = \phi + (1 - \phi)P(W = 0) \]

where W is distributed as a negative binomial distribution (see rnbinom.) See negbinomial, a VGAM family function, for the formula of the probability density function and other details of the negative binomial distribution.

Value
dzinegbin gives the density, pzinegbin gives the distribution function, qzinegbin gives the quantile function, and rzinegbin generates random deviates.
Note

The argument \( pstr0 \) is recycled to the required length, and must have values which lie in the interval \([0,1] \).

These functions actually allow for zero-deflation. That is, the resulting probability of a zero count is less than the nominal value of the parent distribution. See \texttt{zipois} for more information.

Author(s)

T. W. Yee

See Also

\texttt{zinegbinomial}, \texttt{rnbinom}, \texttt{rzipois}.

Examples

```r
mumb <- 3; pstr0 <- 0.2; size <- k <- 10; x <- 0:10
(ii <- dzinegbin(x, pstr0 = pstr0, mu = mumb, size = k))
max(abs(cumsum(ii) - pzinegbin(x, pstr0 = pstr0, mu = mumb, size = k))) # 0
table(rzinegbin(100, pstr0 = pstr0, mu = mumb, size = k))

table(qzinegbin(runif(1000), pstr0 = pstr0, mu = mumb, size = k))
round(dzinegbin(x, pstr0 = pstr0, mu = mumb, size = k) * 1000) # Should be similar

## Not run: barplot(rbind(dzinegbin(x, pstr0 = pstr0, mu = mumb, size = k),
dbinom(x, mu = mumb, size = k)), las = 1,
beside = TRUE, col = c("blue", "green"), ylab = "Probability",
main = paste("ZINB(mu = ", mumb, ", k = ", k, ", pstr0 = ", pstr0,
" (blue) vs NB(mu = ", mumb, ", size = ", k, ") (green)", sep = ")",
names.arg = as.character(x))
## End(Not run)
```

zinegbinomial

Zero-Inflated Negative Binomial Distribution Family Function

Description

Fits a zero-inflated negative binomial distribution by full maximum likelihood estimation.

Usage

```r
zinegbinomial(lpstr0 = "logit", lnumb = "loge", lsize = "loge",
type.fitted = c("mean", "pobs0", "pstr0", "onempstr0"),
ipstr0 = NULL, isize = NULL, zero = -3,
imethod = 1, ishrinkage = 0.95, nsimEIM = 250)
zinegbinomialff(lnumb = "loge", lsize = "loge", lonempstr0 = "logit",
type.fitted = c("mean", "pobs0", "pstr0", "onempstr0"),
```
isize = NULL, ionempstr0 = NULL, zero = c(-2, -3),
imethod = 1, ishrinkage = 0.95, nsimEIM = 250)

Arguments

lpstr0, lmub, lsize
Link functions for the parameters \( \phi \), the mean and \( k \); see negbinomial for details, and Links for more choices. For the zero-deflated model see below.

type.fitted
See CommonVGAMffArguments and fittedvlm for more information.
ipstr0, isize
Optional initial values for \( \phi \) and \( k \). The default is to compute an initial value internally for both. If a vector then recycling is used.

ionempstr0, ionempstr
Corresponding arguments for the other parameterization. See details below.

imethod
An integer with value 1 or 2 or 3 which specifies the initialization method for the mean parameter. If failure to converge occurs try another value and/or else specify a value for ishrinkage.

zero
Integers specifying which linear/additive predictor is modelled as intercepts only. If given, their absolute values must be either 1 or 2 or 3. The default is the \( \phi \) and \( k \) parameters (both for each response). See CommonVGAMffArguments for more information.

ishrinkage, nsimEIM
See CommonVGAMffArguments for information.

Details

These functions are based on

\[
P(Y = 0) = \phi + (1 - \phi)(k/(k + \mu))^k,
\]

and for \( y = 1, 2, \ldots \),

\[
P(Y = y) = (1 - \phi)\ dabinom(y, \mu, k).
\]

The parameter \( \phi \) satisfies \( 0 < \phi < 1 \). The mean of \( Y \) is \( (1 - \phi)\mu \) (returned as the fitted values). By default, the three linear/additive predictors for zinegbinomial() are \( (\logit(\phi), \log(\mu), \log(k))^T \).

See negbinomial, another VGAM family function, for the formula of the probability density function and other details of the negative binomial distribution.

Independent multivariate responses are handled. If so then arguments ipstr0 and isize may be vectors with length equal to the number of responses.

The VGAM family function zinegbinomialff() has a few changes compared to zinegbinomial(). These are: (i) the order of the linear/additive predictors is switched so the NB mean comes first; (ii) onempstr0 is now 1 minus the probability of a structural 0, i.e., the probability of the parent (NB) component, i.e., onempstr0 is 1-pstr0; (iii) argument zero has a new default so that the onempstr0 is intercept-only by default. Now zinegbinomialff() is generally recommended over zinegbinomial(). Both functions implement Fisher scoring and can handle multiple responses.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, and vgam.
Warning

Numerical problems can occur, e.g., when the probability of zero is actually less than, not more than, the nominal probability of zero. Half-stepping is not uncommon. If failure to converge occurs, try using combinations of arguments stepsize (in \texttt{vglm.control}), imethod, ishrinkage, ipstr0, isize, and/or zero if there are explanatory variables.

An infinite loop might occur if some of the fitted values (the means) are too close to 0.

This \texttt{VGAM} family function is computationally expensive and usually runs slowly; setting \texttt{trace = TRUE} is useful for monitoring convergence.

Note

Estimated probabilities of a structural zero and an observed zero can be returned, as in \texttt{zipoisson}; see \texttt{fittedvlm} for more information.

If $k$ is large then the use of \texttt{VGAM} family function \texttt{zipoisson} is probably preferable. This follows because the Poisson is the limiting distribution of a negative binomial as $k$ tends to infinity.

The zero-deflated negative binomial distribution might be fitted by setting \texttt{lpstr0 = identitylink}, albeit, not entirely reliably. See \texttt{zipoisson} for information that can be applied here. Else try the zero-altered negative binomial distribution (see \texttt{zanegbinomial}).

Author(s)

T. W. Yee

See Also

\texttt{Zinegbin}, \texttt{negbinomial}, \texttt{rpois}, \texttt{CommonVGAMffArguments}.

Examples

```r
# Not run:  # Example 1
ndata <- data.frame(x2 = runif(nn <- 1000))
ndata <- transform(ndata, pstr0 = logit(-0.5 + 1 * x2, inverse = TRUE),
                  munb = exp( 3 + 1 * x2),
                  size = exp( 0 + 2 * x2))
ndata <- transform(ndata,
                   y1 = rzinegbin(nn, mu = munb, size = size, pstr0 = pstr0),
                   y2 = rzinegbin(nn, mu = munb, size = size, pstr0 = pstr0))
with(ndata, table(y1)['0'] / sum(table(y1)))
fit <- vglm(cbind(y1, y2) ~ x2, zinegbinomial(zero = NULL), data = ndata)
coef(fit, matrix = TRUE)
summary(fit)
head(cbind(fitted(fit), with(ndata, (1 - pstr0) * munb)))
round(vcov(fit), 3)

# Example 2: RR-ZINB could also be called a COZIVGLM-ZINB-2
ndata <- data.frame(x2 = runif(nn <- 2000))
ndata <- transform(ndata, x3 = runif(nn))
ndata <- transform(ndata, etal = 3 + 1 * x2 + 2 * x3)
```
ndata <- transform(ndata, pstr0 = logit(-1.5 + 0.5 * eta1), inverse = TRUE),
munb = exp(eta1),
size = exp(4))
ndata <- transform(ndata,
  y1 = rzinegbin(nn, pstr0 = pstr0, mu = munb, size = size))
with(ndata, table(y1)["0"] / sum(table(y1)))
rrzinb <- rrvglm(y1 ~ x2 + x3, zinegbinomial(zero = NULL), data = ndata,
                Index.corner = 2, str0 = 3, trace = TRUE)
coef(rrzinb, matrix = TRUE)
Coef(rrzinb)
## End(Not run)

---

### Description

Fits an exchangeable bivariate odds-ratio model to two binary responses with a complementary log-log link. The data are assumed to come from a zero-inflated Poisson distribution that has been converted to presence/absence.

### Usage

```r
zipebcom(lmu12 = "cloglog", lphi12 = "logit", loratio = "loge",
        imu12 = NULL, iphi12 = NULL, ioratio = NULL,
        zero = 2:3, tol = 0.001, addRidge = 0.001)
```

### Arguments

- **lmu12, imu12**: Link function, extra argument and optional initial values for the first (and second) marginal probabilities. Argument `lmu12` should be left alone. Argument `imu12` may be of length 2 (one element for each response).
- **lphi12**: Link function applied to the \( \phi \) parameter of the zero-inflated Poisson distribution (see `zipoisson`). See [Links](#) for more choices.
- **loratio**: Link function applied to the odds ratio. See [Links](#) for more choices.
- **iphi12, ioratio**: Optional initial values for \( \phi \) and the odds ratio. See [CommonVGAMffArguments](#) for more details. In general, good initial values (especially for `iphi12`) are often required, therefore use these arguments if convergence failure occurs. If inputted, the value of `iphi12` cannot be more than the sample proportions of zeros in either response.
- **zero**: Which linear/additive predictor is modelled as an intercept only? A NULL means none. The default has both \( \phi \) and the odds ratio as not being modelled as a function of the explanatory variables (apart from an intercept).
tol

Tolerance for testing independence. Should be some small positive numerical value.

addRidge

Some small positive numerical value. The first two diagonal elements of the working weight matrices are multiplied by $1 + \text{addRidge}$ to make it diagonally dominant, therefore positive-definite.

Details

This VGAM family function fits an exchangeable bivariate odds ratio model (binom2.or) with a cloglog link. The data are assumed to come from a zero-inflated Poisson (ZIP) distribution that has been converted to presence/absence. Explicitly, the default model is

$$\text{cloglog}[P(Y_j = 1)/(1 - \phi)] = \eta_1, \quad j = 1, 2$$

for the (exchangeable) marginals, and

$$\text{logit}[\phi] = \eta_2,$$

for the mixing parameter, and

$$\log[P(Y_{00} = 1)P(Y_{11} = 1)/(P(Y_{01} = 1)P(Y_{10} = 1))] = \eta_3,$$

specifies the dependency between the two responses. Here, the responses equal 1 for a success and a 0 for a failure, and the odds ratio is often written $\psi = p_{00}p_{11}/(p_{10}p_{01})$. We have $p_{10} = p_{01}$ because of the exchangeability.

The second linear/additive predictor models the $\phi$ parameter (see zipoisson). The third linear/additive predictor is the same as binomRNor, viz., the log odds ratio.

Suppose a dataset1 comes from a Poisson distribution that has been converted to presence/absence, and that both marginal probabilities are the same (exchangeable). Then binom2.or("cloglog", exch=TRUE) is appropriate. Now suppose a dataset2 comes from a zero-inflated Poisson distribution. The first linear/additive predictor of zipebcom() applied to dataset2 is the same as that of binom2.or("cloglog", exch=TRUE) applied to dataset1. That is, the $\phi$ has been taken care of by zipebcom() so that it is just like the simpler binom2.or.

Note that, for $\eta_1$, mu12 = prob12 / (1-phi12) where prob12 is the probability of a 1 under the ZIP model. Here, mu12 correspond to mu1 and mu2 in the binom2.or-Poisson model.

If $\phi = 0$ then zipebcom() should be equivalent to binom2.or("cloglog", exch=TRUE). Full details are given in Yee and Dirnbock (2009).

The leading $2 \times 2$ submatrix of the expected information matrix (EIM) is of rank-1, not 2! This is due to the fact that the parameters corresponding to the first two linear/additive predictors are unidentifiable. The quick fix around this problem is to use the addRidge adjustment. The model is fitted by maximum likelihood estimation since the full likelihood is specified. Fisher scoring is implemented.

The default models $\eta_2$ and $\eta_3$ as single parameters only, but this can be circumvented by setting zero=NULL in order to model the $\phi$ and odds ratio as a function of all the explanatory variables.
Value
An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

When fitted, the fitted.values slot of the object contains the four joint probabilities, labelled as $(Y_1, Y_2) = (0,0), (0,1), (1,0), (1,1)$, respectively. These estimated probabilities should be extracted with the fitted generic function.

Warning
The fact that the EIM is not of full rank may mean the model is naturally ill-conditioned. Not sure whether there are any negative consequences wrt theory. For now it is certainly safer to fit binom2.or to bivariate binary responses.

Note
The "12" in the argument names reinforce the user about the exchangeability assumption. The name of this VGAM family function stands for zero-inflated Poisson exchangeable bivariate complementary log-log odds-ratio model or ZIP-EBCOM.

See binom2.or for details that are pertinent to this VGAM family function too. Even better initial values are usually needed here.

The xij (see vglm.control) argument enables environmental variables with different values at the two time points to be entered into an exchangeable binom2.or model. See the author’s webpage for sample code.

References

See Also
binom2.or, zipoisson, cloglog, CommonVGAMffArguments.

Examples
zdata <- data.frame(x2 = seq(0, 1, len = (nsites <- 2000)))
zdata <- transform(zdata, etal = -3 + 5 * x2,
    phi1 = logit(-1, inverse = TRUE),
    oratio = exp(2))
zdata <- transform(zdata, mul2 = cloglog(eta1, inverse = TRUE) * (1-phi1))
tmat <- with(zdata, rbinom2.or(nsites, mul = mul2, oratio = oratio, exch = TRUE))
zdata <- transform(zdata, ybin1 = tmat[, 1], ybin2 = tmat[, 2])

with(zdata, table(ybin1, ybin2)) / nsites # For interest only
## Not run:
# Various plots of the data, for interest only
par(mfrow = c(2, 2))
plot(jitter(ybin1) ~ x2, data = zdata, col = "blue")
plot(jitter(ybin2) - jitter(ybin1), data = zdata, col = "blue")

plot(mu2 ~ x2, data = zdata, col = "blue", type = "l", ylim = 0:1, 
ylab = "Probability", main = "Marginal probability and phi")
with(zdata, abline(h = phi1[1], col = "red", lty = "dashed"))

tmat2 <- with(zdata, dbinom2.or(mu1 = mu2, oratio = oratio, exch = TRUE))
with(zdata, matplot(x2, tmat2, col = 1:4, type = "l", ylim = 0:1, 
ylab = "Probability", main = "Joint probabilities"))
## End(Not run)

# Now fit the model to the data.
fit <- vglm(cbind(ybin1, ybin2) ~ x2, zipfcom, data = zdata, trace = TRUE)
coef(fit, matrix = TRUE)
summary(fit)
vcov(fit)

---

**Zipf**

*The Zipf Distribution*

**Description**

Density, and cumulative distribution function for the Zipf distribution.

**Usage**

```r
dzipf(x, N, s, log = FALSE)
pzipf(q, N, s, log.p = FALSE)
```

**Arguments**

- `x, q` vector of quantiles.
- `N, s` the number of elements, and the exponent characterizing the distribution. See `zipf` for more details.
- `log` Logical. If `log = TRUE` then the logarithm of the density is returned.
- `log.p` Same meaning as in `pnorm` or `qnorm`.

**Details**

This is a finite version of the zeta distribution. See `zipf` for more details.

**Value**

dzipf gives the density, and pzipf gives the cumulative distribution function.

**Author(s)**

T. W. Yee
zipf

See Also

zipf.

Examples

```r
N <- 10; s <- 0.5; y <- 1:N
proby <- dzipf(y, N = N, s = s)
## Not run: plot(proby - y, type = "h", col = "blue", ylab = "Probability",
     ylim = c(0, 0.2), main = paste("Zipf(N = ", N, ", s = ", s, ")", sep = ""),
     lwd = 2, las = 1)
## End(Not run)
sum(proby) # Should be 1
max(abs(cumsum(proby) - pzipf(y, N = N, s = s))) # Should be 0
```

zipf

Zipf Distribution Family Function

Description

Estimates the parameter of the Zipf distribution.

Usage

```r
zipf(N = NULL, link = "log", init.s = NULL)
```

Arguments

- `N` Number of elements, an integer satisfying \( 1 < N < \text{Inf} \). The default is to use the maximum value of the response. If given, \( N \) must be no less that the largest response value. If \( N = \text{Inf} \) and \( s > 1 \) then this is the zeta distribution (use zetaff instead).
- `link` Parameter link function applied to the (positive) parameter \( s \). See Links for more choices.
- `init.s` Optional initial value for the parameter \( s \). The default is to choose an initial value internally. If converge failure occurs use this argument to input a value.

Details

The probability function for a response \( Y \) is

\[
P(Y = y) = y^{-s} / \sum_{i=1}^{N} i^{-s}, \quad s > 0, \quad y = 1, 2, \ldots, N,
\]

where \( s \) is the exponent characterizing the distribution. The mean of \( Y \), which are returned as the fitted values, is \( \mu = H_{N,s-1}/H_{N,s} \) where \( H_{n,m} = \sum_{i=1}^{m} i^{-m} \) is the \( n \)th generalized harmonic number.
Zipf’s law is an experimental law which is often applied to the study of the frequency of words in a corpus of natural language utterances. It states that the frequency of any word is inversely proportional to its rank in the frequency table. For example, “the” and “of” are first two most common words, and Zipf’s law states that “the” is twice as common as “of”. Many other natural phenomena conform to Zipf’s law.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm and vgam.

Note

Upon convergence, the N is stored as @misc$N$.

Author(s)

T. W. Yee

References


See Also

dzipf, zetaff, simulate.vlm.

Examples

zdata <- data.frame(y = 1:5, ofreq = c(63, 14, 5, 1, 2))
fit <- vglm(y ~ 1, zipf, data = zdata, trace = TRUE, weight = ofreq, crit = "coef")
fit <- vglm(y ~ 1, zipf(link = identitylink, init = 3.4), data = zdata,
trace = TRUE, weight = ofreq)
fit@misc$N

(zhat <- Coef(fit))
with(zdata, weighted.mean(y, ofreq))
fitted(fit, matrix = FALSE)

---

**Zero-Inflated Poisson Distribution**

**Description**

Density, distribution function, quantile function and random generation for the zero-inflated Poisson distribution with parameter `psirho`. 
Usage

\texttt{dzipois(x, lambda, pstr0 = 0, log = FALSE)}
\texttt{pzipois(q, lambda, pstr0 = 0)}
\texttt{qzipois(p, lambda, pstr0 = 0)}
\texttt{rzipois(n, lambda, pstr0 = 0)}

Arguments

\begin{itemize}
  \item \texttt{x, q} \hspace{1cm} \text{vector of quantiles.}
  \item \texttt{p} \hspace{1cm} \text{vector of probabilities.}
  \item \texttt{n} \hspace{1cm} \text{number of observations. Must be a single positive integer.}
  \item \texttt{lambda} \hspace{1cm} \text{Vector of positive means.}
  \item \texttt{pstr0} \hspace{1cm} \text{Probability of a structural zero (i.e., ignoring the Poisson distribution), called } \phi. \text{ The default value of } \phi = 0 \text{ corresponds to the response having an ordinary Poisson distribution.}
  \item \texttt{log} \hspace{1cm} \text{Logical. Return the logarithm of the answer?}
\end{itemize}

Details

The probability function of \( Y \) is 0 with probability \( \phi \), and \( \text{Poisson}(\lambda) \) with probability \( 1 - \phi \). Thus

\[ P(Y = 0) = \phi + (1 - \phi)P(W = 0) \]

where \( W \) is distributed \( \text{Poisson}(\lambda) \).

Value

dzipois gives the density, pzipois gives the distribution function, qzipois gives the quantile function, and rzipois generates random deviates.

Note

The argument \texttt{pstr0} is recycled to the required length, and must have values which lie in the interval \([0, 1]\).

These functions actually allow for the zero-deflated \textit{Poisson} distribution. Here, \texttt{pstr0} is also permitted to lie in the interval \([-1/\expm1(\lambda), 0]\). The resulting probability of a zero count is \textit{less than} the nominal Poisson value, and the use of \texttt{pstr0} to stand for the probability of a structural zero loses its meaning. When \texttt{pstr0} equals \(-1/\expm1(\lambda)\) this corresponds to the positive-Poisson distribution (e.g., see \texttt{dpospois}).

Author(s)

T. W. Yee

See Also

\texttt{zipoisson, dpois, rzinegbin}.
Examples

```r
lambda <- 3; pstr0 <- 0.2; x <- (-1):7
(ii <- dzipois(x, lambda, pstr0 = pstr0))
max(abs(cumsum(ii) - pzipois(x, lambda, pstr0 = pstr0))) # Should be 0

table(rzipois(100, lambda, pstr0 = pstr0))

table(qzipois(runif(100), lambda, pstr0))
round(dzipois(0:10, lambda, pstr0 = pstr0) * 100) # Should be similar

## Not run: x <- 0:10
par(mfrow = c(2, 1)) # Zero-inflated Poisson
barplot(rbind(dzipois(x, lambda, pstr0 = pstr0), dpois(x, lambda)),
beside = TRUE, col = c("blue", "orange"),
main = paste("ZIP(", lambda, ",", pstr0 = ",", pstr = ") (blue) vs",
" Poisson(" lambda, ") (orange)" sep = ""),
names.arg = as.character(x))

deflat.limit <- -1 / expm1(lambda) # Zero-deflated Poisson
newpstr0 <- round(deflat.limit / 1.5, 3)
barplot(rbind(dzipois(x, lambda, pstr0 = newpstr0),
dpois(x, lambda)),
beside = TRUE, col = c("blue", "orange"),
main = paste("ZDP(", lambda, ",", pstr0 = ",", newpstr0, ") (blue) vs",
" Poisson(" lambda, ") (orange)" sep = ""),
names.arg = as.character(x))

## End(Not run)
```

---

**zipoisson**

Zero-Inflated Poisson Distribution Family Function

### Description

Fits a zero-inflated Poisson distribution by full maximum likelihood estimation.

### Usage

```r
zipoisson(lpstr0 = "logit", llambda = "log",
type.fitted = c("mean", "pobs0", "pstr0", "onempstr0"),
lpstr = NULL, llambda = NULL,
imethod = 1, ishrinkage = 0.8, zero = NULL)
zipoissonff(llambda = "log", lonomempstr0 = "logit",
type.fitted = c("mean", "pobs0", "pstr0", "onempstr0"),
ilambda = NULL, ionempstr0 = NULL,
imethod = 1, ishrinkage = 0.8, zero = -2)
```
Arguments

lpstrφ, llambda
Link function for the parameter φ and the usual λ parameter. See Links for more choices; see CommonVGAMffArguments for more information. For the zero-deflated model see below.

ipstrφ, ilambda
Optional initial values for φ, whose values must lie between 0 and 1. Optional initial values for λ, whose values must be positive. The defaults are to compute an initial value internally for each. If a vector then recycling is used.

lonempstrφ, ionempstrλ
Corresponding arguments for the other parameterization. See details below.

type.fitted
Character. The type of fitted value to be returned. The first choice (the expected value) is the default. The estimated probability of an observed 0 is an alternative, else the estimated probability of a structural 0, or one minus the estimated probability of a structural 0. See CommonVGAMffArguments and fittedvlm for more information.

imethod
An integer with value 1 or 2 which specifies the initialization method for λ. If failure to converge occurs try another value and/or else specify a value for ishrinkage and/or else specify a value for ipstrφ. See CommonVGAMffArguments for more information.

ishrinkage
How much shrinkage is used when initializing λ. The value must be between 0 and 1 inclusive, and a value of 0 means the individual response values are used, and a value of 1 means the median or mean is used. This argument is used in conjunction with imethod. See CommonVGAMffArguments for more information.

zero
An integer specifying which linear/additive predictor is modelled as intercepts only. If given, the value must be either 1 or 2, and the default is none of them. Setting zero = 1 makes φ a single parameter. See CommonVGAMffArguments for more information.

Details

These models are a mixture of a Poisson distribution and the value 0; it has value 0 with probability φ else is Poisson(λ) distributed. Thus there are two sources for zero values, and φ is the probability of a structural zero. The model for zipoisson() can be written

$$P(Y = 0) = \phi + (1 - \phi) \exp(-\lambda),$$

and for \(y = 1, 2, \ldots\),

$$P(Y = y) = (1 - \phi) \exp(-\lambda)\lambda^y/y!. $$

Here, the parameter φ satisfies \(0 < \phi < 1\). The mean of \(Y\) is \((1 - \phi)\lambda\) and these are returned as the fitted values, by default. The variance of \(Y\) is \((1 - \phi)\lambda(1 + \phi\lambda)\). By default, the two linear/additive predictors of zipoisson() are \((\text{logit}(\phi), \log(\lambda))^T\).

The VGAM family function zipoissonff() has a few changes compared to zipoisson(). These are: (i) the order of the linear/additive predictors is switched so the Poisson mean comes first; (ii) onempstrθ is now l minus the probability of a structural 0, i.e., the probability of the parent
(Poisson) component, i.e., onempstrθ is 1−pstrθ; (iii) argument zero has a new default so that the onempstrθ is intercept-only by default. Now zipoissonff() is generally recommended over zipoisson() (and definitely recommended over yip88). Both functions implement Fisher scoring and can handle multiple responses.

Value

An object of class "vglmff" (see vglmff-class). The object is used by modelling functions such as vglm, rrvglm and vgam.

Warning

Numerical problems can occur, e.g., when the probability of zero is actually less than, not more than, the nominal probability of zero. For example, in the Angers and Biswas (2003) data below, replacing 182 by 1 results in nonconvergence. Half-stepping is not uncommon. If failure to converge occurs, try using combinations of imethod, ishrinkage, ipstrθ, and/or zipoisson(zero = 1) if there are explanatory variables. The default for zipoissonff() is to model the structural zero probability as an intercept-only.

Note

Although the functions in Zipois can handle the zero-deflated Poisson distribution, this family function cannot estimate this very well in general. One sets lpstrθ = identitylink, however, the iterations might fall outside the parameter space. Practically, it is restricted to intercept-models only (see example below). Also, one might need inputting good initial values or using a simpler model to obtain initial values.

A (somewhat) similar and more reliable method for zero-deflation is to try the zero-altered Poisson model (see zapoisson).

The use of this VGAM family function with rrvglm can result in a so-called COZIGAM or COZIGLM. That is, a reduced-rank zero-inflated Poisson model (RR-ZIP) is a constrained zero-inflated generalized linear model. See COZIGAM. A RR-ZINB model can also be fitted easily; see zinegbinomial. Jargon-wise, a COZIGLM might be better described as a COZIVGLM-ZIP.

Author(s)

T. W. Yee

References


See Also

| zipoisson, Zipois, yip88, rrvglm, zipebcom, rpois, simulate.vlm

Examples

```r
# Example 1: simulated ZIP data
zdata <- data.frame(x2 = runif(nn <- 1000))
zdata <- transform(zdata, pstr01 = logit(-0.5 + 1*x2, inverse = TRUE),
pstr02 = logit( 0.5 - 1*x2, inverse = TRUE),
Ps01 = logit(-0.5 , inverse = TRUE),
Ps02 = logit( 0.5 , inverse = TRUE),
lambda1 = loge(-0.5 + 2*x2, inverse = TRUE),
lambda2 = loge( 0.5 + 2*x2, inverse = TRUE))
zdata <- transform(zdata, y1 = rzipois(nn, lambda = lambda1, pstr0 = Ps01),
y2 = rzipois(nn, lambda = lambda2, pstr0 = Ps02))

with(zdata, table(y1))  # Eyeball the data
with(zdata, table(y2))
fit1 <- vglm(y1 ~ x2, zipoisson(zerol = 1), data = zdata, crit = "coef")
fit2 <- vglm(y2 ~ x2, zipoisson(zerol = 1), data = zdata, crit = "coef")
coef(fit1, matrix = TRUE)  # These should agree with the above values
coef(fit2, matrix = TRUE)  # These should agree with the above values

# Fit all two simultaneously, using a different parameterization:
fit12 <- vglm(cbind(y1, y2) ~ x2, zipoissonff, data = zdata, crit = "coef")
coef(fit12, matrix = TRUE)  # These should agree with the above values

# For the first observation compute the probability that y1 is
# due to a structural zero.
(fitted(fit1, type = "pstr0") / fitted(fit1, type = "pobs0"))[1]
```

# Example 2: McKendrick (1926). Data from 223 Indian village households
cholera <- data.frame(ncases = 0:4,  # Number of cholera cases,
wfreq = c(168, 32, 16, 6, 1))  # Frequencies
fit <- vglm(ncases ~ 1, zipoisson, wei = wfreq, cholera, trace = TRUE)
coef(fit, matrix = TRUE)
with(cholera, cbind(actual = wfreq,
   fitted = round(dzipois(ncases, lambda = Coef(fit)[2],
   pstr0 = Coef(fit)[1]) * sum(wfreq), digits = 2)))

# Example 3: data from Angers and Biswas (2003)
abdata <- data.frame(y = 0:7, w = c(182, 41, 12, 2, 2, 0, 0, 1))
abdata <- subset(abdata, w > 0)
fit <- vglm(y ~ 1, zipoisson(lpstr0 = probit, ipstr0 = 0.8),
data = abdata, weight = w, trace = TRUE)
fitted(fit, type = "pobs0")  # Estimate of P(Y = 0)
coef(fit, matrix = TRUE)
Coef(fit)  # Estimate of pstr0 and lambda
fitted(fit)
with(abdata, weighted.mean(y, w))  # Compare this with fitted(fit)
summary(fit)

# Example 4: zero-deflated model for an intercept-only data
zdata <- transform(zdata, lambda3 = loge(0.0, inverse = TRUE))
zdata <- transform(zdata, deflat.limit = -1 / expm1(lambda3))  # Boundary
# The 'pstr0' parameter is negative and in parameter space:
zdata <- transform(zdata, usepstr0 = deflat.limit / 1.5)
zdata <- transform(zdata, y3 = rzipois(mm, lambda3, pstr0 = usepstr0))
head(zdata)
with(zdata, table(y3))  # A lot of deflation
fit3 <- vglm(y3 ~ 1, zipoisson(zero = -1, lpstr0 = identitylink),
            data = zdata, trace = TRUE, crit = "coef")
coef(fit3, matrix = TRUE)
# Check how accurate it was:
zdata[1, "usepstr0"]  # Answer
c coef(fit3)[1]  # Estimate
Coef(fit3)

# Example 5: This RR-ZIP is known as a COZIGAM or COZIVGLM-ZIP
set.seed(123)
rrzip <- rrvglm(Alopacce ~ sm.bs(WaterCon, df = 3), zipoisson(zero = NULL),
                data = hspider, trace = TRUE, Index.corner = 2)
coef(rrzip, matrix = TRUE)
Coef(rrzip)
summary(rrzip)
## Not run: plotvgam(rrzip, lcol = "blue")
Index

*Topic **classes**
- biplot-methods, 109
- calibrate-methods, 126
- Coef.qrrvglm-class, 164
- Coef.rrvglm-class, 167
- concoef-methods, 178
- rrvglm-class, 630
- SurvS4-class, 675
- vgam-class, 707
- vglm-class, 717
- vglmff-class, 723

*Topic **datagen**
- posbernUC, 555
- rcqo, 608
- simulate.vlm, 649

*Topic **datasets**
- auuc, 40
- backPain, 43
- beggs, 44
- bmi.nz, 116
- cfibrosis, 152
- chest.nz, 154
- chinese.nz, 155
- coalminers, 160
- corbet, 180
- crashes, 188
- deermice, 200
- ducklings, 218
- enzyme, 219
- finney44, 254
- flourbeetle, 261
- gew, 304
- grain.us, 315
- hormone, 333
- hspider, 335
- Huggins89.t1, 338
- hunua, 340
- lake0, 369
- leukemia, 379
- lirat, 395
- machinists, 445
- marital.nz, 451
- melbmaxtemp, 457
- olympics, 499
- oxttemp, 504
- pneumo, 537
- prats, 573
- prinia, 580
- ruge, 637
- toxop, 687
- ucberk, 698
- V1, 700
- venice, 702
- waitakere, 730
- wine, 737

*Topic **distribution**
- alaplaceUC, 27
- Benford, 45
- Benini, 46
- Betabinom, 49
- Betageom, 59
- Betanorm, 63
- Biawhocop, 68
- Biclaytoncop, 71
- Bifgmcop, 75
- bilogis, 84
- Binom2.or, 88
- Binom2.rho, 93
- Binorm, 100
- Binormcop, 105
- Biplackett, 106
- Bisa, 110
- Bistudentt, 113
- Bort, 118
- Card, 138
- Dagum, 195
- dAR1, 198
- dgenpois, 204
dhuber, 206
Log, 213
Expectiles-Exponential, 223
Expectiles-Normal, 224
Expectiles-sc.t2, 226
Expectiles-Uniform, 228
expgeom, 234
explog, 239
expois, 244
Felix, 246
Fisk, 256
Foldnorm, 262
Frank, 267
Frechet, 268
GenbetaII, 284
gengammaUC, 290
genray, 293
gcvUc, 302
Gompertz, 307
gpdUc, 313
Gumbel-II, 324
gumbelUC, 327
Hzeta, 345
Inv.gaussian, 352
Inv.Iomax, 355
Inv.paralogistic, 358
Kumar, 366
laplaceUC, 373
IgammaUC, 383
Lindley, 385
Lino, 392
Log, 404
loglapUC, 419
Lomax, 429
Makeham, 446
Maxwell, 453
Nakagami, 480
Paralogistic, 505
Pareto, 508
ParetoIV, 511
Perks, 516
PoissonPoints, 542
Polono, 545
posbernUC, 555
Posbinom, 557
Posgeom, 561
Posnegbin, 562
Posnorm, 566
Pospois, 569
Rayleigh, 603
rcqo, 608
rdiric, 612
Rice, 619
Simplex, 646
Sinmad, 651
Skellam, 654
skewnorm, 657
Slash, 660
Tikuv, 676
Tobit, 679
Triangle, 688
Truncpareto, 694
Yules, 743
Zabinom, 745
Zageom, 748
Zanegbin, 752
Zapois, 756
Zeta, 761
Zibinom, 765
Zigeom, 769
Zinegbin, 773
Zipf, 780
Zipois, 782
*Topic dpplot
  plotdeplot.lmscereg, 524
  plotqrrvgelm, 526
  plotvgam.control, 534
*Topic graphs
  deplot.lmscereg, 201
  lvplot.qrrvgelm, 438
  lvplot.rrvglm, 442
  persqrrvgelm, 519
  plotqplotl.lmscereg, 527
  plotvgam, 531
  plotvglm, 535
  prplot, 584
  qtpplot.gumbel, 592
  qtpplot.lmscereg, 594
  rlpplot.egev, 623
  trplot.gev, 692
*Topic htest
  lrtest, 435
*Topic manip
  iam, 348
*Topic math
  cauchit, 141
INDEX

158
cloglog
120
erf
220
expint
237
explink
238
fisherz
255
foldsqrt
265
golf
305
identitylink
349
kendall.tau
365
lambertW
370
lerch
378
log1pexp
405
logc
406
loge
407
logit
413
loglog
426
logoff
428
multilogit
474
nbcanlink
483
nbolf
485
ordpoisson
502
pgamma.deriv
522
pgamma.deriv.unscaled
523
polf
543
powerlink
572
probit
581
reciprocal
616
rhobit
618
zeta
762

*Topic methods
  biplot-methods
  calibrate-methods
  coef-methods

*Topic models
  A1A2A3
  AA.Aa.aa
  AB.Ab.ab.ab
  ABO
  acat
  AICv1m
  alaplace
  ambinomial
  amlexponential
  amlnormal
  amlpoisson
  AR1
  aux.posbernoulli.t
  beninni1

betabinomial
betabinomialff
betaff
betageometric
betaII
betaplateaux
betaprime
betar
betahc
biclaytoncop
BICv1m
bifgmcop
bifgmexp
bifrankcop
bigamma.mckay
bigumbellexp
bilogistic
binom2.or
binom2.rho
binomialff
binormal
binormalcop
bipackettcop
biplot-methods
bisa
bistudentt
borel.tanner
Brat
brat
bratt
brat
brat
bratl
brat
bolog
calibrate
calibrate-methods
calibrate.qrrvglm
calibrate.qrrvglm.control
cao
cao.control
cardio
cauc
cauchit
cauchy
CDF.lmscreg
cens.gumbel
cens.normal
cens.poisson
cgo
chisq
clo
cloglog
coef
coef.qrrvglm
lognormal, 427
logoff, 428
lomax, 431
lqnorm, 433
lvplot, 436
lvplot.qrrvglm, 438
lvplot.rrvglm, 442
makeham, 447
margeff, 449
Max, 452
maxwell, 454
mccullagh89, 456
meplot, 458
micmen, 460
mix2exp, 462
mix2normal, 464
mix2poisson, 466
MNs, 468
model.frame, 470
model.matrix, 471
multilogit, 474
multinomial, 476
nakagami, 481
nbcanlink, 483
nbolf, 485
negbinomial, 487
negbinomial.size, 493
normal.vcm, 494
nparam.vlm, 498
Opt, 500
ordpoisson, 502
paralogistic, 506
paretoff, 509
paretoIV, 513
perks, 517
perspqqrvgm, 519
plotdeplot, 524
plotqrvgm, 526
plotqplot, 527
plotvglm, 531
plotvglm.control, 534
plotvglm, 535
poisson.points, 538
poissonf, 539
polf, 543
posbernoulli.b, 547
posbernoulli.t, 549
posbernoulli.tb, 552
posbinomial, 559
posnegbinomial, 564
posnormal, 568
pospoisson, 571
powerlink, 572
predictqrvgm, 575
predictvglm, 576
prentice74, 578
probit, 581
propodds, 583
prplot, 584
put.smart, 585
qrvgm.control, 586
qplot.gumbel, 592
qplot.1mscreg, 594
quasibinomialff, 595
quasipoissonff, 597
Qvar, 599
qvar, 602
rayleigh, 605
rec.expl, 614
rec.normal, 615
reciprocal, 616
rhobit, 618
riceff, 620
rigff, 622
rlplot.egev, 623
rrar, 625
rrvgm, 627
rrvgm.control, 633
rrvgm.optim.control, 636
s, 638
sc.student2, 639
Select, 641
seq2binomial, 643
setup.smart, 645
simplex, 647
simulate.vlm, 649
sinmad, 652
skellam, 655
skewnormal, 658
slash, 661
smart.expression, 663
smart.mode.is, 664
smartpred, 665
sratio, 667
studentt, 669
SURff, 671
<table>
<thead>
<tr>
<th>Package</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>tikuv, 678</td>
<td></td>
</tr>
<tr>
<td>tobit, 681</td>
<td></td>
</tr>
<tr>
<td>Tol, 685</td>
<td></td>
</tr>
<tr>
<td>triangle, 689</td>
<td></td>
</tr>
<tr>
<td>trplot, 691</td>
<td></td>
</tr>
<tr>
<td>trplot.qrrvglm, 692</td>
<td></td>
</tr>
<tr>
<td>truncweibull, 696</td>
<td></td>
</tr>
<tr>
<td>uninormal, 699</td>
<td></td>
</tr>
<tr>
<td>vgam, 703</td>
<td></td>
</tr>
<tr>
<td>vgam-class, 707</td>
<td></td>
</tr>
<tr>
<td>VGAM-package, 11</td>
<td></td>
</tr>
<tr>
<td>vgam.control, 710</td>
<td></td>
</tr>
<tr>
<td>vglm, 712</td>
<td></td>
</tr>
<tr>
<td>vglm.control, 720</td>
<td></td>
</tr>
<tr>
<td>vonmises, 726</td>
<td></td>
</tr>
<tr>
<td>waldff, 732</td>
<td></td>
</tr>
<tr>
<td>weibullR, 733</td>
<td></td>
</tr>
<tr>
<td>weightsvglm, 736</td>
<td></td>
</tr>
<tr>
<td>wrapup.smart, 738</td>
<td></td>
</tr>
<tr>
<td>yeo.johnson, 739</td>
<td></td>
</tr>
<tr>
<td>yip88, 740</td>
<td></td>
</tr>
<tr>
<td>yulesimon, 744</td>
<td></td>
</tr>
<tr>
<td>zabinomial, 746</td>
<td></td>
</tr>
<tr>
<td>zageometric, 750</td>
<td></td>
</tr>
<tr>
<td>zanegbinomial, 753</td>
<td></td>
</tr>
<tr>
<td>zapoisson, 757</td>
<td></td>
</tr>
<tr>
<td>zero, 759</td>
<td></td>
</tr>
<tr>
<td>zetaff, 764</td>
<td></td>
</tr>
<tr>
<td>zebinomial, 767</td>
<td></td>
</tr>
<tr>
<td>zigeometric, 771</td>
<td></td>
</tr>
<tr>
<td>zinegbinomial, 774</td>
<td></td>
</tr>
<tr>
<td>zipebcom, 777</td>
<td></td>
</tr>
<tr>
<td>zipf, 781</td>
<td></td>
</tr>
<tr>
<td>zipoisson, 784</td>
<td></td>
</tr>
</tbody>
</table>

**Topic package**
- VGAM-package, 11

**Topic programming**
- get.smart, 297
- get.smart.prediction, 298
- iam, 348
- is.smart, 363
- put.smart, 585
- setup.smart, 645
- smart.expression, 663
- smart.mode.is, 664
- smartpred, 665
- wrapup.smart, 738
- zero, 759

**Topic regression**
- A1A2A3, 14
- AA.Aa.aa, 15
- AB.Ab.aB.ab, 17
- ABO, 18
- acat, 19
- AICvglm, 21
- alaplace, 23
- alaplacebf, 29
- alaplaceexponential, 31
- alaplanormal, 33
- almglm, 36
- ARI, 38
- aux.posbernoulli.t, 41
- benini1, 48
- betabinomial, 51
- betabinomialbf, 54
- betaff, 57
- betageometric, 60
- betaiI, 62
- betaPrime, 65
- betaR, 66
- biamhcop, 69
- biclaytoncop, 72
- BICvglm, 74
- bifgmncop, 77
- bifgmexp, 78
- bifrankcop, 80
- bigamma.mckay, 81
- bigumbel, 83
- bilogistic, 86
- binom2.or, 89
- binom2.rho, 94
- binomialbf, 97
- binormal, 102
- binormalcop, 103
- biplackettcop, 108
- biplot.methods, 109
- bisa, 111
- bistudentt, 114
- borel.tanner, 117
- Brat, 119
- brat, 121
- bratt, 123
- calibrate, 125
- calibrate.methods, 126
- calibrate.qrrvglm, 126
- calibrate.qrrvglm.control, 129
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cao</td>
<td>130</td>
</tr>
<tr>
<td>cao.control</td>
<td>134</td>
</tr>
<tr>
<td>cardioid</td>
<td>139</td>
</tr>
<tr>
<td>cauchit</td>
<td>141</td>
</tr>
<tr>
<td>cauchy</td>
<td>143</td>
</tr>
<tr>
<td>cdf.lmscreg</td>
<td>145</td>
</tr>
<tr>
<td>cens.gumbel</td>
<td>146</td>
</tr>
<tr>
<td>cens.normal</td>
<td>148</td>
</tr>
<tr>
<td>cens.poisson</td>
<td>150</td>
</tr>
<tr>
<td>cgo</td>
<td>153</td>
</tr>
<tr>
<td>chisq</td>
<td>156</td>
</tr>
<tr>
<td>clo</td>
<td>157</td>
</tr>
<tr>
<td>cloglog</td>
<td>158</td>
</tr>
<tr>
<td>Coef</td>
<td>161</td>
</tr>
<tr>
<td>Coef.qrrvglm</td>
<td>162</td>
</tr>
<tr>
<td>Coef.rrvglm</td>
<td>166</td>
</tr>
<tr>
<td>Coef.vlm</td>
<td>168</td>
</tr>
<tr>
<td>coefvlm</td>
<td>169</td>
</tr>
<tr>
<td>concoef</td>
<td>176</td>
</tr>
<tr>
<td>concoef-methods</td>
<td>178</td>
</tr>
<tr>
<td>constraints</td>
<td>178</td>
</tr>
<tr>
<td>cpo</td>
<td>181</td>
</tr>
<tr>
<td>cratio</td>
<td>190</td>
</tr>
<tr>
<td>cumulative</td>
<td>191</td>
</tr>
<tr>
<td>dagum</td>
<td>196</td>
</tr>
<tr>
<td>deplot.lmscreg</td>
<td>201</td>
</tr>
<tr>
<td>depvar</td>
<td>202</td>
</tr>
<tr>
<td>df.residual</td>
<td>203</td>
</tr>
<tr>
<td>dirichlet</td>
<td>207</td>
</tr>
<tr>
<td>dirmul.old</td>
<td>209</td>
</tr>
<tr>
<td>dirmultinomial</td>
<td>211</td>
</tr>
<tr>
<td>double.cens.normal</td>
<td>214</td>
</tr>
<tr>
<td>double.expbinomial</td>
<td>216</td>
</tr>
<tr>
<td>erlang</td>
<td>222</td>
</tr>
<tr>
<td>expexpff</td>
<td>230</td>
</tr>
<tr>
<td>expexpff1</td>
<td>232</td>
</tr>
<tr>
<td>expgeometric</td>
<td>235</td>
</tr>
<tr>
<td>explink</td>
<td>238</td>
</tr>
<tr>
<td>explogff</td>
<td>241</td>
</tr>
<tr>
<td>expoexponential</td>
<td>242</td>
</tr>
<tr>
<td>expoexponential</td>
<td>242</td>
</tr>
<tr>
<td>expoexponential</td>
<td>242</td>
</tr>
<tr>
<td>feli</td>
<td>247</td>
</tr>
<tr>
<td>fff</td>
<td>249</td>
</tr>
<tr>
<td>fill</td>
<td>250</td>
</tr>
<tr>
<td>fisherz</td>
<td>255</td>
</tr>
<tr>
<td>fisk</td>
<td>257</td>
</tr>
<tr>
<td>fittedvlm</td>
<td>259</td>
</tr>
<tr>
<td>foldnormal</td>
<td>263</td>
</tr>
<tr>
<td>foldedsqrt</td>
<td>265</td>
</tr>
<tr>
<td>frechet</td>
<td>270</td>
</tr>
<tr>
<td>freund61</td>
<td>271</td>
</tr>
<tr>
<td>gamma</td>
<td>274</td>
</tr>
<tr>
<td>gamma2</td>
<td>275</td>
</tr>
<tr>
<td>gammahyperbola</td>
<td>277</td>
</tr>
<tr>
<td>gammar</td>
<td>278</td>
</tr>
<tr>
<td>gammaR</td>
<td>280</td>
</tr>
<tr>
<td>gaussianff</td>
<td>282</td>
</tr>
<tr>
<td>genbetaII</td>
<td>285</td>
</tr>
<tr>
<td>gengamma.stacy</td>
<td>288</td>
</tr>
<tr>
<td>genpoisson</td>
<td>291</td>
</tr>
<tr>
<td>genrayleigh</td>
<td>294</td>
</tr>
<tr>
<td>geometric</td>
<td>296</td>
</tr>
<tr>
<td>get.smart</td>
<td>297</td>
</tr>
<tr>
<td>get.smart.prediction</td>
<td>298</td>
</tr>
<tr>
<td>gev</td>
<td>299</td>
</tr>
<tr>
<td>golf</td>
<td>305</td>
</tr>
<tr>
<td>gompertz</td>
<td>308</td>
</tr>
<tr>
<td>gpd</td>
<td>310</td>
</tr>
<tr>
<td>grc</td>
<td>316</td>
</tr>
<tr>
<td>gumbel</td>
<td>321</td>
</tr>
<tr>
<td>gumbelII</td>
<td>326</td>
</tr>
<tr>
<td>guplot</td>
<td>329</td>
</tr>
<tr>
<td>hatvalues</td>
<td>331</td>
</tr>
<tr>
<td>huber2</td>
<td>336</td>
</tr>
<tr>
<td>hyperg</td>
<td>342</td>
</tr>
<tr>
<td>hypersecant</td>
<td>343</td>
</tr>
<tr>
<td>hzeta</td>
<td>346</td>
</tr>
<tr>
<td>identitylink</td>
<td>349</td>
</tr>
<tr>
<td>inv.binomial</td>
<td>350</td>
</tr>
<tr>
<td>inv.gaussianff</td>
<td>353</td>
</tr>
<tr>
<td>inv.lomax</td>
<td>356</td>
</tr>
<tr>
<td>inv.paralogistic</td>
<td>359</td>
</tr>
<tr>
<td>is.buggy</td>
<td>361</td>
</tr>
<tr>
<td>is.parallel</td>
<td>362</td>
</tr>
<tr>
<td>is.smart</td>
<td>363</td>
</tr>
<tr>
<td>is.zero</td>
<td>364</td>
</tr>
<tr>
<td>kumar</td>
<td>368</td>
</tr>
<tr>
<td>laplace</td>
<td>371</td>
</tr>
<tr>
<td>latvar</td>
<td>375</td>
</tr>
<tr>
<td>leipnik</td>
<td>376</td>
</tr>
<tr>
<td>levy</td>
<td>380</td>
</tr>
<tr>
<td>lgamma1</td>
<td>381</td>
</tr>
<tr>
<td>lindley</td>
<td>386</td>
</tr>
<tr>
<td>linkfun</td>
<td>387</td>
</tr>
<tr>
<td>linkfun.vglm</td>
<td>388</td>
</tr>
<tr>
<td>lino</td>
<td>393</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>lms.bcg</td>
<td>396</td>
</tr>
<tr>
<td>lms.bcn</td>
<td>398</td>
</tr>
<tr>
<td>lms.yjn</td>
<td>401</td>
</tr>
<tr>
<td>logc</td>
<td>406</td>
</tr>
<tr>
<td>loge</td>
<td>407</td>
</tr>
<tr>
<td>logF</td>
<td>409</td>
</tr>
<tr>
<td>logff</td>
<td>410</td>
</tr>
<tr>
<td>logistic</td>
<td>412</td>
</tr>
<tr>
<td>logit</td>
<td>413</td>
</tr>
<tr>
<td>loglaplace</td>
<td>416</td>
</tr>
<tr>
<td>logLikNvlm</td>
<td>421</td>
</tr>
<tr>
<td>loglinb2</td>
<td>422</td>
</tr>
<tr>
<td>loglinb3</td>
<td>424</td>
</tr>
<tr>
<td>loglog</td>
<td>426</td>
</tr>
<tr>
<td>lognormal</td>
<td>427</td>
</tr>
<tr>
<td>logoff</td>
<td>428</td>
</tr>
<tr>
<td>lomax</td>
<td>431</td>
</tr>
<tr>
<td>lqnorm</td>
<td>433</td>
</tr>
<tr>
<td>lvplot</td>
<td>436</td>
</tr>
<tr>
<td>lvplot.qrrvglm</td>
<td>438</td>
</tr>
<tr>
<td>lvplot.rrvglm</td>
<td>442</td>
</tr>
<tr>
<td>makeham</td>
<td>447</td>
</tr>
<tr>
<td>margeff</td>
<td>449</td>
</tr>
<tr>
<td>Max</td>
<td>452</td>
</tr>
<tr>
<td>maxwell</td>
<td>454</td>
</tr>
<tr>
<td>mccullagh89</td>
<td>456</td>
</tr>
<tr>
<td>meplot</td>
<td>458</td>
</tr>
<tr>
<td>micmen</td>
<td>460</td>
</tr>
<tr>
<td>mix2exp</td>
<td>462</td>
</tr>
<tr>
<td>mix2normal</td>
<td>464</td>
</tr>
<tr>
<td>mix2poisson</td>
<td>466</td>
</tr>
<tr>
<td>MNSs</td>
<td>468</td>
</tr>
<tr>
<td>multilogit</td>
<td>474</td>
</tr>
<tr>
<td>multinomial</td>
<td>476</td>
</tr>
<tr>
<td>nakagami</td>
<td>481</td>
</tr>
<tr>
<td>nbcanlink</td>
<td>483</td>
</tr>
<tr>
<td>nbolff</td>
<td>485</td>
</tr>
<tr>
<td>negbinomial</td>
<td>487</td>
</tr>
<tr>
<td>negbinomial.size</td>
<td>493</td>
</tr>
<tr>
<td>normal.vcm</td>
<td>494</td>
</tr>
<tr>
<td>npparam.vlm</td>
<td>498</td>
</tr>
<tr>
<td>Opt</td>
<td>500</td>
</tr>
<tr>
<td>ordpoisson</td>
<td>502</td>
</tr>
<tr>
<td>paralognistic</td>
<td>506</td>
</tr>
<tr>
<td>paretoff</td>
<td>509</td>
</tr>
<tr>
<td>paretoIv</td>
<td>513</td>
</tr>
<tr>
<td>perks</td>
<td>517</td>
</tr>
<tr>
<td>perspqr</td>
<td>519</td>
</tr>
<tr>
<td>plotdeplot.lmscreg</td>
<td>524</td>
</tr>
<tr>
<td>plotqrrvglm</td>
<td>526</td>
</tr>
<tr>
<td>plotqtpplot.lmscreg</td>
<td>527</td>
</tr>
<tr>
<td>plotvgam</td>
<td>531</td>
</tr>
<tr>
<td>plotvgam.control</td>
<td>534</td>
</tr>
<tr>
<td>plotvglm</td>
<td>535</td>
</tr>
<tr>
<td>poisson.points</td>
<td>538</td>
</tr>
<tr>
<td>poissonff</td>
<td>539</td>
</tr>
<tr>
<td>polf</td>
<td>543</td>
</tr>
<tr>
<td>posbernoulli.b</td>
<td>547</td>
</tr>
<tr>
<td>posbernoulli.t</td>
<td>549</td>
</tr>
<tr>
<td>posbernoulli.tb</td>
<td>552</td>
</tr>
<tr>
<td>posbinomial</td>
<td>559</td>
</tr>
<tr>
<td>posnegbinomial</td>
<td>564</td>
</tr>
<tr>
<td>posnormal</td>
<td>568</td>
</tr>
<tr>
<td>pospoisson</td>
<td>571</td>
</tr>
<tr>
<td>powerlink</td>
<td>572</td>
</tr>
<tr>
<td>predictqrrvglm</td>
<td>575</td>
</tr>
<tr>
<td>predictvglm</td>
<td>576</td>
</tr>
<tr>
<td>prentice74</td>
<td>578</td>
</tr>
<tr>
<td>probit</td>
<td>581</td>
</tr>
<tr>
<td>propodds</td>
<td>583</td>
</tr>
<tr>
<td>prplot</td>
<td>584</td>
</tr>
<tr>
<td>put.smart</td>
<td>585</td>
</tr>
<tr>
<td>qrrvglm.control</td>
<td>586</td>
</tr>
<tr>
<td>qtpplot.gumbel</td>
<td>592</td>
</tr>
<tr>
<td>qtpplot.lmscreg</td>
<td>594</td>
</tr>
<tr>
<td>quasibinomial.ff</td>
<td>595</td>
</tr>
<tr>
<td>quasipoissonff</td>
<td>597</td>
</tr>
<tr>
<td>qvar</td>
<td>599</td>
</tr>
<tr>
<td>qvar</td>
<td>602</td>
</tr>
<tr>
<td>rayleigh</td>
<td>605</td>
</tr>
<tr>
<td>rec.expl</td>
<td>614</td>
</tr>
<tr>
<td>rec.normal</td>
<td>615</td>
</tr>
<tr>
<td>reciprocal</td>
<td>616</td>
</tr>
<tr>
<td>rhobit</td>
<td>618</td>
</tr>
<tr>
<td>riceff</td>
<td>620</td>
</tr>
<tr>
<td>rigff</td>
<td>622</td>
</tr>
<tr>
<td>rlplot.egev</td>
<td>623</td>
</tr>
<tr>
<td>rrar</td>
<td>625</td>
</tr>
<tr>
<td>rrvglm</td>
<td>627</td>
</tr>
<tr>
<td>rrvglm.control</td>
<td>633</td>
</tr>
<tr>
<td>rrvglm.optim.control</td>
<td>636</td>
</tr>
<tr>
<td>s</td>
<td>638</td>
</tr>
<tr>
<td>sc.studentt2</td>
<td>639</td>
</tr>
<tr>
<td>Select</td>
<td>641</td>
</tr>
<tr>
<td>seq2binomial</td>
<td>643</td>
</tr>
<tr>
<td>setup.smart</td>
<td>645</td>
</tr>
</tbody>
</table>
INDEX

simplex, 647
sinmad, 652
skellam, 655
skewnormal, 658
slash, 661
smart.expression, 663
smart.mode.is, 664
smartpred, 665
sratio, 667
studentt, 669
SURff, 671
tikuv, 678
tobit, 681
Tol, 685
triangle, 689
trplot, 691
trplot.qrrvglm, 692
truncweibull, 696
uninormal, 699
vgam, 703
vgam-class, 707
VGAM-package, 11
vgam.control, 710
vglm, 712
vglm.control, 720
vonnines, 726
vsmooth.spline, 728
waldff, 732
weibullR, 733
weightsvgam, 736
wrapup.smart, 738
eyo.johnson, 739
yip88, 740
yulesimon, 744
zabinomial, 746
zageometric, 750
zanegbinomial, 753
zapoissson, 757
zero, 759
zetaaff, 764
zabinomial, 767
zageometric, 771
zanegbinomial, 774
zipebcom, 777
zipf, 781
zipoisssoon, 784

*Topic smooth
plotvgam, 531
plotvgam.control, 534
plotvglm, 535
s, 638
vgam, 703
vgam-class, 707
vgam.control, 710
vsmooth.spline, 728

*Topic survival
SurvS4, 673

*Topic ts
rrar, 625
.Random.seed, 610
[.SurvS4(SurvS4), 673

A1A2A3, 14, 16, 17, 19, 469
AA.Aa.aa, 15, 15, 17, 19, 469
AB.Ab.aB.ab, 15, 16, 17, 19, 469
ABO, 15–17, 18, 469
acat, 19, 172, 191, 193, 194, 362, 478, 669
AIC, 22, 75, 422
AICqrrvglm (AICvlm), 21
AICrrvgam (AICvlm), 21
AICrrvglm (AICvlm), 21
AICvgam (AICvlm), 21
AICvlm, 21, 74, 75, 499, 551, 559, 560
alaplace, 23
alaplace1, 30, 32, 35, 37, 400, 416–418, 650
alaplace1 (alaplace), 23
alaplace2, 318, 373, 650
alaplace2 (alaplace), 23
alaplace3, 28, 420
alaplace3 (alaplace), 23
alaplaceUC, 27
alclevels (crashes), 188
alcoff, 319, 473, 474
alcoff (crashes), 188
amlbinomial, 29, 32, 34, 35, 37, 99
amlexponential, 30, 31, 32, 35, 243, 397
amlnormal, 25, 30, 32, 33, 37, 225, 400, 403
amloisson, 30, 32, 34, 35, 36, 541
anova, 435
AR1, 38, 199, 700
arima.sim, 39
array, 675
as.character.SurvS4(SurvS4), 673
as.data.frame.SurvS4(SurvS4), 673
atanh, 256
auuc, 40, 319
aux.posbernoulli.t, 41, 548
cauchy1 (cauchy), 143
cbind, 178, 642
cdf.lm.sreg, 145, 397, 400, 403
cenormal (cens.normal), 148
cens.gumbel, 146, 323
cens.normal, 148, 215, 682, 683, 700
cens.poisson, 150, 541, 675
cens.rayleigh (rayleigh), 605
cfibrosis, 152, 644
cgo, 153
chest.nz, 154, 451
chinese.nz, 155
chisq, 156, 650
Chisquare, 157
choose, 209, 212
clo, 157
cloglog, 90, 92, 98, 132, 142, 158, 159, 181, 191, 192, 194, 280, 390, 407, 415, 416, 495, 582, 597, 669, 778, 779
coalminers, 92, 96, 160
Coef, 133, 161, 169, 375, 436, 452, 500, 685, 691
coeff, 161, 162, 168–170, 177
Coef.qrrvglm, 129, 130, 162, 163, 165, 185, 438, 440, 441, 519, 521, 575, 591
Coef.qrrvglm-class, 164
Coef.rrvglm, 162, 166, 444, 629
Coef.rrvglm-class, 167
Coef.vlm, 162, 168
coef.vlm, 169
colMeans, 553
colnames, 319
concoef, 133, 176, 177
concoef, cao-method (concoef-methods), 178
concoef, Coef. cao-method (concoef-methods), 178
concoef, Coef.qrrvglm-method (concoef-methods), 178
concoef, Coef.rrvglm-method (concoef-methods), 178
concoef, qrrvglm-method (concoef-methods), 178
cor, 366
corbet, 180, 565
crashbc (crashes), 188
crashes, 188
crashf (crashes), 188
crashi, 319, 474, 629
crashl (crashes), 188
crashmc (crashes), 188
crashp (crashes), 188
crasht (crashes), 188
cratio, 20, 172, 190, 193, 194, 362, 478, 668, 669
cumsum, 459
cut, 502, 608, 610, 611
Dagum, 195, 198
dagum, 63, 195, 196, 196, 259, 287, 357, 360, 432, 507, 650, 654
dalap, 420
dalap (alaplaceUC), 27
dAR1, 39, 198
data.frame, 470
dbenf (Benford), 45
dbenini (Benini), 46
dbetabinom (BinomBeta), 49
dbetageom (Betageom), 59
dbetanorm (Betanorm), 63
dbiamhcop (Biamhcop), 68
dbiclaytoncop, 73
dbiclaytoncop (Biclaytoncop), 71
dbifgmcop (Bifgmcop), 75
dbifrankcop (Frank), 267
dbilogis (bilogis), 84
dbinom, 557, 745, 748, 766
dbinom2.or (Binom2.or), 88
dbinom2.rho (Binom2.rho), 93
dbinorm (Binorm), 100
dbinormcop (Binormcop), 105
dbiplackcop (Biplackett), 106
dbisa (Bisa), 110
dbistudentt, 115
dbistudentt (Bistudentt), 113
dbort (Bort), 118
dcard (Card), 138
ddagum (Dagum), 195
ddearmice, 42, 200, 548, 551, 554
dexp, 32, 225, 229
dexp (Expectiles-Exponential), 223
denorm, 30, 35, 224, 229, 400
denorm (Expectiles-Normal), 224
deplot.lmscreg, 146, 201, 397, 400, 403, 525, 595
depvar, 162, 202
deninf, 223–225
deninf (Expectiles-Uniform), 228
deviance, 204
dexp, 224
dexpgeom, 236
dexpgeom (expgeom), 234
dexplog, 242
dexplog (explog), 239
dexpois, 246
dexpois (expois), 244
df.residual, 203
df.residual_vlm (df.residual), 203
dfbeta (hatvalues), 331
dfbetavlm (hatvalues), 331
dfelix, 248
dfelix (Felix), 246
dfisk (Fisk), 256
dfoldnorm (Foldnorm), 262
dfrchbet (Frechet), 268
dgenbetaII, 287
dgenbetaII (GenbetaII), 284
dgengamma.stacy (gengammaUC), 290
dgenpois, 204, 292
dgenray, 295
dgenray (genray), 293
dgeom, 749, 751, 769, 770
dgev, 300
dgev (gevUC), 302
dgompertz, 309, 329
dgompertz (Gompertz), 307
dgpd, 311, 312
dgpd (gpdUC), 313
dgumbel, 308, 325
dgumbel (gumbelUC), 327
dgumbelII, 323, 327
dgumbelII (Gumbel-II), 324
dhuber, 206
dhyper, 342, 343
dhzeta, 347
dhzeta (Hzeta), 345
digamma, 579
dinv.gaussian (Inv.gaussian), 352
dinv.lomax (Inv.lomax), 355
dinv.paralogistic (Inv.paralogistic), 358
drichtlet, 207, 210, 213, 219, 349, 478, 613, 648, 650
dirmul.old, 209, 213
dirmultinomial, 53, 56, 208, 210, 211, 478
dkumar, 369
dkumar (Kumar), 366
dlaplace (laplaceUC), 373
dlglm (lgammaUC), 383
dlind, 387
dlind (Lindley), 385
dlino (Lino), 392
dlog (Log), 404
dlogF, 213, 410
INDEX

**dlogis.** 412  
**dloglap.** 418  
**dloglap (loglapUC).** 419  
**dlomax (Lomax).** 429  
**dmakeham.** 308, 447, 448  
**dmakeham (Makeham).** 446  
**dmaxwell (Maxwell).** 453  
**dnaka (Nakagami).** 480  
**dnbinom.** 563, 752, 755, 773  
**dnorm.** 225, 263, 568, 700  
**double.cens.normal.** 149, 214, 616, 683, 700  
**double.expbinomial.** 99, 216, 687  
**dparalogistic (Paralogistic).** 505  
**dpareto (Pareto).** 508  
**dparetoI (ParetoIV).** 511  
**dparetoII (ParetoIV).** 511  
**dparetoIII (ParetoIV).** 511  
**dparetoIV (ParetoIV).** 511  
**dperks.** 518  
**dperks (Perks).** 516  
**dpois.** 205, 292, 467, 542, 655, 656, 783  
**dpois.points (PoissonPoints).** 542  
**dpolono (Polono).** 545  
**dposbern.** 548, 551, 558  
**dposbern (posbernUC).** 555  
**dposbinom (Posbinom).** 557  
**dposgeom (Posgeom).** 561  
**dposnegbin (Posnegbin).** 562  
**dposnorm (Posnorm).** 566  
**dpospois.** 783  
**dpospois (Pospois).** 569  
**drayleigh (Rayleigh).** 603  
**drice.** 621  
**drice (Rice).** 619  
**dsc.t2.** 229, 640  
**dsc.t2 (Expectiles-sc.t2).** 226  
**dssimplex.** 648  
**dssimplex (Simplex).** 646  
**dsinmad (Sinmad).** 651  
**dskellam.** 656  
**dskellam (Skellam).** 654  
**dskewnorm (skewnorm).** 657  
**dslash (Slash).** 660  
**dt.** 113, 114, 227, 412  
**dtikuv.** 679  
**dtikuv (Tikuv).** 676  
**dtobit.** 681  
**dtobit (Tobit).** 679  
**dtriangle (Triangle).** 688  
**dtruncpareto (Truncpareto).** 694  
**ducklings.** 218  
**dunif.** 228, 229  
**dweibull.** 696, 697, 735  
**dyules.** 744  
**dyules (Yules).** 743  
**dzabinom.** 748  
**dzabinom (Zabinom).** 745  
**dzageom.** 751  
**dzageom (Zageom).** 748  
**dzanegeb.** 755  
**dzanegeb (Zanegeb).** 752  
**dzapois (Zapois).** 756  
**dzeta.** 765  
**dzeta (Zeta).** 761  
**dzinegbin (Zinegbin).** 773  
**dzipf.** 782  
**dzipf (Zipf).** 780  
**dzipois.** 742, 757  
**dzipois (Zipois).** 782  

**edhuber (dhuber).** 206  
**eexp (Expectiles-Exponential).** 223  
**egev.** 303, 323, 350, 625  
**egev (gev).** 299  
**egumbel.** 148, 301, 329, 330  
**egumbel (gumbel).** 321  
**enorm (Expectiles-Normal).** 224  
**enzyme.** 219, 461  
**erf.** 220  
**erfc (erf).** 220  
**erlang.** 222, 650  
**eunif (Expectiles-Uniform).** 228  
**exp.** 237, 371, 406  
Expectiles-Exponential, 223  
Expectiles-Normal, 224  
Expectiles-sc.t2, 226  
Expectiles-Uniform, 228  
expexpf, 230, 233, 276, 279, 735  
expexpff, 231, 232  
expexppint (expint), 237  
expgeom, 234  
expgeometric, 234, 235, 235, 243, 297  
expint, 237  
explink, 238, 408, 600–603, 699
INDEX

hyperg, 342
hypersecant, 214, 343
hypersecant01 (hypersecant), 343
Hzeta, 345, 347
hzeta, 346, 346, 650, 765
I, 638
iam, 348
identity, 390, 495, 496, 540, 617, 621
identitylink, 349
influence.measures, 332
integrate, 545, 546, 660, 661
inv.binomial, 350, 491, 541
Inv.gaussian, 352, 354
inv.gaussianff, 112, 352, 353, 353, 732, 733
Inv.lomax, 355
inv.lomax, 63, 198, 259, 287, 355, 356, 356,
357, 357, 532, 507, 650, 654
Inv.paralogistic, 358, 360
inv.paralogistic, 63, 198, 259, 287,
357–359, 359, 432, 507, 650, 654
iris, 478
is.buggy, 361, 639, 705, 706
is.na.SurvS4 (SurvS4), 673
is.parallel, 179, 362
is.smart, 363
is.SurvS4 (SurvS4), 673
is.zero, 179, 364
kendall.tau, 73, 104, 365
Kumar, 366
kumar, 358, 68, 367, 680, 650
lakeoO, 369
lambertW, 370, 408, 446, 447
laplace, 25, 243, 337, 371, 374
laplaceUC, 373
latvar, 133, 375, 375, 437
leipnik, 376, 457
lerch, 378, 763
leukemia, 379, 675
levy, 380
lgamma, 342, 383, 580
lgamma1, 275, 381, 384, 650
lgamma3, 384, 397, 580, 650
lgamma3 (lgamma1), 381
lgammaUC, 383
Lindley, 385
lindley, 273, 385, 386, 650
linkfun, 387, 389, 391
linkfun.vglm, 387, 388, 716
Links, 12, 14, 15, 17–19, 23, 38, 48, 51, 54,
60, 62, 65, 67, 69, 72, 77, 78, 80, 81,
83, 86, 90, 95, 97, 99, 102, 104, 108,
111, 114, 117, 139, 141, 143, 147,
148, 150, 151, 159, 171, 175, 190,
192, 194, 197, 207, 209, 211, 214,
216, 222, 230, 232, 236, 238, 239,
242, 245, 248, 249, 255, 256, 258,
263, 265, 266, 270, 272, 274, 275,
277, 279, 286, 288, 291, 294, 296,
299, 305, 306, 309, 310, 322, 326,
337, 342, 344, 347, 349–351, 354,
357, 359, 368, 372, 376, 380, 382,
386, 389, 393, 399, 406–410, 412,
414–416, 426–429, 431, 433, 447,
455, 456, 460, 462, 464, 466, 469,
475, 482–486, 488, 506, 509, 514,
517, 538, 540, 541, 543, 544, 564,
568, 571–573, 579, 581, 582, 595,
598, 605, 614, 615, 617–619, 621,
622, 640, 643, 644, 648, 652, 656,
658, 662, 667, 669, 678, 681, 689,
696, 699, 726, 732, 733, 741, 744,
747, 750, 753, 754, 757, 764, 767,
771, 775, 777, 781, 785
Lino, 392, 394
lino, 287, 392, 393, 650
lirat, 53, 56, 395
lm, 34, 170, 204, 298, 299, 496, 645, 646, 665
lms.bcg, 32, 146, 202, 396, 400, 403, 595
lms.bcn, 25, 35, 146, 202, 225, 396, 397, 398,
402, 403, 458, 595
lms.yjn, 146, 201, 202, 297, 400, 401, 528,
594, 595, 740
lms.yjn2 (lms.yjn), 401
Log, 404
log, 237, 371, 408, 411
log10, 261
log1p, 405, 406
loglpexp, 405
logc, 390, 406, 408
log, 142, 239, 280, 350, 390, 394, 407, 407,
411, 415, 417, 427, 429, 484, 495,
540, 541, 573, 599
logF, 213, 409
logff, 404, 405, 410, 410, 565, 650
logistic, 87, 412, 414, 650, 670
logistic1, 194, 412, 415, 650
logistic1(logistic), 412
logit, 92, 98, 132, 142, 159, 181, 191, 194,
256, 280, 350, 390, 408, 413, 416, 475, 495, 582, 597, 669
logitlaplace, 417
logitlaplace1 (loglaplace), 416
loglaplace, 416
loglaplace1, 420
loglaplace1(loglaplace), 416
loglapUC, 419
logLik, 435
logLik.vlm, 421
loglinb2, 92, 96, 422, 425
loglinb3, 423, 424
loglog, 347, 390, 407, 408, 426, 488, 495,
669, 764
logneg (log), 407
Lognormal, 545
lognormal, 427, 546, 650, 735
logoff, 299, 311, 390, 407, 408, 411, 427,
428, 457, 495
Lomax, 429, 432
lomax, 63, 198, 259, 287, 357, 360, 430, 431,
507, 650, 654
lqnorm, 284, 433
lrtest, 435
lrtest_vglm, 716
lrtest_vglm (lrtest), 435
lv, 375
lv (latvar), 375
lvplot, 109, 133, 185, 376, 436, 441, 444, 691
lvplot.qrrvglm, 163, 164, 185, 437, 438,
520, 521, 527, 591
lvplot.rrvglm, 442, 444, 629, 632

machinists, 445
Makeham, 446
makeham, 309, 446, 447, 447, 650
margeff, 194, 449, 478
marital.nz, 451
match.call, 663
Math.SurvS4 (SurvS4), 673
matrix, 88, 93, 222, 493, 675
Max, 133, 452, 501, 686
Maxwell, 453, 455, 542
maxwell, 453, 454, 454, 538, 539, 604, 606,
735

mccullagh89, 377, 456
median, 373
medpolish, 318, 319
medmaxtemp, 457
meplot, 312, 458
micmen, 220, 460, 714
mix2exp, 423, 462
mix2normal, 464, 467, 700
mix2poisson, 463, 465, 466, 541
MNSs, 15–17, 19, 468
model.frame, 470
model.frame.vlm, 470, 471, 472
model.matrix, 203, 471, 472
model.matrix.default, 132, 713
model.matrix.vlm, 470, 471
moffset, 319, 473, 473, 531, 607
multilogit, 171, 175, 387–389, 415, 474,
478, 495
multinom, 478
Multinomial, 478

multinomial, 20, 121, 122, 191, 193, 194,
208, 210, 212, 213, 251, 319, 349,
362, 443, 449, 450, 475, 476, 668,
669

Nakagami, 480
nakagami, 480, 481, 481
nbcanlink, 483, 491, 494
nbglm, 192, 194, 306, 390, 485, 491, 544

nef.h5 (hypersecant), 343
negbinomial, 133, 173, 181, 185, 296, 297,
318, 352, 445, 483–486, 487, 493,
494, 546, 564, 565, 588, 589, 591,
599, 609, 611, 629, 650, 754, 755,
773, 775, 776
negbinomial.size, 483, 484, 491, 493, 650
negidentity, 390
negidentity (identitylink), 349
negloge, 276, 279, 390, 488, 489, 514
negloge (log), 407
negreciprocal, 390
negreciprocal (reciprocal), 616
Normal, 64, 465
normal.vcm, 171, 175, 475, 494, 700
normal1 (uninormal), 699
nparam (nparam.vlm), 498
nparam.vlm, 498
ns, 251, 301, 312, 361, 436, 639, 665, 666
INDEX

oldClass, 675
olym08, 319
olym08 (olympics), 499
olym12, 319
olym12 (olympics), 499
olympics, 499
Ops.Surv54 (Surv54), 673
Opt, 133, 452, 500, 686
optim, 127, 129, 131, 135, 136, 588, 589, 591, 636, 637
options, 131, 182, 713
order, 165
ordered, 20, 190, 193, 503, 583, 668
ordpoisson, 502, 541, 544
oxtemp, 301, 504
palap (alaplaceUC), 27
par, 330, 331, 438–441, 443, 444, 458, 510–521, 525, 526, 528, 530, 585, 592, 593, 623, 624, 693, 694
Paralogistic, 505, 507
paralogistic, 63, 198, 259, 287, 357, 360, 432, 505, 506, 506, 650, 654
Pareto, 508, 511, 513
paretoff, 312, 508, 509, 509, 514, 515
ParetoI (ParetoIV), 511
ParetoII (ParetoIV), 511
paretoII (paretoIV), 513
ParetoIII (ParetoIV), 511
paretoIII (paretoIV), 513
ParetoIV, 509, 511, 515
paretoV, 510–513, 513
pbefn (Benford), 45
penini (Benini), 46
pbtbinom (Betabinom), 49
pbtgeom (Betageom), 59
pbtlnorm (Btlnorm), 63
pbtmhcop (Btmhcop), 68
pbfmcop (Bfmcop), 75
pbfrcop (Franc), 267
pblap (bilateral), 84
pbnorm, 765
pbnorm (Binorm), 104
pbnorm (Binorm), 100
pbnormcop (Binnormcop), 105
pbplackcop (Biplackett), 106
pbisa, 112
pbisa (Bisa), 110
pcard (Card), 138
pdagum (Dagum), 195
peeexp (Expectiles-Exponential), 223
penorm (Expectiles-Normal), 224
Perks, 516
perks, 516, 517, 650
persp, 520, 521
persqrrvglm, 185, 441, 519, 691
peunif (Expectiles-Uniform), 228
pexp, 223
pexpgeom (expgeom), 234
pexpl (explog), 239
pexpois (expois), 244
pfisk (Fisk), 256
pfoldnorm (Foldnorm), 262
pfrechet (Frechet), 268
pgamma, 522–524
pgamma.deriv, 522, 523, 524, 697
pgamma.deriv.unscaled, 523, 523, 697
pgengamma.stacy (gengammaUC), 290
pgenray (genray), 293
pgiev (gevUC), 302
pgompertz (Gompertz), 307
pgpd (gpudUC), 313
pgumbel (gumbelUC), 327
pgumbellII (Gumbel-II), 324
phuber (dhuber), 206
phzeta (Hzeta), 345
pinv.gaussian (Inv.gaussian), 352
pinv.lomax (Inv.lomax), 355
pinv.paralogistic (Inv.paralogistic), 358
pkumar (Kumar), 366
plaplace (laplaceUC), 373
plgamma (lgammaUC), 383
plindley (Lindley), 385
plino (Lino), 392
pllog (Log), 404
plloglap (logplogU), 419
plomax (Lomax), 429
plot, 330, 458, 520, 530, 693
plot.vgam (plotvgam), 531
plot.window, 530
plotdeplot.lmscreg, 201, 202, 524
plotqrrvglm, 526
plotqtrplot.lmscreg, 527, 594, 595
plotrcim, 319, 473, 474, 529, 607
plotvgam, 531, 535, 536, 631, 706, 708, 715, 718
INDEX

pzigeom (Zigeom), 769
pzinegbin (Zinegbin), 773
pzipf (Zipf), 780
pzipois (Zipois), 782

qalap (alaplaceUC), 27
qbenf (Benford), 45
qbenini (Benini), 46
qbetanorm (Betanorm), 63
qbisa (Bisa), 110
qcddagum (Cddagum), 195
qqexp (Expectiles-Exponential), 223
qgen (Exexpectiles-Normal), 224
qgenif (Expectiles-Uniform), 228
qexp, 223
qexpgeom (expgeom), 234
qexplog (explog), 239
qexpopois (expopois), 244
qfisk (Fisk), 256
qfoldnorm (Foldnorm), 262
qfrechet (Frechet), 268
qgengamma.stacy (gengammaUC), 290
qgenray (genray), 293
qgev (gevUC), 302
qgompertz (Gompertz), 307
qgpd (gpdUC), 313
qgumbel (gumbelUC), 327
qgumbelIII (Gumbel-III), 324
qhuber (huber), 206
qhzeta (Hzeta), 345
qinv.lomax (Inv.lomax), 355
qinv.paralogistic (Inv.paralogistic), 358
qkumar (Kumar), 366
qlaplace (laplaceUC), 373
qlgamma (lgammaUC), 383
qlino (Lino), 392
qlloglap (loglapUC), 419
qlomax (Lomax), 429
qmakeham (Makeham), 446
qmaxwell (Maxwell), 453
qnaka (Nakagami), 480
qnb (Nbinom), 488, 491
qnorm, 28, 45, 47, 110, 138, 195, 206, 225,
        244, 256, 262, 290, 293, 307, 325,
        345, 355, 358, 367, 373, 384, 385,
        392, 420, 430, 446, 453, 480, 505,
        508, 512, 516, 604, 619, 651, 660,
        677, 688, 695, 743, 780
qparalogistic (Paralogistic), 505
qpareto (Pareto), 508
qparetoI (ParetoIV), 511
qparetoII (ParetoIV), 511
qparetoIII (ParetoIV), 511
qparetoIV, 514
qparetoIV (ParetoIV), 511
qperks (Perks), 516
qposbinom (Posbinom), 557
qposgeom (Posgeom), 561
qposnegbin (Posnegbin), 562
qposnorm (Posnorm), 566
qpospois (Pospois), 569
qrayleigh (Rayleigh), 603
qrice (Rice), 619
qrrvglm.control, 135, 136, 182–185, 275,
        488, 586, 610, 611
qsc.t2 (Expectiles-sc.t2), 226
qsinmad (Sinmad), 651
qt, 226
qtikuv (Tikuv), 676
qtobit (Tobit), 679
qtplot.gumbel (qtplot.gumbel), 592
qtplot.gumbel, 592
qtplot.lmscreg, 146, 202, 397, 400, 403,
        529, 594
qtriangle (Triangle), 688
qtruncpareto (Truncpareto), 694
quantile, 173, 462, 464, 467, 489, 662
quasibinomial, 174, 597
quasibinomialff, 99, 181, 183, 541, 595,
        598, 599
quasipoisson, 599
quasipoissonff, 181, 183, 489–491, 494,
        541, 597, 597
qunif, 228, 269, 302, 314, 328
Qvar, 239, 319, 599, 603, 700
qvar, 601, 602
qzabinom (Zabinom), 745
qzageom (Zageom), 748
qzanegbin (Zanegbin), 752
qzapos (Zapos), 756
qzbinom (Zbinom), 765
qzigeom (Zigeom), 769
qzinegbin (Zinegbin), 773
qzipois (Zipois), 782
ralap, 25
ralap(alaplaceUC), 27
Rayleigh, 454, 603, 606
rayleigh, 294, 295, 454, 455, 482, 538, 539, 604, 605, 606, 621, 650, 735
rbenf (Benford), 45
rbenini (Benini), 46
rbetabinom (Betabinom), 49
rbetageom, 58, 61, 68, 297
rbetageom (Betageom), 59
rbetanorm, 58, 68
rbetanorm (Betanorm), 63
rbiamhcop, 70
rbiamhcop (Biamhcop), 68
rbiclaytoncop, 73
rbiclaytoncop (Biclaytoncop), 71
rbifgmcp, 78
rbifgmcp (Bifgmcp), 75
rbifrankcop, 80, 81
rbifrankcop (Frank), 267
rbilogis, 70, 87
rbilogis (bilogis), 84
rbinom, 558, 768
rbinom2. or, 91, 92
rbinom2. or (Binom2.or), 88
rbinom2. rho, 96
rbinom2. rho (Binom2. rho), 93
rbinorm (Binorm), 100
rbinormcop, 104
rbinormcop (Binormcop), 105
rbplackett, 108, 109
rbplackett (Biplackett), 106
rbisa (Bisa), 110
rbort, 118, 119
rbort (Bort), 118
rcard, 140
rcard (Card), 138
Rcim, 319, 473, 474, 531, 607
rcim, 11, 12, 44, 189, 239, 473, 474, 530, 531, 599–603, 607, 686, 724, 726
rcim(grc), 316
rcqo, 185, 591, 608
rdagum (Dagum), 195
rditic, 208, 612
rec.exp1, 614
rec.normal, 615
reciprocal, 280, 390, 488, 489, 616
reexp (Expectiles-Exponential), 223
renorm (Expectiles-Normal), 224
reunif (Expectiles-Uniform), 224
rexp, 223, 243, 463
rexpgam (expgeom), 234
rexplom (explom), 239
rexpob (expob), 244
rfrisk (Fisk), 256
rfoldnorm, 265
rfoldnorm (Foldnorm), 262
rfruchet, 271
rfruchet (Frechet), 268
rgamma, 222, 274, 276, 279
rgengamma.stacy, 289
rgengamma.stacy (gengammaUC), 290
rgennorm (gennorm), 293
rgeom, 562, 772
rgev, 301
rgev (gevUC), 302
rgompertz (Gompertz), 307
rgpd, 312
rgpd (gpdUC), 313
rgumbel, 148, 323
rgumbel (gumbelUC), 327
rgumbelII (Gumbel-II), 324
rhobit, 96, 102, 255, 256, 291, 292, 390, 457, 495, 618
rhuber, 337
rhuber (dhuber), 206
rhzeta (Hzeta), 345
Rice, 619
rice, 606, 619, 620, 620, 650
rig, 648
rigiff, 622
rinv.gaussian, 733
rinv.gaussian (Inv.gaussian), 352
rinv.lomax (Inv.lomax), 355
rinv.paral (Inv.paral), 358
rkumar (Kumar), 366
rlaplace, 373
rlaplace (laplaceUC), 373
rlgamma, 383
rlgamma (lgammaUC), 383
rlind (Lindley), 385
rlino (Lino), 392
rlnorm, 428
rlog, 411
rlog (Log), 404
INDEX

rlogis, 85, 413
rloglap (loglapUC), 419
rliomax (Lomax), 429
rlplot.eveg, 301, 623
rlplot.gev (rlplot.eveg), 623
rmakeham (Makeham), 446
rmaxwell (Maxwell), 453
rnaka, 482
rnaka (Nakagami), 480
rnbinom, 488, 490, 491, 493, 494, 563, 565, 773, 774
RNG, 650
rnorm, 71, 101, 105, 225, 262, 567, 680, 683
Round, 24, 29, 31, 34, 36
rownames, 319
rparalogistic (Paralogistic), 505
rpareto (Pareto), 508
rpareto1 (ParetoIV), 511
rparetoII (ParetoIV), 511
rparetoIII (ParetoIV), 511
rparetoIV (ParetoIV), 511
rperks (Perks), 516
rpois, 243, 467, 570, 776, 787
rpois.points (PoissonPoints), 542
rpolono (Polono), 545
rposbern, 548, 551
rposbern (posbernUC), 555
rposbinom, 746
rposbinom (Posbinom), 557
rposgeom, 749
rposgeom (Posgeom), 561
rposnegbin, 565, 753, 755
rposnegbin (Posnegbin), 562
rposnorm (Posnorm), 566
rpospois, 759
rpospois (Pospois), 569
rrar, 625
rrayleigh (Rayleigh), 603
rrice (Rice), 619
rrvglm-class, 630
rrvglm.control, 185, 317, 319, 444, 591, 627–629, 633, 637
rrvglm.optim.control, 634, 635, 636
rsc.t2 (Expectiles-sc.t2), 226
rsimplex (Simplex), 646
rsinmad, 173
rsinmad (Sinmad), 651
rskellam (Skellam), 654
rskewnorm (skewnorm), 657
rslash, 663
rslash (Slash), 660
rtikuv (Tikuv), 676
rtobit, 683
rtobit (Tobit), 679
rtriangle (Triangle), 688
rtruncpareto (Truncpareto), 694
ruge, 541, 637
rweibull, 173
ryules, 745
ryules (Yules), 743
rzabinom (Zabinom), 745
rzageom (Zageom), 748
rzanegbin (Zanegbin), 752
rzpois, 759
rzpois (Zapois), 756
rzbinom, 767, 768
rzbinom (Zibinom), 765
rzbinom (Zibinom), 765
rzigeom, 772
rzigeom (Zigeom), 769
rzinegbin, 783
rzinegbin (Zinegbin), 773
rzipois, 774
rzipois (Zipois), 782
s, 251, 301, 312, 361, 638, 704–706, 709, 710, 730
sc.studentt2, 25, 227, 639, 670
scale, 132, 530, 591, 665, 666
Select, 251, 319, 551, 554, 641, 721, 722
seq2binomial, 99, 152, 643

vgam-class, 717


vglm-class, 717

vglmff-class, 723
vonmises, 140, 726
vsmooth.spline, 11, 24, 399, 639, 706, 711, 728
waitakere, 341, 730
waldff, 353, 354, 732
waldtest, 435
weibullR, 173, 231, 301, 326, 327, 606, 696, 697, 733
weightsvglm, 715, 736
wine, 737
wrapup.smart, 666, 738
yeo.johnson, 739
yip88, 740, 758, 786, 787
Yules, 743
yulesimon, 650, 743, 744
Zabinom, 745
zabinomial, 558, 746, 768
zabinomialff (zabinomial), 746
Zageom, 748
zageometric, 297, 562, 650, 749, 750, 772
zageometricff, 650
zageometricff (zageometric), 750
Zanegbin, 752
zanegbinomial, 563, 565, 650, 753, 753, 776
zanegbinomialff, 650
zanegbinomialff(zanegbinomial), 753
Zapois, 756
zapoission, 570, 650, 742, 757, 757, 786, 787
zapoissionff, 650
zapoissionff(zapoission), 757
zero, 180, 759
Zeta, 761
zeta, 346, 347, 379, 761, 762, 764, 765
zetaff, 346, 347, 761, 763, 764, 781, 782
Zibinom, 765
zibinomial, 99, 558, 746–748, 765, 766, 767
zibinomialff(zibinomial), 767
Zigeom, 769
zigeometric, 297, 562, 650, 749, 751, 770, 771
zigeometricff, 650
zigeometricff(zigeometric), 771
Zinegbin, 773, 776
zinegbinomial, 173, 491, 563, 629, 650, 755, 774, 774, 786
zinegbinomialff(zinegbinomial), 774
zipebcom, 92, 777, 787
Zipf, 780
zipf, 650, 765, 780, 781, 781
Zipois, 742, 766, 770, 774, 782, 786, 787
zipoission, 317, 319, 541, 570–572, 629, 650, 741, 742, 755, 758, 759, 768, 771, 772, 776–779, 783, 784
zipoissionff, 317–319, 650
zipoissionff(zipoission), 784