Package 'CountsEPPM'

October 12, 2022

Type Package

Title Mean and Variance Modeling of Count Data

Version 3.0

Imports Formula, expm, numDeriv, stats, lmtest, grDevices, graphics

Date 2018-10-18

Author David M Smith, Malcolm J Faddy

Maintainer David M. Smith <smithdm1@us.ibm.com>

Depends R (>= 3.5.0)

Description Modeling under- and over-dispersed count data using extended Poisson process models (EPPM) as in the article Faddy and Smith (2011) <doi:10.18637/jss.v069.i06> .

License GPL-2

Suggests R.rsp

VignetteBuilder R.rsp

NeedsCompilation no

Repository CRAN

Date/Publication 2018-10-28 19:10:14 UTC

R topics documented:

CountsEPPM-package 2
ceriodaphnia.group 3
coef.CountsEPPM 4
cooks.distance.CountsEPPM 5
CountsEPPM 6
EPPMprob 9
Faddyprob.general 9
Faddyprob.limiting 10
fitted.CountsEPPM 11
hatvalues.CountsEPPM 12
herons.case 13
herons.group 13

LL.gradient 14
LL.Regression.Counts 16
logLik.CountsEPPM 18
LRTruncation 19
Luningetal.litters 19
Model.Counts 20
Model.Faddy 22
Model.FaddyJMV.general 23
Model.FaddyJMV.limiting 25
plot.CountsEPPM 27
predict.CountsEPPM 28
print.CountsEPPM 29
print.summaryCountsEPPM 30
residuals.CountsEPPM 31
summary.CountsEPPM 32
takeover.bids.case 32
Titanic.survivors.case 34
vcov.CountsEPPM 35
waldtest.CountsEPPM 36
Williams.litters 37

Index **38**

`CountsEPPM-package` *Fitting of EPPM models to count and binary data.*

Description

Fits regression models to under- and over-dispersed count data using extended Poisson process models.

Details

Package:	CountsEPPM
Type:	Package
Version:	2.1
Date:	2016-03-04
License:	GPL-2

Using Generalized Linear Model (GLM) terminology, the functions utilize linear predictors for mean and variance with log link functions to fit the regression models. Smith and Faddy (2016) gives further details about the package as well as examples of its use.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Faddy M, Smith D. (2011). Analysis of count data with covariate dependence in both mean and variance. *Journal of Applied Statistics*, **38**, 2683-2694. doi: 10.1002/bimj.201100214.

Smith D, Faddy M. (2016). Mean and Variance Modeling of Under- and Overdispersed Count Data. *Journal of Statistical Software*, **69**(6), 1-23. doi: 10.18637/jss.v069.i06.

Zeileis A, Croissant Y. (2010). Extended Model Formulas in R: Multiple Parts and Multiple Responses. *Journal of Statistical Software*, **34**(1), 1-13. doi: 10.18637/jss.v034.i01.

Examples

```
data(herons.group)
initial <- c(1.9871533,1.9900881,3.6841305,0.4925816)
names(initial) <- c("Adult mean","Immature mean", "Variance","log(b)")
output.fn  <- CountsEPPM(number.attempts~0+group | 1, herons.group,initial=initial)
print(output.fn)
```

---

ceriodaphnia.group *Ceriodaphnia data*

Description

Ceriodaphnia dubia are water fleas used to test the impact of effluents on water quality. The data are counts of young at varying effluent concentrations.

Usage

```
data(ceriodaphnia.group)
```

Format

The format is: List of 5 $ vdose : num [1:5] 0 1.56 3.12 6.25 12.5 $ vdose2 : num [1:5] 0 2.44 9.77 39.06 156.25 $ fdose : Factor w/ 5 levels "0","1.5625","3.125",..: 1 2 3 4 5 $ twofdose : Factor w/ 2 levels "0","1": 1 1 2 2 1 $ number.young:List of 5 ..$: num [1:32] 0 0 0 0 0 0 0 0 0 0 ... ..$: num [1:36] 0 0 0 0 0 0 0 0 0 0 ... ..$: num [1:45] 0 0 0 0 0 0 0 0 0 0 ... ..$: num [1:37] 0 0 0 0 0 0 0 0 0 0 ... ..$: num [1:17] 0 0 1 0 0 0 0 3 0 0 ...

Details

The data is used in Faddy and Smith (2011) as an example. Faddy and Smith (2011) is the main reference for the methods implemented. The data are grouped into number of fleas for each count value.

Source

Bailer, A., Oris, J. (1997). Estimating inhibition concentrations for different response scales using Generalized Linear Models. *Environmental Toxicology and Chemistry*, **16**, 1554-1559. doi: 10.1002/etc.5620160732.

References

Faddy M, Smith D. (2011). Analysis of count data with covariate dependence in both mean and variance. *Journal of Applied Statistics*, **38**, 2683-2694. doi: 10.1002/bimj.201100214.

Examples

```
data(ceriodaphnia.group)
print(ceriodaphnia.group)
```

`coef.CountsEPPM` *Extraction of model coefficients for CountsEPPM Objects*

Description

Extract the regression model coefficients from models of class "BinaryEPMM".

Usage

```
## S3 method for class 'CountsEPPM'
coef(object, prtpar = c("full", "mean", "scale.factor"), ...)
```

Arguments

`object`	fitted model object of class "CountsEPPM".
`prtpar`	character indicating coefficients of the fitted model to be output: all coefficients ("full"), coefficients of the model for probability of success ("mean"), coefficients of the model for scale-factor ("scale.factor")
`...`	some methods for this generic function require additional arguments.

Details

One of a set of standard extractor functions for fitted model objects of class "CountsEPPM.

Value

Vector of coefficients of fitted regression model.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

See Also

`betareg`

Examples

```
data(herons.group)
output.fn  <- CountsEPPM(number.attempts ~ 0 + group,
 herons.group, model.type = 'mean only', model.name = 'Poisson')
coef(output.fn, prtpar = "full")
coef(output.fn, prtpar = "mean")
coef(output.fn, prtpar = "scale.factor")
```

`cooks.distance.CountsEPPM` *Cook's distance for CountsEPPM Objects*

Description

Calculates Cook's distances for CountsEPPM objects.

Usage

```
## S3 method for class 'CountsEPPM'
cooks.distance(model, ...)
```

Arguments

`model`	fitted model object of class "CountsEPPM".
`...`	some methods for this generic function require additional arguments.

Details

Cook's distances as in GLMs.

Value

A vector of Cook's distances.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Cribari-Neto F, Zeileis A. (2010). Beta Regression in R. *Journal of Statistical Software*, **34**(2), 1-24. doi: 10.18637/jss.v034.i02.

See Also

`betareg`

Examples

```
data("herons.group")
output.fn  <- CountsEPPM(number.attempts ~ 0 + group,
 herons.group, model.type = 'mean only', model.name = 'Poisson')
cooks.distance(output.fn)
```

CountsEPPM *Fitting of EPPM models to count data.*

Description

Fits regression models to under- and over-dispersed count data using extended Poisson process models.

Usage

```
CountsEPPM(formula, data, subset=NULL, na.action=NULL, weights=NULL,
model.type = "mean and scale-factor", model.name = "general",
link="log", initial = NULL, ltvalue = NA, utvalue = NA,
method = "Nelder-Mead", control = NULL, fixed.b = NA)
```

Arguments

formula	Formulae for the mean and variance. The package 'Formula' of Zeileis and Croissant (2010) which allows multiple parts and multiple responses is used. 'formula' should consist of a left hand side (lhs) of single response variable and a right hand side (rhs) of one or two sets of variables for the linear predictors for the mean and (if two sets) the variance. This is as used for the R function 'glm' and also, for example, as for the package 'betareg' (Cribari-Neto and Zeileis, 2010). The function identifies from the argument data whether a data frame (as for use of 'glm') or a list (as required in Version 1.0 of this function) has been input. The list should be exactly the same as for a data frame except that the response variable is a list of vectors of frequency distributions rather than a vector of single counts as for the data frame. As with version 1.0 of this function, the subordinate functions fit models where the response variables are 'mean.obs', 'variance.obs' or 'scalef.obs' according to the model type being fitted. The values for these response variables are not input as part of 'data', they are calculated within the function from a list of grouped count data input. If the 'model.type' is 'mean only' 'formula' consists of a lhs of the response variable and and a rhs of the terms of the linear predictor for the mean model. If the 'model.type' is 'mean and variance' and 'scale.factor.model'='no' there are two set of terms in the rhs of 'formula' i.e., 'mean.obs' and 'variance.obs' together with the two sets of terms for the linear predictors of mean and variance. If 'scale.factor.model'='yes' the second response variable used by the subordinate functions would be 'scalef.obs'.

data	'data' should be either a data frame (as for use of 'glm') or a list (as required in Version 1.0 of this function). The list should be exactly the same as for a data frame except that the response variable is a list of vectors of frequency distributions rather than a vector of single counts as for the data frame. Within the function a working list 'listcounts' and data frames with components such as 'mean.obs', 'variance.obs', 'scalef.obs', 'covariates', 'offset.mean', 'offset.variance' are set up . The component 'covariates' is a data frame of vectors of covariates in the model. The component 'listcounts' is a list of vectors of the grouped counts, or the single counts in grouped form if 'data' is a data frame.
subset	Subsetting commands.
na.action	Action taken for NAs in data.
weights	Vector of list of lists of weights.
model.type	Takes one of two values i.e. 'mean only' or 'mean and variance'. The 'mean only' value fits a linear predictor function to the parameter 'a' in equation (3) of Faddy and Smith (2011). If the model type being fitted is Poisson modeling 'a' is the same as modeling the mean. For the negative binomial the mean is 'b'(exp('a')-1), 'b' also being as in equation (3) of Faddy and Smith (2011). The 'mean and variance' value fits linear predictor functions to both the mean and the variance.
model.name	If model.type is 'mean only' the model being fitted is one of the three 'Poisson', 'negative binomial', 'Faddy distribution'. If model.type is 'mean and scale-factor' the model being fitted is either 'general' i.e. as equations (4) and (6) of Faddy and Smith (2011), or 'limiting' i.e. as equations (9) and (10) of Faddy and Smith (2011).
link	Takes one of one values i.e., 'log'. The default is 'log'.
initial	This is a vector of initial values for the parameters. If this vector is NULL then initial values based on a fitting Poisson models using 'glm' are calculated within the function.
ltvalue	Lower truncation value.
utvalue	Upper truncation value.
method	Optimization method takes one of the two values 'Nelder-Mead' or 'BFGS' these being options for the optim function.
control	'control' is a list of control parameters as used in 'optim' or 'nlm'. If this list is NULL the defaults for 'optim' are set as 'control <- list(fnscale=-1,trace=0,maxit=1000)' and for 'nlm' are set as 'control <- list(fscale=1,print.level=0,stepmax=1,gradtol=1e-8,steptol=1e-10,iterlim=500)'. For 'optim' the control parameters that can be changed by inputting a variable length list are 'fnscale, trace, maxit, abstol, reltol, alpha, beta, gamma'. For 'nlm' the parameters are 'fscale, print.level, stepmax, gradtol,steptol, iterlim'. Details of 'optim' and 'nlm' and their control parameters are available in the online R help manuals.
fixed.b	Set to the value of the parameter b if a fixed.b model is being used.

Details

Smith and Faddy (2016) gives further details as well as examples of use.

Value

`model.type`	The type of model being fitted
`model`	The model being fitted
`covariates.matrix.mean`	The design matrix for the means
`covariates.matrix.variance`	The design matrix for the variances
`offset.mean`	The offset vector for the means
`offset.variance`	The offset vector for the variances
`ltvalue`	The lower truncation value
`utvalue`	The upper truncation value
`estimates`	Estimates of model parameters
`vnmax`	Vector of maximums of grouped count data vectors in list.counts
`loglikelihood`	Loglikelihood

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Cribari-Neto F, Zeileis A. (2010). Beta Regression in R. *Journal of Statistical Software*, **34**(2), 1-24. doi: 10.18637/jss.v034.i02.

Grun B, Kosmidis I, Zeileis A. (2012). Extended Beta Regression in R: Shaken, Stirred, Mixed, and Partitioned. *Journal of Statistical Software*, **48**(11), 1-25. doi: 10.18637/jss.v048.i11.

Faddy M, Smith D. (2011). Analysis of count data with covariate dependence in both mean and variance. *Journal of Applied Statistics*, **38**, 2683-2694. doi: 10.1002/bimj.201100214.

Smith D, Faddy M. (2016). Mean and Variance Modeling of Under- and Overdispersed Count Data. *Journal of Statistical Software*, **69**(6), 1-23. doi: 10.18637/jss.v069.i06.

Zeileis A, Croissant Y. (2010). Extended Model Formulas in R: Multiple Parts and Multiple Responses. *Journal of Statistical Software*, **34**(1), 1-13. doi: 10.18637/jss.v034.i01.

Examples

```
data(herons.group)
initial <- c(0.5623042, 0.4758576, 0.5082486)
names(initial) <- c("Adult mean", "Immature mean", "log(b)")
output.fn <- CountsEPPM(number.attempts ~ 0 + group,
 herons.group, model.type = 'mean only', model = 'negative binomial',
 initial = initial)
print(output.fn)
```

`EPPMprob` *Calculation of vector of probabilities for a extended Poisson process model (EPPM).*

Description

Calculates a vector of probabilities given a vector of rates vlambda using the matrix exponential function from Bates and Maechler (2012).

Usage

```
EPPMprob(vlambda)
```

Arguments

`vlambda`	vlambda is a vector of rates of an extended Poisson process.

Value

The value returned is a vector of probabilities.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Bates D, Maechler M (2016). Matrix: Sparse and Dense Matrix Classes and Methods. R package version 1.2-4, https://CRAN.R-project.org/package=Matrix.

`Faddyprob.general` *Calculation of vector of probabilities for a Faddy distribution.*

Description

Given a vector of parameters and a scalar of the maximum count the function calculates the vector of lambdas for a Faddy distribution and returns a vector of probabilities.

Usage

```
Faddyprob.general(parameter, nmax)
```

Arguments

`parameter`	A vector of the parameters of the Faddy distribution.
`nmax`	The value of the maximum count.

Value

Vector of probabilities

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Faddy M, Smith D. (2011). Analysis of count data with covariate dependence in both mean and variance. *Journal of Applied Statistics*, **38**, 2683-2694. doi: 10.1002/bimj.201100214.

Examples

```
all.counts=c(rep(0,5),352,479,530,291,101,17)
nmax   <- length(all.counts) - 1
parameter <- c(exp(53.047752),exp(3.801599),-13.205655)
names(parameter) <- c("a","b","c")
probability <- Faddyprob.general(parameter,nmax)
print(probability)
```

Faddyprob.limiting *Calculation of vector of probabilities for the limiting form of the Faddy distribution.*

Description

Given a vector of parameters and a scalar of the maximum count the function calculates the vector of lambdas for the limiting form of a Faddy distribution applicable to under-dispersed data and returns a vector of probabilities. This limiting form is described in Faddy and Smith (2011) and it is appropriate for use on count data displaying under dispersion with respect to the Poisson. If the general model of Faddyprob.general is fitted to such under-dispersed data and a large value of b results, possibly with the hessian at the apparent maximum being poorly conditioned, it is possible that the limiting model having one less parameter than the general model will fit better.

Usage

```
Faddyprob.limiting(parameter, nmax)
```

Arguments

`parameter`	A vector of the parameters of the limiting form of a Faddy distribution.
`nmax`	The value of the maximum count.

Value

Vector of probabilities

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Faddy M, Smith D. (2011). Analysis of count data with covariate dependence in both mean and variance. *Journal of Applied Statistics*, **38**, 2683-2694. doi: 10.1002/bimj.201100214.

Examples

```
all.counts=c(rep(0,5),352,479,530,291,101,17)
nmax1  <- length(all.counts)
nmax   <- nmax1 - 1
parameter <- c(1.8388023,0.6009881)
names(parameter) <- c("beta0 log(mean)","beta0 log(variance)")
probability <- Faddyprob.limiting(parameter,nmax)
print(probability)
```

---

fitted.CountsEPPM *Extraction of fitted values from CountsEPPM Objects*

Description

This function is generic. Extract the fitted values from models of class "BinaryEPMM".

Usage

```
## S3 method for class 'CountsEPPM'
fitted(object, ...)
```

Arguments

`object`	fitted model object of class "CountsEPPM".
`...`	currently not used.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

See Also

`fitted`

Examples

```
data("herons.group")
output.fn  <- CountsEPPM(number.attempts ~ 0 + group,
 herons.group, model.type = 'mean only', model.name = 'Poisson')
cooks.distance(output.fn)
fitted(output.fn)
```

`hatvalues.CountsEPPM` *Extraction of hat matrix values from CountsEPPM Objects*

Description

Extract the values of the hat matrix from models of class "CountsEPMM".

Usage

```
## S3 method for class 'CountsEPPM'
hatvalues(model, ...)
```

Arguments

`model`	fitted model object of class "CountsEPPM".
`...`	some methods for this generic function require additional arguments.

Value

The calculated hat values for the fitted model. These are used to calculate Cook's distances.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Cribari-Neto F, Zeileis A. (2010). Beta Regression in R. *Journal of Statistical Software*, **34**(2), 1-24. doi: 10.18637/jss.v034.i02.

See Also

`betareg`

Examples

```
data("herons.group")
output.fn  <- CountsEPPM(number.attempts ~ 0 + group,
 herons.group, model.type = 'mean only', model.name = 'Poisson')
cooks.distance(output.fn)
hatvalues(output.fn)
```

herons.case *Green-backed herons as two groups*

Description

The data are the numbers of attempts at foraging by 20 adult and 20 immature green-backed herons. The data are listed as grouped (adult or immature) count data i.e. number of herons having a particular count value.

Usage

```
data("herons.case")
```

Format

A data frame with 40 observations on the following 2 variables.

`group` a factor with levels `Adult` `Immature`

`number.attempts` a numeric vector

Source

Zhu J, Eickhoff J, Kaiser M (2003). Modelling the Dependence between Number of Trials and Success Probability in Beta-Binomial-Poisson Mixture Distributions. *Biometrics*, **59**, 955-961. doi: 10.1111/j.0006341X.2003.00110.x.

Examples

```
data(herons.case)
print(herons.case)
```

herons.group *Green-backed herons as two groups*

Description

The data are the numbers of attempts at foraging by 20 adult and 20 immature green-backed herons. The data are listed as grouped (adult or immature) count data i.e. number of herons having a particular count value.

Usage

```
data(herons.group)
```

Format

The format is: List of 2 $ group : Factor w/ 2 levels " Adult"," Immature": 1 2 $ number.attempts:List of 2 ..$: num [1:25] 0 5 2 1 1 1 0 2 0 1 ... ..$: num [1:26] 0 2 2 1 5 1 2 2 1 1 ...

Source

Zhu J, Eickhoff J, Kaiser M (2003). Modelling the Dependence between Number of Trials and Success Probability in Beta-Binomial-Poisson Mixture Distributions. *Biometrics*, **59**, 955-961. doi: 10.1111/j.0006341X.2003.00110.x.

Examples

```
data(herons.group)
print(herons.group)
```

`LL.gradient` *Function used to calculate the first derivatives of the log likelihood with respect to the model parameters.*

Description

Function used to calculate the first derivatives of the log likelihood with respect to the model parameters. These are numerical derivatives calculated using the numerical derivative functions of Gilbert and Varadhan (2015).

Usage

```
LL.gradient(parameter, model.type, model.name, link, list.data,
   covariates.matrix.mean, covariates.matrix.scalef,
   offset.mean, offset.scalef, ltvalue, utvalue, fixed.b,
   weights, grad.method)
```

Arguments

`parameter`	A vector of the parameters of the model which is set to initial estimates on function call.
`model.type`	Takes one of two values i.e. 'mean only' or 'mean and scale-factor'. The 'mean only' value fits linear predictor functions to the mean as in Faddy and Smith (2012). The 'mean and scale-factor' value fits linear predictor functions to both the 'mean' and the scale-factor. The default is 'mean and scale-factor'.
`model.name`	If model.type is 'mean only' the model being fitted is one of the four 'binomial', 'generalized binomial', 'beta binomial' or 'correlated binomial'. If model.type is 'mean and scale-factor' the model being fitted is one of the three 'generalized binomial', 'beta binomial' or 'correlated binomial'. Information about these models is given in Faddy and Smith (2012). The default is 'generalized binomial'.

link	Takes one of one values i.e., 'log'. The default is 'log'.
list.data	A list of vectors of the counts as grouped data i.e. number of observations for each possible count value.
covariates.matrix.mean	A matrix of covariates for the mean where rows are the number of values in list.data and columns the covariates. This matrix is extracted from the formulae in function CountsEPPM. However, in the accompanying example it is shown how it can be constructed independently of function CountsEPPM.
covariates.matrix.scalef	A matrix of covariates for the variance where rows are the number of values in list.binary and columns the covariates. The default is a vector of ones. This matrix is extracted from the formulae in function CountsEPPM. However, in the accompanying example it is shown how it can be constructed independently of function CountsEPPM.
offset.mean	An offset vector for the probability of success p. The default is a vector of ones.
offset.scalef	An offset vector for the scale-factor. The default is a vector of ones.
ltvalue	Lower truncation value.
utvalue	Upper truncation value.
fixed.b	Set to the value of the parameter b if a fixed.b model is being used.
weights	A vector or list of weights for the modeling of probability of success. The default is a vector of ones.
grad.method	Set to the method to be used to calculate the gradients either "simple" or "Richardson".

Value

A vector of numerical first derivatives.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Gilbert P, Varadhan R. (2015). numDeriv: Accurate Numerical Derivatives. R Package version 2014.2-1, https://CRAN.R-project.org/package=numDeriv.

Examples

```
## Not run:
gradient <- grad( LL.Regression.Binary, x = parameter,
 model.type = model.type, model.name = model.name, link = link,
 ntrials = ntrials, nsuccess = nsuccess,
 covariates.matrix.mean = covariates.matrix.mean,
 covariates.matrix.scalef = covariates.matrix.scalef,
 offset.mean = offset.mean, offset.scalef = offset.scalef,
 weights = weights, grad.method = "Richardson")
```

```
 return(gradient)

## End(Not run)
```

`LL.Regression.Counts` *Function called by optim to calculate the log likelihood from the probabilities and hence perform the fitting of regression models to the binary data.*

Description

Fits specified regression models to the data.

Usage

```
LL.Regression.Counts(parameter, model.type, model.name,
 link, list.data, covariates.matrix.mean,
 covariates.matrix.scalef, offset.mean, offset.scalef,
 ltvalue, utvalue, fixed.b, weights, grad.method)
```

Arguments

`parameter`	A vector of the parameters of the model which is set to initial estimates on function call.
`model.type`	Takes one of two values i.e. 'mean only' or 'mean and scale-factor'. The 'mean only' value fits a linear predictor function to the parameter 'a' in equation (3) of of Faddy and Smith (2011). If the model type being fitted is Poisson modeling 'a' is the same as modeling the mean. The 'mean and scale-factor' value fits linear predictor functions to both the mean and the scale-factor.
`model.name`	If model.type is 'mean only' the model being fitted is one of the three 'Poisson', 'negative binomial', 'Faddy distribution'. If model.type is 'mean and scale-factor' the model being fitted is either 'general' i.e. as equations (4) and (6) of Faddy and Smith (2011), or 'limiting' i.e. as equations (9) and (10) of of Faddy and Smith (2011).
`link`	Takes one of one values i.e., 'log'. The default is 'log'.
`list.data`	A list of vectors of the counts as grouped data i.e. number of observations for each possible count value.
`covariates.matrix.mean`	A matrix of covariates for the mean where rows are the number of values in list.counts and columns the covariates. This matrix is extracted from the formulae in function CountsEPPM. However, in the accompanying example it is shown how it can be constructed independently of function CountsEPPM.
`covariates.matrix.scalef`	A matrix of covariates for the scale-factor where rows are the number of values in list.counts and columns the covariates. The default is a vector of ones. This matrix is extracted from the formulae in function CountsEPPM. However, in the

	accompanying example it is shown how it can be constructed independently of function CountsEPPM.
offset.mean	An offset vector for the mean. The default is a vector of ones.
offset.scalef	An offset vector for the scale-factor. The default is a vector of ones.
ltvalue	Lower truncation value.
utvalue	Upper truncation value.
fixed.b	Set to the value of the parameter b if a fixed.b model is being used.
weights	A vector or list of weights for the modeling of probability of success. The default is a vector of ones.
grad.method	Set to the method to be used to calculate the gradients either "simple" or "Richardson".

Value

The log likelihood with an attribute of the gradients produced by the function grad from the package numDerive is returned.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Faddy M, Smith D. (2011). Analysis of count data with covariate dependence in both mean and scale-factor. *Journal of Applied Statistics*, **38**, 2683-2694. doi: 10.1002/bimj.201100214.

Examples

```
all.counts=c(rep(0,5),352,479,530,291,101,17)
nmax1  <- length(all.counts)
nmax   <- nmax1 - 1
cnum   <- 0:nmax
ncount <- sum(all.counts)
all.mean <- sum(cnum*all.counts)/ncount
all.scalef <- ((sum(cnum*cnum*all.counts) - ncount*all.mean*all.mean)
 / (ncount - 1)) / all.mean
alldata <- data.frame(all.mean, all.scalef)
mf  <- model.frame(formula = all.mean~1, data = alldata)
covariates.matrix.mean <- model.matrix(attr(mf, "terms"), data = mf)
mf  <- model.frame(formula=all.scalef~1, data=alldata)
covariates.matrix.scalef <- model.matrix(attr(mf, "terms"), data=mf)
list.data <- list(all.counts)
parameter <- c(1.8388023, 0.6009881)
names(parameter) <- c("beta0 log(mean)", "beta0 log(scale-factor)")
offset.mean <- matrix(c(rep(0,nrow(covariates.matrix.mean))), ncol=1)
offset.scalef <- matrix(c(rep(0,nrow(covariates.matrix.scalef))), ncol=1)
link <- "log"
attr(link, which="mean") <- make.link(link)
output <- LL.Regression.Counts(parameter,
```

```
 model.type = "mean and scale-factor", model.name = "limiting",
 link, list.data, covariates.matrix.mean,
 covariates.matrix.scalef, offset.mean, offset.scalef, ltvalue=4,
 utvalue=11, fixed.b=NA, weights = NULL, grad.method = "simple")
print(output)
```

`logLik.CountsEPPM` *Method for CountsEPPM object*

Description

This function is generic and enables the use of functions related to the model fitting involved with `lm` and `glm` objects such as `AIC`.

Usage

```
## S3 method for class 'CountsEPPM'
logLik(object, ...)
```

Arguments

`object`	The object output from `CountsEPPM`.
`...`	currently not used.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

Examples

```
## Not run:
output.fn <- CountsEPPM(mean.obs ~1, Luningetal.all,
 model.type,model, initial, ltvalue = 4, utvalue = 11,
 optimization.method = "nlm")
logLik.CountsEPPM(object=output.fn)

## End(Not run)
```

LRTruncation — *Probabilities for distributions truncated on the left (lower) and/or right (upper).*

Description

Given left (lower) and/or right (upper) truncation values and probabilities for a distribution calculates and returns the probabilities for the truncated distribution.

Usage

```
LRTruncation(probability, ltvalue, utvalue)
```

Arguments

probability	Probabilities for untruncated distribution.
ltvalue	Left (lower) truncation value.
utvalue	Right (upper) truncation value.

Value

Vector of probabilities for truncated distribution.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

Examples

```
probability <- c(3.375659e-08, 1.023277e-05, 5.440752e-04,
 8.768246e-03, 5.663573e-02, 1.735599e-01, 2.819850e-01,
 2.625282e-01, 1.482712e-01, 5.305443e-02, 1.244452e-02)
probabilities <- LRTruncation(probability, ltvalue=4, utvalue=11)
print(probabilities)
```

Luningetal.litters — *Number of trials (implantations) in data of Luning, et al. (1966)*

Description

The data are arranged as a list of grouped counts where the grouping is by dose where dose is included both as a variate (vdose) and as a factor (fdose).

Usage

```
data(Luningetal.litters)
```

Format

The format is: List of 3 $ vdose : num [1:3] 0 300 600 $ fdose : Factor w/ 3 levels "0","300","600": 1 2 3 $ number.trials:List of 3 ..$: num [1:11] 0 0 0 0 0 71 156 224 150 70 ... ..$: num [1:11] 0 0 0 0 0 121 170 186 99 24 ... ..$: num [1:11] 0 0 0 0 0 160 153 120 45 7 ...

Source

Luning K, Sheridan W, Ytterborn K, Gullberg U (1966). The relationship between the number of implantations and the rate of intra-uterine death in mice. *Mutation Research*, **3**, 444-451. doi: 10.1016/00275107(66)900546.

Examples

```
data(Luningetal.litters)
print(Luningetal.litters)
```

`Model.Counts` *Function for obtaining output from distributional models.*

Description

Produces output of model, parameters and probabilities from the various models.

Usage

```
Model.Counts(parameter, model.type, model.name, link,
 covariates.matrix.mean, covariates.matrix.scalef,
 offset.mean, offset.scalef, fixed.b, vnmax)
```

Arguments

`parameter`	A vector of the parameters of the model which is set to initial estimates on function call.
`model.type`	Takes one of two values i.e. 'mean only' or 'mean and scale-factor'. The 'mean only' value fits a linear predictor function to the parameter 'a' in equation (3) of Faddy and Smith (2011). If the model type being fitted is Poisson modeling 'a' is the same as modeling the mean. The 'mean and scale-factor' value fits linear predictor functions to both the mean and the scale-factor.
`model.name`	If model.type is 'mean only' the model being fitted is one of the three 'Poisson', 'negative binomial', 'Faddy distribution'. If model.type is 'mean and scale-factor' the model being fitted is either 'general' i.e. as equations (4) and (6) of Faddy and Smith (2011), or 'limiting' i.e. as equations (9) and (10) of Faddy and Smith (2011).
`link`	Takes one of one values i.e., 'log'. The default is 'log'.

`covariates.matrix.mean`
: A matrix of covariates for the mean where rows are the number of values in list-counts and columns the covariates. This matrix is extracted from the formulae in function CountsEPPM. However, in the accompanying example it is shown how it can be constructed independently of function CountsEPPM.

`covariates.matrix.scalef`
: A matrix of covariates for the scale-factor where rows are the number of values in listcounts and columns the covariates. The default is a vector of ones. This matrix is extracted from the formulae in function CountsEPPM. However, in the accompanying example it is shown how it can be constructed independently of function CountsEPPM.

`offset.mean`
: An offset vector for the mean. The default is a vector of ones.

`offset.scalef`
: An offset vector for the scale-factor. The default is a vector of ones.

`fixed.b`
: Set to the value of the parameter b if a fixed.b model is being used.

`vnmax`
: A vector of the maximum counts for each vector in list.counts i.e. the list of grouped counts.

Value

Output which is the output from either Model.Faddy, Model.Faddy.general, or Model.Faddy.limiting

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Faddy M, Smith D. (2011). Analysis of count data with covariate dependence in both mean and variance. *Journal of Applied Statistics*, **38**, 2683-2694. doi: 10.1002/bimj.201100214.

Examples

```
all.counts=c(rep(0,5),352,479,530,291,101,17)
nmax1  <- length(all.counts)
nmax   <- nmax1 - 1
cnum   <- 0:nmax
ncount <- sum(all.counts)
all.mean <- sum(cnum*all.counts)/ncount
all.scalef <- ((sum(cnum*cnum*all.counts) - ncount*all.mean*all.mean) / (ncount - 1)) / all.mean
alldata <- data.frame(all.mean, all.scalef)
mf  <- model.frame(formula = all.mean~1 ,data=alldata)
covariates.matrix.mean <- model.matrix(attr(mf,"terms"), data=mf)
mf  <- model.frame(formula = all.scalef~1, data = alldata)
covariates.matrix.scalef <- model.matrix(attr(mf,"terms"), data = mf)
list.counts <- list(all.counts)
parameter <- c(1.8388023, 0.6009881)
names(parameter) <- c("beta0 log(mean)" ,"beta0 log(scale-factor)")
offset.mean     <- matrix(c(rep(0, nrow(covariates.matrix.mean))), ncol=1)
offset.scalef <- matrix(c(rep(0, nrow(covariates.matrix.mean))), ncol=1)
link <- "log"
```

```
attr(link, which="mean") <- make.link(link)
output <- Model.Counts(parameter, model.type = "mean and scale-factor",
 model.name = "limiting", link, covariates.matrix.mean,
 covariates.matrix.scalef, offset.mean, offset.scalef,
 fixed.b = NA, vnmax = c(10))
print(output)
```

`Model.Faddy` *Function for Faddy distribution with log link.*

Description

Returns probabilities for a Faddy distribution given inputs of model and parameters.

Usage

```
Model.Faddy(parameter, model.name, link, covariates.matrix.mean,
 offset.mean, fixed.b, vnmax)
```

Arguments

`parameter`	A vector of the parameters of the model which is set to initial estimates on function call.
`model.name`	The model being fitted is one of the five 'Poisson', 'negative binomial', 'negative binomial fixed b', 'Faddy distribution', 'Faddy distribution fixed b'.
`link`	Takes one of one values i.e., 'log'. The default is 'log'.
`covariates.matrix.mean`	A matrix of covariates for the mean where rows are the number of values in list-counts and columns the covariates. This matrix is extracted from the formulae in function CountsEPPM. However, in the accompanying example it is shown how it can be constructed independently of function CountsEPPM.
`offset.mean`	An offset vector for the mean. The default is a vector of ones. This matrix is extracted from the formulae in function CountsEPPM.
`fixed.b`	Set to the value of the parameter b if a fixed.b model is being used.
`vnmax`	A vector of the maximum counts for each vector in list.counts i.e. the list of grouped counts.

Value

The list output with elements

`model`	The model being fitted
`estimate`	Estimates of parameters
`probabilities`	Vector of probabilities

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Faddy M, Smith D. (2011). Analysis of count data with covariate dependence in both mean and variance. *Journal of Applied Statistics*, **38**, 2683-2694. doi: 10.1002/bimj.201100214.

Examples

```
all.counts=c(rep(0,5), 352, 479, 530, 291, 101, 17)
nmax1  <- length(all.counts)
nmax   <- nmax1 - 1
cnum   <- 0:nmax
ncount <- sum(all.counts)
all.mean <- t(cnum)
alldata <- data.frame(all.mean)
mf <- model.frame(formula = all.mean ~ 1, data = alldata)
covariates.matrix.mean <- model.matrix(attr(mf, "terms"), data = mf)
list.counts <- list(all.counts)
parameter <- c(53.047752, -13.205655, 3.801599)
names(parameter) <- c('log(a)', 'c', 'log(b)')
model.name <- 'Faddy distribution'
link <- "log"
attr(link, which="mean") <- make.link(link)
offset.mean <- matrix(c(rep(0, nrow(covariates.matrix.mean))), ncol=1)
output <- Model.Faddy(parameter, model.name, link,
 covariates.matrix.mean, offset.mean, fixed.b = NA,
 vnmax = c(10))
print(output)
```

`Model.FaddyJMV.general` *Function for a general Faddy distribution modeled by means and scale-factors.*

Description

Outputs probabilities for a general Faddy distribution modeled by means and scale-factors i.e. with the design matrices for mean and scale-factor input together with data and offsets.

Usage

```
Model.FaddyJMV.general(parameter, link, covariates.matrix.mean,
 covariates.matrix.scalef, offset.mean, offset.scalef,
 fixed.b, vnmax)
```

Arguments

parameter	A vector of the parameters of the model which is set to initial estimates on function call.
link	Takes one of one values i.e., 'log'. The default is 'log'.
covariates.matrix.mean	A matrix of covariates for the mean where rows are the number of values in list-counts and columns the covariates. This matrix is extracted from the formulae in function CountsEPPM. However, in the accompanying example it is shown how it can be constructed independently of function CountsEPPM.
covariates.matrix.scalef	A matrix of covariates for the scale factor where rows are the number of values in listcounts and columns the covariates. The default is a vector of ones. This matrix is extracted from the formulae in function CountsEPPM. However, in the accompanying example it is shown how it can be constructed independently of function CountsEPPM.
offset.mean	An offset vector for the mean. The default is a vector of ones.
offset.scalef	An offset vector for the scale-factor. The default is a vector of ones.
fixed.b	Set to the value of the parameter b if a fixed.b model is being used.
vnmax	A vector of the maximum counts for each vector in list.counts i.e. the list of grouped counts.

Value

The list output with elements

model	The model being fitted
estimate	Estimates of parameters
probabilities	Vector of probabilities

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Faddy M, Smith D. (2011). Analysis of count data with covariate dependence in both mean and variance. *Journal of Applied Statistics*, **38**, 2683-2694. doi: 10.1002/bimj.201100214.

Examples

```
all.counts=c(rep(0,5),352,479,530,291,101,17)
nmax1  <- length(all.counts)
nmax   <- nmax1 - 1
cnum   <- 0:nmax
ncount <- sum(all.counts)
all.mean <- sum(cnum*all.counts)/ncount
all.scalef <- ((sum(cnum*cnum*all.counts) - ncount*all.mean*all.mean) / (ncount - 1)) / all.mean
```

```
alldata <- data.frame(all.mean, all.scalef)
mf <- model.frame(formula = all.mean~1, data=alldata)
covariates.matrix.mean <- model.matrix(attr(mf, "terms"), data=mf)
mf <- model.frame(formula = all.scalef~1, data = alldata)
covariates.matrix.scalef <- model.matrix(attr(mf, "terms"), data = mf)
list.counts <- list(all.counts)
parameter <- c(1.8386079, 0.6021198, 6.0714071)
names(parameter) <- c("beta0 log(mean)", "beta0 log(scale-factor)", "log(b)")
link <- "log"
attr(link, which = "mean") <- make.link(link)
offset.mean <- matrix(c(rep(0,nrow(covariates.matrix.mean))), ncol = 1)
offset.scalef <- matrix(c(rep(0,nrow(covariates.matrix.mean))), ncol = 1)
output <- Model.FaddyJMV.general(parameter, link,
 covariates.matrix.mean, covariates.matrix.scalef,
 offset.mean, offset.scalef, fixed.b = NA, vnmax = c(10))
print(output)
```

`Model.FaddyJMV.limiting` *Function to fit the limiting form of a Faddy distribution for under-dispersed counts.*

Description

Outputs probabilities for the limiting form of a Faddy distribution modeled by means and scale-factors i.e. with the design matrices for mean and scale-factor input together with data and offsets.

Usage

```
Model.FaddyJMV.limiting(parameter, link, covariates.matrix.mean,
covariates.matrix.scalef, offset.mean, offset.scalef, vnmax)
```

Arguments

`parameter`	A vector of the parameters of the model which is set to initial estimates on function call.
`link`	Takes one of one values i.e., 'log'. The default is 'log'.
`covariates.matrix.mean`	A matrix of covariates for the mean where rows are the number of values in list-counts and columns the covariates. This matrix is extracted from the formulae in function CountsEPPM. However, in the accompanying example it is shown how it can be constructed independently of function CountsEPPM.
`covariates.matrix.scalef`	A matrix of covariates for the scale factor where rows are the number of values in listcounts and columns the covariates. The default is a vector of ones. This matrix is extracted from the formulae in function CountsEPPM. However, in the accompanying example it is shown how it can be constructed independently of function CountsEPPM.

offset.mean	An offset vector for the mean. The default is a vector of ones.
offset.scalef	An offset vector for the scale-factor. The default is a vector of ones.
vnmax	A vector of the maximum counts for each vector in list.counts i.e. the list of grouped counts.

Value

The list output with elements

model	The model being fitted
estimate	Estimates of parameters
probabilities	Vector of probabilities

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Faddy M, Smith D. (2011). Analysis of count data with covariate dependence in both mean and variance. *Journal of Applied Statistics*, **38**, 2683-2694. doi: 10.1002/bimj.201100214.

Examples

```
all.counts=c(rep(0,5),352,479,530,291,101,17)
nmax1  <- length(all.counts)
nmax   <- nmax1 - 1
cnum   <- 0:nmax
ncount <- sum(all.counts)
all.mean <- sum(cnum*all.counts)/ncount
all.scalef <- ((sum(cnum*cnum*all.counts) - ncount*all.mean*all.mean) / (ncount - 1)) / all.mean
alldata <- data.frame(all.mean, all.scalef)
mf  <- model.frame(formula = all.mean~1, data = alldata)
covariates.matrix.mean <- model.matrix(attr(mf, "terms"), data = mf)
mf  <- model.frame(formula = all.scalef~1, data = alldata)
covariates.matrix.scalef <- model.matrix(attr(mf, "terms"), data = mf)
list.counts <- list(all.counts)
parameter <- c(1.8388023, 0.6009881)
names(parameter) <- c("beta0 log(mean)", "beta0 log(scale-factor)")
link <- "log"
attr(link, which = "mean") <- make.link(link)
offset.mean <- matrix(c(rep(0, nrow(covariates.matrix.mean))), ncol=1)
offset.scalef <- matrix(c(rep(0, nrow(covariates.matrix.mean))), ncol=1)
output <- Model.FaddyJMV.limiting(parameter, link,
 covariates.matrix.mean, covariates.matrix.scalef,
 offset.mean, offset.scalef, vnmax = c(10))
print(output)
```

plot.CountsEPPM *Diagnostic Plots for CountsEPPM Objects*

Description

Various types of standard diagnostic plots can be produced, involving various types of residuals, influence measures etc.

Usage

```
## S3 method for class 'CountsEPPM'
plot(x, which = 1:4, caption = c("Residuals vs indices of obs.", "Cook's distance plot",
 "Leverage vs predicted values", "Residuals vs linear predictor",
 "Normal Q-Q plot of residuals", "Predicted vs observed values"),
 sub.caption = " ", main = "", ask = prod(par("mfcol"), 1) <
 length(which) && dev.interactive(), ..., type = "spearson")
```

Arguments

x	fitted model object of class "CountsEPPM".
which	numeric. If a subset of plots is required, specify a subset of the numbers 1:6.
caption	character. Captions to appear above the plots.
sub.caption	character. Common title-above figures if there are multiple.
main	chaacter. Title to each plot in addition to the above caption.
ask	logical. If true, the user is asked before each plot.
...	other parameters to be passed through to plotting functions.
type	character indicating type of residual to be used, see residuals.CountsEPPM.

Details

The plot method for CountsEPPM objects produces various plots of diagnostic plots similar to those produced by **betareg**. See Ferrari and Cribari-Neto (2004) for further details of the displays of **betareg**.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Cribari-Neto F, Zeileis A. (2010). Beta Regression in R. *Journal of Statistical Software*, **34**(2), 1-24. doi: 10.18637/jss.v034.i02.

See Also

plot.betareg

Examples

```
## Not run:
data("herons.group")
output.fn  <- CountsEPPM(number.attempts ~ 0 + group,
 herons.group, model.type = 'mean only', model.name = 'Poisson')
cooks.distance(output.fn)
plot(output.fn, which = 1, type= "sdeviance")

## End(Not run)
```

`predict.CountsEPPM` *Prediction Method for CountsEPPM Objects*

Description

Extract various types of predictions from CountsEPPM regression models.

Usage

```
## S3 method for class 'CountsEPPM'
predict(object, newdata = NULL,
 type = c("response", "linear.predictor.mean",
 "linear.predictor.scale.factor", "scale.factor",
 "mean", "variance", "distribution", "distribution.parameters"),
 na.action = na.pass, ...)
```

Arguments

`object`	fitted model object of class "CountsEPPM".
`newdata`	optionally, a data frame in which to look for variables with which to predict. If omitted, the original observations are used.
`type`	character indicating type of predictions: fitted means of responses ("response"), linear predictors ("linear.predictor.mean", "linear.predictor.scale.factor"), fitted value of mean ("mean"), fitted value of scale-factor ("scale.factor"), fitted value of variance ("variance"), fitted probability distribution ("distribution"), parameters of fitted distributions ("distribution.parameters")
`na.action`	function determining what to do with missing values in *newdata*. The default is to predict NA.
`...`	some methods for this generic function require additional arguments.

Value

A vector or list of the predicted values from the fitted model object.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Cribari-Neto F, Zeileis A. (2010). Beta Regression in R. *Journal of Statistical Software*, **34**(2), 1-24. doi: 10.18637/jss.v034.i02.

See Also

predict.betareg

Examples

```
data("herons.group")
output.fn  <- CountsEPPM(number.attempts ~ 0 + group,
 herons.group, model.type = 'mean only', model.name = 'Poisson')
predict(output.fn, type = "response")
predict(output.fn, type = "linear.predictor.mean")
```

---

print.CountsEPPM *Printing of CountsEPPM Objects*

Description

Prints objects of class "CountsEPPM".

Usage

```
## S3 method for class 'CountsEPPM'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	fitted model object of class "CountsEPPM".
digits	digits of printed output.
...	not currently used.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Cribari-Neto F, Zeileis A. (2010). Beta Regression in R. *Journal of Statistical Software*, **34**(2), 1-24. doi: 10.18637/jss.v034.i02.

See Also

betareg

Examples

```
data("herons.group")
output.fn  <- CountsEPPM(number.attempts ~ 0 + group,
 herons.group, model.type = 'mean only', model.name = 'Poisson')
print(output.fn)
```

`print.summaryCountsEPPM` *Printing of summaryCountsEPPM Objects*

Description

Prints the objects of class "summaryCountsEPPM".

Usage

```
## S3 method for class 'summaryCountsEPPM'
print(x, ...)
```

Arguments

`x`	object output by `summary.CountsEPPM`.
`...`	not currently used.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Cribari-Neto F, Zeileis A. (2010). Beta Regression in R. *Journal of Statistical Software*, **34**(2), 1-24. doi: 10.18637/jss.v034.i02.

See Also

`betareg`

Examples

```
data("herons.group")
output.fn  <- CountsEPPM(number.attempts ~ 0 + group,
 herons.group, model.type = 'mean only', model.name = 'Poisson')
print(summary(output.fn))
```

`residuals.CountsEPPM` *Residuals for CountsEPPM Objects*

Description

This function is generic. Extract various types of residuals from objects of class "CountsEPPM".

Usage

```
## S3 method for class 'CountsEPPM'
residuals(object, type = c("spearson",
 "deviance", "pearson", "response", "likelihood", "sdeviance"),
 ...)
```

Arguments

`object`	Fitted model object of class "CountsEPPM".
`type`	Type of residuals wanted i.e., standardized Pearson "spearson", deviance "deviance", Pearson "pearson", response "response", likelihood "likelihood", standardized deviance "sdeviance".
`...`	some methods for this geneic function require additional arguments.

Details

Residuals as Cribari-Neto and Zeileis (2010).

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Cribari-Neto F, Zeileis A. (2010). Beta Regression in R. *Journal of Statistical Software*, **34**(2), 1-24. doi: 10.18637/jss.v034.i02.

See Also

`residuals.betareg`

summary.CountsEPPM *Method for CountsEPPM object*

Description

This function is generic and is for printing out a summary of the results of fitting EPPM models to count data.

Usage

```
## S3 method for class 'CountsEPPM'
summary(object, ...)
```

Arguments

object	The object output from CountsEPPM. This list includes a vector vnmax of the maximums of the grouped count vectors in list.counts. The vector vnmax can be changed before calling this function in order to give more complete probability vectors i.e. closer to a total of 1.
...	currently not used.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

Examples

```
## Not run:
output.fn <- CountsEPPM(mean.obs ~ 1, Luningetal.all, model.type,
 model,initial, ltvalue = 4, utvalue = 11, optimization.method =  "nlm")
summary(object=output.fn)

## End(Not run)
```

takeover.bids.case *Takeover bids data.*

Description

Data of the number of bids received by 126 U.S. firms that were targets of tender offers from 1978 to 1985.

Usage

```
data("takeover.bids.case")
```

Format

A data frame with 126 observations on the following 12 variables.

`DOCNO` a numeric vector

`WEEKS` a numeric vector

`NUMBIDS` a numeric vector

`BIDPREM` a numeric vector

`INSTHOLD` a numeric vector

`SIZE` a numeric vector

`LEGLREST` a numeric vector

`REALREST` a numeric vector

`FINREST` a numeric vector

`REGULATN` a numeric vector

`WHTKNGHT` a numeric vector

`SIZESQ` a numeric vector

Details

Data originally from Jaggia and Thosar (1993) and used as an example in Cameron and Trivedi (2013) and Saez-Castillo and Conde-Sanchez (2013).

Source

Stata data file obtained from A.C. Cameron's webpage http://cameron.econ.ucdavis.edu/.

References

Cameron, A.C., Trivedi, P.K. (2013). Regression Analysis of Count Data. Cambridge University Press, second edition.

Jaggia, S., Thosar, S. (1993). Multiple Bids as a Consequence of Target Management Resistance. *Review of Quantitative Finance and Accounting*, 447-457.

Saez-Castillo, A.J., Conde-Sanchez, A. (2013). A hyper-Poisson regression model for overdispersed and underdispersed count data. *Computational Statistics and Data Analysis*, **61**, 148-157.doi: 10.1016/j.csda.2012.12.009

Examples

```
data(takeover.bids.case)
print(takeover.bids.case)
```

`Titanic.survivors.case` *Titanic survivors data*

Description

These data are from the survival log of the Titanic and consist of the number of survivors out of the number of passengers broken down into age, sex and class categories.

Usage

```
data(Titanic.survivors.case)
```

Format

A data frame with 12 observations on the following 5 variables.

`age` a factor with levels `child` `adult`

`sex` a factor with levels `female` `male`

`class` a factor with levels `1st class` `2nd class` `3rd class`

`cases` a numeric vector

`survive` a numeric vector

Details

Hilbe (2011) first models these data as a logistic model, then finding that they are overdispersed, modeling them as count data (number of survivors, survive) with offset (log of the number of passengers, cases).

Source

Section 9.5, Example 3, pages 263-268, Hilbe, J. (2011).

References

Hilbe, J. (2011). Negative Binomial Regression. Cambridge University Press, second edition.

Examples

```
data(Titanic.survivors.case)
print(Titanic.survivors.case)
```

vcov.CountsEPPM *Variance/Covariance Matrix for Coefficients*

Description

Variance/covariance matrix for coefficients of fitted model.

Usage

```
## S3 method for class 'CountsEPPM'
vcov(object, model = c("full", "mean", "scale.factor"), ...)
```

Arguments

object	fitted model object of class "CountsEPPM".
model	character indicating variance/covariance matrix for all coefficients to be output: all coefficients ("full"), variance/covariance matrix for coefficients of probability of success ("mean"), variance/covariance matrix for coefficients of scale-factor ("scale.factor")
...	

Value

The variance/covariance matrix of the parameters of the fitted model object.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Cribari-Neto F, Zeileis A. (2010). Beta Regression in R. *Journal of Statistical Software*, **34**(2), 1-24. doi: 10.18637/jss.v034.i02.

See Also

betareg

Examples

```
data("herons.group")
output.fn  <- CountsEPPM(number.attempts ~ 0 + group,
 herons.group, model.type = 'mean only', model.name = 'Poisson')
vcov(output.fn)
```

`waldtest.CountsEPPM` *Wald Test of Nested Models for CountsEPPM Objects*

Description

`waldtest` is a generic function for comparisons of nested (generalized) linear models via Wald tests.

Usage

```
## S3 method for class 'CountsEPPM'
waldtest(object, ..., vcov = NULL,
   test = c("Chisq", "F"))
```

Arguments

`object`	an object of class "CountsEPPM".
`...`	further object specifications passed to methods. See below for details.
`vcov`	a function for estimating the covariance matrix of the regression coefficients. If only two models are compared it can also be the covariance matrix of the more general model.
`test`	character specifying whether to compute the large sample Chi-squared statistic (with asymptotic Chi-squared distribution) or the finite sample F statistic (with approximate F distribution).

Details

`waldtest` is a generic function for comparisons of nested (generalized)linear models via Wald tests. It does not have the same functionality as the versions of **betareg** and **lmtest** with a reduced list of arguments. With these caveats, more details can be obtained from the **Details** pages of those packages.

Value

An object of class `"anova"` which contains the residual degrees of freedom, the difference in degrees of freedom, Wald statistic (either `"Chisq"` or `"F"`) and corresponding p value.

Author(s)

David M. Smith <smithdm1@us.ibm.com>

References

Cribari-Neto F, Zeileis A. (2010). Beta Regression in R. *Journal of Statistical Software*, **34**(2), 1-24. doi: 10.18637/jss.v034.i02.

Zeileis A, Hothorn T. (2002). Diagnostic Checking in Regression Relationships. *R News*, **2**(3), 7-10. https://CRAN.R-project.org/doc/Rnews/.

See Also

waldtest betareg

Examples

```
data("herons.group")
## Not run:
output.fn  <- CountsEPPM(number.attempts ~ 0 + group,
 herons.group, model.type = 'mean only', model.name = 'Poisson')
output.fn.one  <- CountsEPPM(number.attempts ~ 0 + group,
 herons.group, model.type = 'mean only',
 model.name = 'negative binomial')
waldtest(output.fn, output.fn.one, test = c("Chisq", "F"), vcov =  vcov)

## End(Not run)
```

Williams.litters *Number of trials (implantations) of data of Williams (1996).*

Description

The data is arranged as a list of grouped counts where the grouping is by dose where dose is included both as a variate (vdose) and as a factor (fdose).

Usage

```
data(Williams.litters)
```

Format

The format is: List of 3 $ vdose : num [1:4] 0 0.75 1.5 3 $ fdose : Factor w/ 4 levels "0","0.75","1.5",..: 1 2 3 4 $ number.implants:List of 4 ..$: num [1:18] 0 0 0 0 0 0 0 0 0 2 ... ..$: num [1:20] 0 0 0 0 0 0 0 1 0 1 ... ..$: num [1:17] 0 0 0 0 0 0 0 1 1 2 ... ..$: num [1:17] 0 0 0 0 0 0 0 0 0 3 ...

Source

Williams D (1996). Overdispersion in logistic linear model. In B Morgan (ed.), Statistics in Toxicology, pp. 75-84, Oxford Science Publications.

Examples

```
data(Williams.litters)
print(Williams.litters)
```

Index

∗ **IO**
print.CountsEPPM, 29
print.summaryCountsEPPM, 30
∗ **Methods**
logLik.CountsEPPM, 18
summary.CountsEPPM, 32
∗ **datasets**
ceriodaphnia.group, 3
herons.case, 13
herons.group, 13
Luningetal.litters, 19
takeover.bids.case, 32
Titanic.survivors.case, 34
Williams.litters, 37
∗ **distribution**
EPPMprob, 9
Faddyprob.general, 9
Faddyprob.limiting, 10
predict.CountsEPPM, 28
∗ **hplot**
plot.CountsEPPM, 27
∗ **methods**
coef.CountsEPPM, 4
cooks.distance.CountsEPPM, 5
fitted.CountsEPPM, 11
hatvalues.CountsEPPM, 12
predict.CountsEPPM, 28
waldtest.CountsEPPM, 36
∗ **misc**
LRTruncation, 19
∗ **models**
CountsEPPM, 6
Model.Counts, 20
Model.Faddy, 22
Model.FaddyJMV.general, 23
Model.FaddyJMV.limiting, 25
residuals.CountsEPPM, 31
vcov.CountsEPPM, 35
∗ **model**
LL.gradient, 14
LL.Regression.Counts, 16
∗ **package**
CountsEPPM-package, 2

betareg, *4, 5*, *12*, *29, 30*, *35*, *37*

ceriodaphnia.group, 3
coef.CountsEPPM, 4
cooks.distance.CountsEPPM, 5
CountsEPPM, 6
CountsEPPM-package, 2

EPPMprob, 9

Faddyprob.general, 9
Faddyprob.limiting, 10
fitted, *11*
fitted.CountsEPPM, 11

hatvalues.CountsEPPM, 12
herons.case, 13
herons.group, 13

LL.gradient, 14
LL.Regression.Counts, 16
logLik.CountsEPPM, 18
LRTruncation, 19
Luningetal.litters, 19

Model.Counts, 20
Model.Faddy, 22
Model.FaddyJMV.general, 23
Model.FaddyJMV.limiting, 25

plot.betareg, *27*
plot.CountsEPPM, 27
predict.betareg, *29*
predict.CountsEPPM, 28
print.CountsEPPM, 29
print.summaryCountsEPPM, 30

residuals.betareg, *31*
residuals.CountsEPPM, 31

summary.CountsEPPM, 32

takeover.bids.case, 32
Titanic.survivors.case, 34

vcov.CountsEPPM, 35

waldtest, *37*
waldtest.CountsEPPM, 36
Williams.litters, 37