Package ‘mpcmp’

October 26, 2020

Type Package

Title Mean-Parametrized Conway-Maxwell Poisson (COM-Poisson) Regression

Version 0.3.6

Date 2020-10-23

Description A collection of functions for estimation, testing and diagnostic checking for the mean-parametrized Conway-Maxwell-Poisson (COM-Poisson) regression model of Huang (2017) <doi:10.1177/1471082X17697749>.

Depends R (>= 3.2)

License GPL-3

Imports Rcpp, graphics, grDevices, stats, ggplot2, ggpubr, generics, tibble, dplyr, rlang, stringr, purrr

URL https://github.com/thomas-fung/mpcmp

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, testthat, spelling, covr, markdown (>= 1.1), tidyverse, modelsummary, broom

BugReports https://github.com/thomas-fung/mpcmp/issues

LinkingTo Rcpp

Language en-GB

VignetteBuilder knitr

NeedsCompilation yes

Author Thomas Fung [aut, cre] (<https://orcid.org/0000-0003-2601-0728>),
Aya Alwan [aut],
Justin Wishart [aut] (<https://orcid.org/0000-0002-8235-1845>),
Alan Huang [aut]

Maintainer Thomas Fung <thomas.fung.dr@gmail.com>

Repository CRAN

Date/Publication 2020-10-26 06:30:03 UTC

R topics documented:

mpcmp-package *Mean-parametrized Conway-Maxwell Poisson Regression*

Description

Mean-parametrized Conway-Maxwell Poisson Regression

References

Fung, T., Alwan, A., Wishart, J. and Huang, A. (2019). `mpcmp`: Mean-parametrized Conway-Maxwell Poisson Regression. R package version 0.2.0.

Huang, A. (2017). Mean-parametrized Conway-Maxwell-Poisson regression models for dispersed counts. *Statistical Modelling* **17**, 359–380.

AIC.cmp *Akaike's Information Criterion*

Description

A function calculating Akaike's Information Criterion (AIC) based on the log-likelihood value extracted from `logLik.cmp`, according to the formula -2log-likelihood + k*npar*, where *npar* represents the number of parameters in the fitted model, and *k=2* for the usual AIC or *k=log(n)* (*n* being the number of observations) for the so-called BIC (Bayesian Information Criterion).

Usage

```
## S3 method for class 'cmp'
AIC(object, ..., k = 2)
```

Arguments

`object`	an object class 'cmp' object, obtained from a call to `glm.cmp`
`...`	other arguments passed to or from other methods (currently unused).
`k`	numeric: the *penalty* per parameter to be used; the default k = 2 is the classical AIC.

Details

When comparing models fitted by maximum likelihood to the same data, the smaller the AIC or BIC, the better the fit.

Value

A numeric value with the corresponding AIC (or BIC, or ..., depends on k).

See Also

`logLik.cmp`, `nobs.cmp`, `glm.cmp`

`attendance` *Attendance data set*

Description

This data set gives the number of days absent from high school and the gender, maths score (standardized score out of 100) and academic programme ('General', 'Academic' and 'Vocational') of 314 students sampled from two urban high schools. The attendance data frame has 314 observations on 5 variables.

Usage

```
data(attendance)
```

Format

A data frame with 314 observations on 5 variables.

id Identifier

gender gender

math standardized math score out of 100

daysabs number of days absent from high school

prog academic programme ('General', 'Academic' and 'Vocational')

Source

https://stats.idre.ucla.edu/stat/stata/dae/nb_data.dta

Examples

```
## For examples see example(glm.cmp)
```

augment.cmp *Augment data with information from a(n) CMP model object*

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Usage

```
## S3 method for class 'cmp'
augment(
  x,
  data = model.frame.cmp(x),
  newdata = NULL,
  type.predict = c("link", "response"),
  type.residuals = c("deviance", "pearson", "response"),
  se_fit = FALSE,
  ...
)
```

Arguments

`x` an object class 'cmp' object, obtained from a call to `glm.cmp`

`data` A base::data.frame or `tibble::tibble()` containing the original data that was used to produce the object x. Defaults to model.frame.cmp(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.

`newdata` A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.

`type.predict` Passed to `predict.cmp()` type argument. Defaults to `"link"`.

`type.residuals` Passed to `residuals.cmp()` type arguments. Defaults to `"deviance"`.

`se_fit` Logical indicating whether or not a .se.fit column should be added to the augmented output. Defaults to `FALSE`.

`...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.level = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that all columns used to fit the model are present.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a `tibble::tibble` with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble.

We are in the process of defining behaviours for models fit with various na.action arguments, but make no guarantees about behaviour when data is missing at this time.

Value

A `tibble::tibble()` with columns:

`.cooksd` Cooks distance.

`.fitted` Fitted or predicted value.

`.hat` Diagonal of the hat matrix.

`.resid` The difference between observed and fitted values.

`.se.fit` Standard errors of fitted values.

`.sigma` Estimated residual standard deviation when corresponding observation is dropped from model.

`.std.resid` Standardised residuals.

Examples

```
data(attendance)
M.attendance <- glm.cmp(daysabs~ gender+math+prog, data=attendance)
augment(M.attendance)
```

`autoplot.cmp` *Plot Diagnostic for a* `glm.cmp` *Object in ggplot style*

Description

`autoplot` uses ggplot2 to draw the diagnostic plots for a 'cmp' class object. `gg_plot` is an *alias* for it.

Usage

```
## S3 method for class 'cmp'
autoplot(
  object,
  which = c(1L, 2L, 6L, 8L),
  bins = 10,
  ask = TRUE,
  nrow = NULL,
  ncol = NULL,
  output_as_ggplot = TRUE,
  ...
)

gg_plot(
  object,
  which = c(1L, 2L, 6L, 8L),
  bins = 10,
  ask = TRUE,
  nrow = NULL,
  ncol = NULL,
  output_as_ggplot = TRUE,
  ...
)
```

Arguments

`object` an object class ’cmp’ object, obtained from a call to `glm.cmp`

`which` if a subset of plots is required, specify a subset of the numbers 1:8. See ’Details’ below.

`bins` numeric; the number of bins shown in the PIT histogram or the PIT Q-Q plot.

`ask` logical; if `TRUE`, the user is asked before each plot.

`nrow` numeric; (optional) number of rows in the plot grid.

`ncol` numeric; (optional) number of columns in the plot grid.

`output_as_ggplot` logical; if `TRUE`, the function would return a list of `ggplot` objects; if `FALSE`, the function would return an `ggarrange` object.

`...` other arguments passed to or from other methods (currently unused).

Details

Eight plots (selectable by `which`) are currently available: a plot of deviance residuals against fitted values, a non-randomized PIT histogram, a uniform Q-Q plot for non-randomized PIT, a histogram of the normal randomized residuals, a Q-Q plot of the normal randomized residuals, a Scale-Location plot of sqrt(| residuals |) against fitted values a plot of Cook’s distances versus row labels a plot of pearson residuals against leverage. By default, four plots (number 1, 2, 6, and 8 from this list of plots) are provided.

The 'Scale-Location' plot, also called 'Spread-Location' plot, takes the square root of the absolute standardized deviance residuals (*sqrt|E|*) in order to diminish skewness is much less skewed than than *|E|* for Gaussian zero-mean E.

The 'Scale-Location' plot uses the standardized deviance residuals while the Residual-Leverage plot uses the standardized pearson residuals. They are given as $R_i/\sqrt{1-h_{ii}}$ where h_{ii} are the diagonal entries of the hat matrix.

The Residuals-Leverage plot shows contours of equal Cook's distance for values of 0.5 and 1.

There are two plots based on the non-randomized probability integral transformation (PIT) using compPIT. These are a histogram and a uniform Q-Q plot. If the model assumption is appropriate, these plots should reflect a sample obtained from a uniform distribution.

There are also two plots based on the normal randomized residuals calculated using compnormRandPIT. These are a histogram and a normal Q-Q plot. If the model assumption is appropriate, these plots should reflect a sample obtained from a normal distribution.

Value

return a list of ggplot objects or a ggarrange object.

See Also

compPIT, compnormRandPIT, glm.cmp and plot.cmp.

Examples

```
data(takeoverbids)
M.bids <- glm.cmp(numbids ~ leglrest + rearest + finrest + whtknght
    + bidprem + insthold + size + sizesq + regulatn, data=takeoverbids)

## The default plots are shown
gg_plot(M.bids) # or autoplot(M.bids)

## The plots for the non-randomized PIT
gg_plot(M.bids, which = c(2,3)) # or autoplot(M.bids, which = c(2,3))
```

CBIND *Combine R Objects by Columns*

Description

Take a sequence of vector, matrix or data-frame arguments and combine them by columns. CBIND is used within the package over cbind to recycle the shorter arguments so that their number of rows would match.

Usage

```
CBIND(..., deparse.level = 1)
```

Arguments

`...` (generalized) vectors or matrices. These can be given as named arguments

`deparse.level` integer; deparse.level = 0 constructs no labels, deparse.level = 1 (the default) or > 1 constructs labels from the arguments names.

`cmplrtest` *Likelihood Ratio Test for nested COM-Poisson models*

Description

Perform a likelihood ratio chi-squared test between nested COM-Poisson models. The test statistics is calculated as 2*(llik- llik_0). The test statistics has degrees of freedom *r* where *r* is the difference in the number of parameters between the full and null models.

Usage

```
cmplrtest(object1, object2, digits = 3)
```

Arguments

`object1` an object class 'cmp', obtained from a call to `glm.cmp`

`object2` an object class 'cmp', obtained from a call to `glm.cmp`

`digits` numeric; minimum number of significant digits to be used for most numbers.

References

Huang, A. (2017). Mean-parametrized Conway-Maxwell-Poisson regression models for dispersed counts. *Statistical Modelling* **17**, 359–380.

See Also

`glm.cmp`, `update.cmp`

Examples

```
## Testing for the mean coefficients
data(takeoverbids)

## Fit full model
M.bids.full <- glm.cmp(numbids ~ leglrest + rearest + finrest + whtknght
    + bidprem + insthold + size + sizesq + regulatn, data=takeoverbids)

## Fit null model; without whtknght
M.bids.null <- update(M.bids.full, .~.-whtknght)

## Likelihood ratio test for the nested models
```

```
cmplrtest(M.bids.full, M.bids.null) # order of objects is not important

## Testing for dispersion coefficients
data(sitophilus)
 M.sit.full <- glm.cmp(formula = ninsect ~ extract, formula_nu = ~extract, data = sitophilus)

## Fit null model; dropping extract from dispersion equation
 M.sit.null1 <- update(M.sit.full, formula_nu. = ~1)
 cmplrtest(M.sit.null1, M.sit.full)

## Fit null model; using constant dispersion specification
M.sit.null2 <- update(M.sit.full, formula_nu. = NULL)
cmplrtest(M.sit.null2, M.sit.full)
```

coef.cmp *Extract Model Coefficients from a COM-Poisson Model Fit*

Description

An function used to extract model coefficients from a 'cmp' object. `coefficients` is an alias for `coef`.

Usage

```
## S3 method for class 'cmp'
coef(object, ...)
```

Arguments

`object`	an object class 'cmp' object, obtained from a call to `glm.cmp`
`...`	other arguments passed to or from other methods (currently unused).

Value

Coefficients extracted from the object `object`.

See Also

`fitted.cmp`, `residuals.cmp`, `glm.cmp`.

comp_expected_values *Functions to Compute Various Expected Values for the COM-Poisson Distribution*

Description

Functions to approximate the various expected values for the COM-Poisson distribution via truncation. The standard COM-Poisson parametrization is being used here. The lambda and nu values are recycled to match the length of the longer one and that would determine the length of the results.

Usage

```
comp_mean_logfactorialy(lambda, nu, log.Z, summax = 100)

comp_mean_ylogfactorialy(lambda, nu, log.Z, summax = 100)

comp_means(lambda, nu, log.Z, summax = 100)

comp_variances(lambda, nu, log.Z, summax = 100)

comp_variances_logfactorialy(lambda, nu, log.Z, summax = 100)
```

Arguments

`lambda, nu,`	rate and dispersion parameters. Must be positives.
`log.Z,`	an optional vector specifying normalizing constant Z in log scale.
`summax`	maximum number of terms to be considered in the truncated sum. The default is to sum to 100.

Value

`comp_mean_logfactorialy` gives the mean of *log(Y!)*.

`comp_mean_ylogfactorialy` gives the mean of *ylog(Y!)*.

`comp_means` gives the mean of *Y*.

`comp_variances` gives the variance of *Y*.

`comp_variances_logfactorialy` gives the variance of *log(Y!)*.

comp_lambdas *Solve for Lambda for a Particular Mean Parametrized COM-Poisson Distribution*

Description

Given a particular mean parametrized COM-Poisson distribution i.e. mu and nu, this function is used to find a lambda that can satisfy the mean constraint with a combination of bisection and Newton-Raphson updates. The function is also vectorized but will only update those that have not converged.

Usage

```
comp_lambdas(
  mu,
  nu,
  lambdalb = 1e-10,
  lambdaub = 1000,
  maxlambdaiter = 1000,
  tol = 1e-06,
  lambdaint = 1,
  summax = 100
)

comp_lambdas_fixed_ub(
  mu,
  nu,
  lambdalb = 1e-10,
  lambdaub = 1000,
  maxlambdaiter = 1000,
  tol = 1e-06,
  lambdaint = 1,
  summax = 100
)
```

Arguments

`mu, nu` mean and dispersion parameters. Must be straightly positive.

`lambdalb, lambdaub`
numeric; the lower and upper end points for the interval to be searched for lambda(s).

`maxlambdaiter` numeric; the maximum number of iterations allowed to solve for lambda(s).

`tol` numeric; the convergence threshold. A lambda is said to satisfy the mean constraint if the absolute difference between the calculated mean and the corresponding mu values is less than tol.

`lambdaint` numeric vector; initial gauss for lambda(s).

`summax` maximum number of terms to be considered in the truncated sum

Value

Both `comp_lambdas` and `comp_lambdas_fixed_ub` returns the lambda value(s) that satisfies the mean constraint(s) as well as the current lambdaub value. lambda value(s) returns by `comp_lambdas_fixed_ub` is bounded by the lambdaub value. `comp_lambdas` has the extra ability to scale up/down lambdaub to find the most appropriate lambda values.

`comp_mu_loglik` *Calculate the Log-Likelihood of the COM-Poisson model*

Description

A function to compute the log-likelihood of the COM-Poisson model.

Usage

```
comp_mu_loglik(param, y, xx, offset, summax)

comp_mu_neg_loglik_log_nu_only(log_nu, mu, y, summax)
```

Arguments

`param`	numeric vector: the model coefficients & the current value of `nu`. It is assumed that `nu` is in the last position of `param`.
`y`	numeric vector: response variable
`xx`	numeric matrix: the explanatory variables
`offset`	numeric vector: a vector of length equal to the number of cases
`summax`	maximum number of terms to be considered in the truncated sum
`log_nu`	numeric: nu in log-scale
`mu`	numeric vector: fitted mean parameters

Value

`comp_mu_loglik` returns the log-likelihood value of the COM-Poisson model based on Huang (2018). `comp_mu_neg_loglik_log_nu_only` returns the negative log-likelihood value of the COM-Poisson model based on Ribeiro Jr et al. (2018)'s specification to use in conjunction with `optim`.

COM_Poisson_Distribution *The Conway-Maxwell-Poisson (COM-Poisson) Distribution.*

Description

Density, distribution function, quantile function and random generation for the Conway-Maxwell-Poisson distribution with parameter mu and nu

Usage

```
dcomp(
  x,
  mu,
  nu = 1,
  lambda,
  log.p = FALSE,
  lambdalb = 1e-10,
  lambdaub = 1000,
  maxlambdaiter = 1000,
  tol = 1e-06,
  summax
)

pcomp(
  q,
  mu,
  nu = 1,
  lambda,
  lower.tail = TRUE,
  log.p = FALSE,
  lambdalb = 1e-10,
  lambdaub = 1000,
  maxlambdaiter = 1000,
  tol = 1e-06,
  summax
)

qcomp(
  p,
  mu,
  nu = 1,
  lambda,
  lower.tail = TRUE,
  log.p = FALSE,
  lambdalb = 1e-10,
  lambdaub = 1000,
  maxlambdaiter = 1000,
  tol = 1e-06,
  summax
)

rcomp(
  n,
  mu,
  nu = 1,
  lambda,
  lambdalb = 1e-10,
  lambdaub = 1000,
  maxlambdaiter = 1000,
  tol = 1e-06,
  summax
)
```

Arguments

`x, q` vector of quantiles

`mu, nu` mean and dispersion parameters. Must be strictly positive.

`lambda` an alternative way than mu to parametrized the distribution. Must be strictly positive

`log.p` logical; if `TRUE`, probabilities/densities p are returned as $log(p)$.

`lambdalb, lambdaub`
numeric: the lower and upper end points for the interval to be searched for lambda(s).

`maxlambdaiter` numeric: the maximum number of iterations allowed to solve for lambda(s).

`tol` numeric: the convergence threshold. A lambda is said to satisfy the mean constraint if the absolute difference between the calculated mean and mu is less than tol.

`summax` numeric; maximum number of terms to be considered in the truncated sum.

`lower.tail` logical; if `TRUE` (default), probabilities are $P(X \leq x)$, otherwise, $P(X > x)$.

`p` vector of probabilities

`n` number of observations. If `length(n)` > 1, the length is taken to be the number required.

Value

`dcomp` gives the density, `pcomp` gives the distribution function, `qcomp` gives the quantile function, and `rcomp` generates random deviates.

Invalid arguments will result in return value `NaN`, with a warning.

The length of the results is determined by `n` for `rcomp`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the results. Only the first argument of the logical arguments are used.

Examples

```
dcomp(0:5, mu = 2, nu = 1.2)
pcomp(5, mu=2, nu =1.2)
p <- (1:9)/10
qcomp(p, mu = 2, nu = 0.8)
rcomp(10, mu = 2, nu = 0.7)
```

---

confint.cmp *Confidence Intervals for CMP Model Parameters*

Description

Computes confidence intervals for one or more parameters in a fitted model.

Usage

```
## S3 method for class 'cmp'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	an object class 'cmp', obtained from a call to `glm.cmp`.
`parm`	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names (comparing to those provided by `coef()`) . If missing, all parameters are considered.
`level`	the confidence level required.
`...`	other arguments passed to or from other methods (currently unused).

Value

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter. These will be labelled as (1-level)/2 and 1 - (1-level)/2 in % (by default 2.5% and 97.5%).

Examples

```
data(attendance)
M.attendance <- glm.cmp(daysabs~ gender+math+prog, data=attendance)
confint(M.attendance)
confint(M.attendance, parm = "math", level = 0.9)
```

cottonbolls *Cotton Bolls data set*

Description

This data set gives the observed number of bolls produced by the cotton plants at five growth stages: vegetative, flower-bud, blossom, fig and cotton boll; to examine the effect of five defoliation levels (0\

Usage

```
data(cottonbolls)
```

Format

A data frame with 125 observations on 4 variables.

nc number of bolls produced by two cotton plants at harvest

stages growth stage

def artificial defoliation level

def2 square of def

Source

Supplementary Content of Zeviani et al. (2014): http://www.leg.ufpr.br/doku.php/publications:papercompanions:zeviani-jas2014

References

Zeviani, W.M., Riberio P.J. Jr., Bonat, W.H., Shimakura S.E. and Muniz J.A. (2014). The Gamma-count distribution in the analysis of experimental underdispersed data. *Journal of Applied Statistics* **41**, 2616–26.

Examples

```
### Huang (2017) Page 373--375: Underdispersed Cotton bolls data
### Model fitting for predictor V

data(cottonbolls)
M.bolls <- glm.cmp(nc~ 1+stages:def+stages:def2, data= cottonbolls)
M.bolls
summary(M.bolls)
```

fish *Fish data set*

Description

This data set gives the the number of fish species in lakes of the world; to examine the effect of the surface area of the lakes. The latitude of the lakes are also recorded.

Usage

```
data(fish)
```

Format

A data frame with 70 observations on 4 variables.

lake name of the lakes

species number of fish species in lakes

area surface area (km squared)

latitude latitude of the lakes

Details

This data set is also used to illustrate that the fitting algorithm can handle some larger count data.

References

Barbour, C. D. and Brown, J. H. (1974). Fish species diversity in lakes. *The American Naturalist*, **108**, 473–488.

Examples

```
### Barbour & Brown (1974): Overdispersed Fish data

data(fish)
M.fish <- glm.cmp(species~ 1+log(area), data=fish)
M.fish
summary(M.fish)
```

fitted.cmp *Extract Fitted Values from a COM-Poisson Model Fit*

Description

An accessor function used to extract the fitted values from a 'cmp' object. `fitted.values` is an alias for `fitted`.

Usage

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

Arguments

`object`	an object class 'cmp' object, obtained from a call to `glm.cmp`
`...`	other arguments passed to or from other methods (currently unused).

Value

Fitted values `mu` extracted from the object `object`.

See Also

`coef.cmp`, `residuals.cmp`, `glm.cmp`.

fit_glm_cmp_const_nu *Fit a Mean Parametrized Conway-Maxwell Poisson Generalized Linear Model with constant dispersion.*

Description

This is a workhorse function in which glm.cmp to call upon to fit a mean-parametrized Conway-Maxwell Poisson generalized linear model with constant dispersion.

Usage

```
fit_glm_cmp_const_nu(
  y = y,
  X = X,
  offset = offset,
  betastart = betastart,
  lambdalb = lambdalb,
  lambdaub = lambdaub,
  maxlambdaiter = maxlambdaiter,
  tol = tol
)
```

Arguments

`y` the response y vector.

`X` the design matrix for regressing the mean

`offset,` this can be used to specify an *a priori* known component to be included in the linear predictor for mean during fitting. This should be `NULL` or a numeric vector

`betastart` starting values for the parameters in the linear predictor for mu.

`lambdalb, lambdaub`
numeric: the lower and upper end points for the interval to be searched for lambda(s). The default value for lambdaub should be sufficient for small to moderate size nu. If nu is large and required a larger `lambdaub`, the algorithm will scale up `lambdaub` accordingly.

`maxlambdaiter` numeric: the maximum number of iterations allowed to solve for lambda(s).

`tol` numeric: the convergence threshold. A lambda is said to satisfy the mean constraint if the absolute difference between the calculated mean and a fitted values is less than tol.

Value

A fitted model object of class `cmp` similar to one obtained from `glm` or `glm.nb`.

Examples

```
## For examples see example(glm.cmp)
```

`fit_glm_cmp_vary_nu` *Fit a Mean Parametrized Conway-Maxwell Poisson Generalized Linear Model with varying dispersion.*

Description

Fit a Mean Parametrized Conway-Maxwell Poisson Generalized Linear Model with varying dispersion.

Usage

```
fit_glm_cmp_vary_nu(
  y = y,
  X = X,
  S = S,
  offset = offset,
  betastart = betastart,
  gammastart = gammastart,
  lambdalb = lambdalb,
  lambdaub = lambdaub,
  maxlambdaiter = maxlambdaiter,
  tol = tol
)
```

Arguments

`y`	the response y vector.
`X`	the design matrix for regressing the mean
`S`	the design matrix for regressing the dispersion
`offset`	this can be used to specify an *a priori* known component to be included in the linear predictor for mean during fitting. This should be `NULL` or a numeric vector
`betastart`	starting values for the parameters in the linear predictor for mu.
`gammastart`	starting values for the parameters in the linear predictor for nu.
`lambdalb, lambdaub`	numeric: the lower and upper end points for the interval to be searched for lambda(s). The default value for lambdaub should be sufficient for small to moderate size nu. If nu is large and required a larger `lambdaub`, the algorithm will scale up `lambdaub` accordingly.
`maxlambdaiter`	numeric: the maximum number of iterations allowed to solve for lambda(s).
`tol`	numeric: the convergence threshold. A lambda is said to satisfy the mean constraint if the absolute difference between the calculated mean and a fitted values is less than tol.

Value

A fitted model object of class `cmp` similar to one obtained from `glm` or `glm.nb`.

Examples

```
## For examples see example(glm.cmp)
```

`getnu` *Parameter Generator for nu*

Description

A function that use the arguments of a `glm.cmp` call to generate a better initial `nu` estimate.

Usage

```
getnu(
  param,
  y,
  xx,
  offset,
  llstart,
  fsscale = 1,
  lambdalb = 1e-10,
  lambdaub = 1000,
  maxlambdaiter = 1000,
  tol = 1e-06,
  summax = 100
)
```

Arguments

param	numeric vector: the model coefficients & the current value of nu. It is assumed that nu is in the last position of param.
y	numeric vector: response variable
xx	numeric matrix: the explanatory variables
offset	numeric vector: a vector of length equal to the number of cases
llstart	numeric: current log-likelihood value
fsscale	numeric: a scaling factor (generally >1) for the relaxed fisher scoring algorithm
lambdalb, lambdaub	numeric: the lower and upper end points for the interval to be searched for lambda(s).
maxlambdaiter	numeric: the maximum number of iterations allowed to solve for lambda(s).
tol	numeric: the convergence threshold. A lambda is said to satisfy the mean constraint if the absolute difference between the calculated mean and a fitted values is less than tol.
summax	maximum number of terms to be considered in the truncated sum

Details

From version 0.3.4, this function is no longer being used as part of the estimation algorithm and this function will be defunct in our next update.

Value

List containing the following:

param	the model coefficients & the updated nu
maxl	the updated log-likelihood
fsscale	the final scaling factor used

glance.cmp *Glance at a(n) CMP model object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Usage

```
## S3 method for class 'cmp'
glance(x, ...)
```

Arguments

`x`	an object class 'cmp' object, obtained from a call to `glm.cmp`
`...`	other arguments passed to or from other methods (currently unused).

Details

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Value

A `tibble::tibble()` with exactly one row and columns:

`AIC`	Akaike's Information Criterion for the model.
`BIC`	Bayesian Information Criterion for the model.
`deviance`	(Residual) Deviance of the model.
`df.null`	Degrees of freedom used by the null model.
`df.residual`	Residual degrees of freedom.
`logLik`	The log-likelihood of the model.
`nobs`	Number of observations used.
`null.deviance`	Deviance of the null model.

Examples

```
data(attendance)
M.attendance <- glm.cmp(daysabs~ gender+math+prog, data=attendance)
glance(M.attendance)
```

---

glm.cmp *Fit a Mean Parametrized Conway-Maxwell Poisson Generalized Linear Model*

Description

The function glm.cmp is used to fit a mean parametrized Conway-Maxwell Poisson generalized linear model with a log-link by using Fisher Scoring iteration.

Usage

```
glm.cmp(
  formula,
  formula_nu = NULL,
  data,
  offset = NULL,
  subset,
  na.action,
  betastart = NULL,
  gammastart = NULL,
  lambdalb = 1e-10,
  lambdaub = 1000,
  maxlambdaiter = 1000,
  tol = 1e-06,
  contrasts_mu = NULL,
  contrasts_nu = NULL
)
```

Arguments

formula an object of class ’formula’: a symbolic description of the model to be fitted to the mean via log-link.

formula_nu an optional object of class ’formula’: a symbolic description of the model to be fitted to the dispersion via log-link.

data an optional data frame containing the variables in the model

offset this can be used to specify an *a priori* known component to be included in the linear predictor for mean during fitting. This should be NULL or a numeric vector of length equal to the number of cases.

subset an optional 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. Another possible value is NULL, no action. Value na.exclude can be useful.

`betastart` starting values for the parameters in the linear predictor for mu.

`gammastart` starting values for the parameters in the linear predictor for nu.

`lambdalb, lambdaub`
numeric: the lower and upper end points for the interval to be searched for lambda(s). The default value for lambdaub should be sufficient for small to moderate size nu. If nu is large and required a larger `lambdaub`, the algorithm will scale up `lambdaub` accordingly.

`maxlambdaiter` numeric: the maximum number of iterations allowed to solve for lambda(s).

`tol` numeric: the convergence threshold. A lambda is said to satisfy the mean constraint if the absolute difference between the calculated mean and a fitted values is less than tol.

`contrasts_mu, contrasts_nu`
optional lists. See the contrasts.arg of model.matrix.default.

Details

Fit a mean-parametrized COM-Poisson regression using maximum likelihood estimation via an iterative Fisher Scoring algorithm.

Currently, the COM-Poisson regression model allows constant dispersion and regression being linked to the dispersion parameter i.e. varying dispersion.

For the constant dispersion model, the model is

$$Y_i\,CMP(\mu_i, \nu),$$

where

$$E(Y_i) = \mu_i = exp(x_i^T\beta),$$

and $\nu > 0$ is the dispersion parameter.

The fitted COM-Poisson distribution is over- or under-dispersed if $\nu < 1$ and $\nu > 1$ respectively.

For the varying dispersion model, the model is

$$Y_i\,CMP(\mu_i, \nu_i),$$

where

$$E(Y_i) = \mu_i = exp(x_i^T\beta),$$

and dispersion parameters are model via

$$\nu_i = exp(s_i^T\gamma),$$

where x_i and s_i are some covariates.

Value

A fitted model object of class `cmp` similar to one obtained from `glm` or `glm.nb`.

The function `summary` (i.e., `summary.cmp`) can be used to obtain and print a summary of the results.

The functions `plot` (i.e., `plot.cmp`) and `gg_plot` can be used to produce a range of diagnostic plots.

The generic assessor functions `coef` (i.e., `coef.cmp`), `logLik` (i.e., `logLik.cmp`) `fitted` (i.e., `fitted.cmp`), `nobs` (i.e., `nobs.cmp`), `AIC` (i.e., `AIC.cmp`) and `residuals` (i.e., `residuals.cmp`) can be used to extract various useful features of the value returned by `glm.cmp`.

The functions `LRTnu` and `cmplrtest` can be used to perform a likelihood ratio chi-squared test for nu = 1 and for nested COM-Poisson model respectively.

An object class 'glm.cmp' is a list containing at least the following components:

`coefficients`	a named vector of coefficients
`coefficients_beta`	a named vector of mean coefficients
`coefficients_gamma`	a named vector of dispersion coefficients
`se_beta`	approximate standard errors (using observed rather than expected information) for mean coefficients
`se_gamma`	approximate standard errors (using observed rather than expected information) for dispersion coefficients
`residuals`	the *response* residuals (i.e., observed-fitted)
`fitted_values`	the fitted mean values
`rank_mu`	the numeric rank of the fitted linear model for mean
`rank_nu`	the numeric rank of the fitted linear model for dispersion
`linear_predictors`	the linear fit for mean on log scale
`df_residuals`	the residuals degrees of freedom
`df_null`	the residual degrees of freedom for the null model
`null_deviance`	The deviance for the null model. The null model will include only the intercept.
`deviance; residual_deviance`	The residual deviance of the model
`y`	the `y` vector used.
`x`	the model matrix for mean
`s`	the model matrix for dispersion
`model_mu`	the model frame for mu
`model_nu`	the model frame for nu
`call`	the matched call
`formula`	the formula supplied for mean
`formula_nu`	the formula supplied for dispersion

`terms_mu`	the `terms` object used for mean
`terms_nu`	the `terms` object used for dispersion
`data`	the `data` argument
`offset`	the `offset` vector used
`lambdaub`	the final `lambdaub` used

References

Fung, T., Alwan, A., Wishart, J. and Huang, A. (2020). `mpcmp`: Mean-parametrized Conway-Maxwell Poisson Regression. R package version 0.3.4.

Huang, A. (2017). Mean-parametrized Conway-Maxwell-Poisson regression models for dispersed counts. *Statistical Modelling* **17**, 359–380.

See Also

`summary.cmp`, `autoplot.cmp`, `plot.cmp`, `fitted.cmp`, `residuals.cmp` and `LRTnu`.

Additional examples may be found in `fish`, `takeoverbids`, `cottonbolls`.

Examples

```
### Huang (2017) Page 368--370: Overdispersed Attendance data
data(attendance)
M.attendance <- glm.cmp(daysabs~ gender+math+prog, data=attendance)
M.attendance
summary(M.attendance)
plot(M.attendance) # or autoplot(M.attendance)


### Ribeiro et al. (2013): Varying dispersion as a function of covariates
data(sitophilus)
M.sit <- glm.cmp(formula = ninsect ~ extract, formula_nu = ~extract, data = sitophilus)
summary(M.sit)
```

`is.wholenumber` *Test for a whole number*

Description

Test for integer/whole number vector

Usage

```
is.wholenumber(x, tol = .Machine$double.eps^0.5)
```

Arguments

`x`	numeric vector to be tested
`tol`	numeric; precision level

`logLik.cmp` *Extract the (Maximized) Log-Likelihood from a COM-Poisson Model Fit*

Description

An accessor function used to extract the (maximized) log-likelihood from a 'cmp' object.

Usage

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

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

Arguments

`object`	an object of class 'cmp' object, obtained from a call to `glm.cmp`
`...`	other arguments passed to or from other methods (currently unused).
`x`	an object of class 'logLik.cmp', obtained from a call to `logLik.cmp`.

See Also

`coef.cmp`, `fitted.cmp`, `glm.cmp`

`logZ` *Calculate the Normalizing Constant in log scale for COM-Poisson distribution*

Description

A function to approximate the normalizing constant for COM-Poisson distributions via truncation. The standard COM-Poisson parametrization is being used here.

Usage

```
logZ(log_lambda, nu, summax = 100)
```

Arguments

`log_lambda`	rate parameter in log scale.
`nu`	dispersion parameter, straightly positive.
`summax`	maximum number of terms to be considered in the truncated sum.

Details

As of version 0.2.0 of this package, `logZ` will supersede `Z` for calculating the normalizing constant. `logZ` utilised a method that can calculate `log(exp(logx) + exp(logy))` in a somewhat numerically stable way.

This function was originally purposed in the `cmpreg` package of Ribeiro Jr, Zeviani & Demétrio (2019).

References

Ribeiro Jr, E. E., Zeviani, W. M., Demétrio, C. G. B. (2019) `cmpreg`: Reparametrized COM-Poisson Regression Models. R package version 0.0.1.

`logZ_c` *Calculate the Normalizing Constant in log scale for COM-Poisson distribution The calculation of the function `logZ` will be performed here. This function is used to approximate the normalizing constant for COM-Poisson distributions via truncation. The standard COM-Poisson parametrization is being used here.*

Description

It is assumed that vectors log_lambda & nu are of equal length.

Usage

```
logZ_c(log_lambda, nu, summax)
```

Arguments

`log_lambda`	rate parameter in log scale.
`nu`	dispersion parameter, straightly positive.
`summax`	maximum number of terms to be considered in the truncated sum.

Details

This function was originally purposed in the `cmpreg` package of Ribeiro Jr, Zeviani & Demétrio (2019).

References

Ribeiro Jr, E. E., Zeviani, W. M., Demétrio, C. G. B. (2019) `cmpreg`: Reparametrized COM-Poisson Regression Models. R package version 0.0.1.

LRTnu *Likelihood Ratio Test for nu = 1 of a COM-Poisson model*

Description

Perform a likelihood ratio chi-squared test for nu = 1 of a COM-Poisson model. The test statistics is calculated as *2*(llik- llik_0)* where *llik* and *llik_0* are the log-likelihood of a COM-Poisson and Poisson model respectively. The test statistic has 1 degrees of freedom.

Usage

```
LRTnu(object, digits = 3)
```

Arguments

`object` an object class 'cmp', obtained from a call to `glm.cmp`

`digits` numeric; minimum number of significant digits to be used for most numbers.

References

Huang, A. (2017). Mean-parametrized Conway-Maxwell-Poisson regression models for dispersed counts. *Statistical Modelling* **17**, 359–380.

Examples

```
data(takeoverbids)
M.bids <- glm.cmp(numbids ~ leglrest + rearest + finrest + whtknght
    + bidprem + insthold + size + sizesq + regulatn, data=takeoverbids)
LRTnu(M.bids)
```

model.frame.cmp *Extract the Model Frame from a COM-Poisson Model Fit*

Description

An accessor function used to extract the model frame from a 'cmp' object.

Usage

```
## S3 method for class 'cmp'
model.frame(formula, ...)
```

Arguments

`formula` an object class 'cmp' object, obtained from a call to `glm.cmp`

`...` other arguments passed to or from other methods (currently unused).

Value

The method will return the saved `data.frame` used when fitting the cmp model.

See Also

`coef.cmp`, `residuals.cmp`, `glm.cmp`.

`model.matrix.cmp` *Extract the Design Matrix from a COM-Poisson Model Fit*

Description

Extract the Design Matrix from a COM-Poisson Model Fit

Usage

```
## S3 method for class 'cmp'
model.matrix(object, ...)
```

Arguments

`object`	an object class 'cmp' object, obtained from a call to `glm.cmp`
`...`	other arguments passed to or from other methods (currently unused).

Value

The method will return the saved `model.matrix` used when fitting the cmp model.

Examples

```
data(attendance)
M.attendance <- glm.cmp(daysabs~ gender+math+prog, data=attendance)
model.matrix(M.attendance)


data(sitophilus)
M.sit <- glm.cmp(formula = ninsect ~ extract, formula_nu = ~extract, data = sitophilus)
model.matrix(M.sit)
```

nobs.cmp *Extract the Number of Observation from a COM-Poisson Model Fit*

Description

An accessor function used to extract the number of observation from a 'cmp' object.

Usage

```
## S3 method for class 'cmp'
nobs(object, ...)
```

Arguments

`object`	an object class 'cmp' object, obtained from a call to `glm.cmp`
`...`	other arguments passed to or from other methods (currently unused).

Value

The number of observations extracted from the object `object`.

See Also

`coef.cmp`, `fitted.cmp`, `glm.cmp`

nrPIT *Non-randomized Probability Integral Transform*

Description

Functions to produce the non-randomized probability integral transform (PIT) to check the adequacy of the distributional assumption of the COM-Poisson model. The majority of the code and descriptions are taken from Dunsmuir and Scott (2015).

Usage

```
compPredProb(object)

compPIT(object, bins = 10)
```

Arguments

`object`	an object class "cmp", obtained from a call to `glm.cmp`.
`bins`	numeric; the number of bins shown in the PIT histogram or the PIT Q-Q plot.

Details

These functions are used to obtain the predictive probabilities and the probability integral transform for a fitted COM-Poisson model. The majority of the code and descriptions are taken from Dunsmuir and Scott (2015).

Value

`compPredprob` returns a list with values:

`upper` the predictive cumulative probabilities used as the upper bound for computing the non-randomized PIT.

`lower` the predictive cumulative probabilities used as the upper bound for computing the non-randomized PIT.

`compPIT` returns a list with values:

`conditionalPIT` the conditional probability integral transformation given the observed counts.

`PIT` the probability integral transformation.

References

Czado, C., Gneiting, T. and Held, L. (2009). Predictive model assessment for count data. *Biometrics*, **65**, 1254–1261.

Dunsmuir, W.T.M. and Scott, D.J. (2015). The `glarma` Package for Observation-Driven Time Series Regression of Counts. *Journal of Statistical Software*, **67**, 1–36.

Examples

```
data(takeoverbids)
M.bids <- glm.cmp(numbids ~ leglrest + rearest + finrest + whtknght
    + bidprem + insthold + size + sizesq + regulatn, data=takeoverbids)
compPredProb(M.bids)
compPIT(M.bids)
```

`PIT_ggPlot` *ggplot version of PIT Plots for a CMP Object*

Description

Two plots for the non-randomized PIT are currently available for checking the distributional assumption of the fitted CMP model: the PIT histogram, and the uniform Q-Q plot for PIT.

Usage

```
gg_histcompPIT(
  object,
  bins = 10,
  ref_line = TRUE,
  col_line = "red",
  col_hist = "royal blue",
  size = 1
)

gg_qqcompPIT(
  object,
  bins = 10,
  col1 = "red",
  col2 = "#999999",
  lty1 = 1,
  lty2 = 2
)
```

Arguments

object	an object class "cmp", obtained from a call to glm.cmp.
bins	numeric; the number of bins shown in the PIT histogram or the PIT Q-Q plot.
ref_line	logical; if TRUE (default), the line for displaying the standard uniform distribution will be shown for the purpose of comparison.
col_line	numeric or character: the colour of the reference line for comparison in PIT histogram.
col_hist	numeric or character; the colour of the histogram for PIT.
size	numeric; the line widths for the comparison line in PIT histogram.
col1	numeric or character; the colour of the sample uniform Q-Q plot in PIT.
col2	numeric or character; the colour of the theoretical uniform Q-Q plot in PIT.
lty1	integer or character string: the line types for the sample uniform Q-Q plot in PIT, see ggplot2::linetype.
lty2	an integer or character string: the line types for the theoretical uniform Q-Q plot in PIT, see ggplot2::linetype.

Details

histcompPIT and qqcompPIT

The histogram and the Q-Q plot are used to compare the fitted profile with a standard uniform distribution. If they match relatively well, it means the CMP distribution is appropriate for the data.

The histcompPIT and qqcompPIT functions would provide the same two plots but in base R format.

References

Czado, C., Gneiting, T. and Held, L. (2009). Predictive model assessment for count data. *Biometrics*, **65**, 1254–1261.

Dunsmuir, W.T.M. and Scott, D.J. (2015). The glarma Package for Observation-Driven Time Series Regression of Counts. *Journal of Statistical Software*, **67**, 1–36.

See Also

histcompPIT, qqcompPIT, plot.cmp and autoplot.

Examples

```
## For examples see example(autoplot)
```

PIT_Plot *PIT Plots for a CMP Object*

Description

Two plots for the non-randomized PIT are currently available for checking the distributional assumption of the fitted CMP model: the PIT histogram, and the uniform Q-Q plot for PIT.

Usage

```
histcompPIT(
  object,
  bins = 10,
  line = TRUE,
  colLine = "red",
  colHist = "royal blue",
  lwdLine = 2,
  main = NULL,
  ...
)

qqcompPIT(
  object,
  bins = 10,
  col1 = "red",
  col2 = "black",
  lty1 = 1,
  lty2 = 2,
  type = "l",
  main = NULL,
  ...
)
```

Arguments

`object` an object class "cmp", obtained from a call to `glm.cmp`.

`bins` numeric; the number of bins shown in the PIT histogram or the PIT Q-Q plot.

`line` logical; if `TRUE` (default), the line for displaying the standard uniform distribution will be shown for the purpose of comparison.

`colLine` numeric or character: the colour of the line for comparison in PIT histogram.

`colHist` numeric or character; the colour of the histogram for PIT.

`lwdLine` numeric; the line widths for the comparison line in PIT histogram.

`main` character string; a main title for the plot.

`...` other arguments passed to plot.default and plot.ts.

`col1` numeric or character; the colour of the sample uniform Q-Q plot in PIT.

`col2` numeric or character; the colour of the theoretical uniform Q-Q plot in PIT.

`lty1` integer or character string: the line types for the sample uniform Q-Q plot in PIT, see par(lty = .).

`lty2` an integer or character string: the line types for the theoretical uniform Q-Q plot in PIT, see par(lty = .).

`type` 1-character string; the type of plot for the sample uniform Q-Q plot in PIT.

Details

The histogram and the Q-Q plot are used to compare the fitted profile with a standard uniform distribution. If they match relatively well, it means the CMP distribution is appropriate for the data.

The `gg_histcompPIT` and `gg_qqcompPIT` functions would provide the same two plots but in ggplot format.

References

Czado, C., Gneiting, T. and Held, L. (2009). Predictive model assessment for count data. *Biometrics*, **65**, 1254–1261.

Dunsmuir, W.T.M. and Scott, D.J. (2015). The `glarma` Package for Observation-Driven Time Series Regression of Counts. *Journal of Statistical Software*, **67**, 1–36.

See Also

`gg_histcompPIT`, `gg_qqcompPIT`, `plot.cmp` and `autoplot`.

Examples

```
## For examples see example(plot.cmp)
```

plot.cmp *Plot Diagnostic for a* glm.cmp *Object*

Description

Eight plots (selectable by which) are currently available: a plot of deviance residuals against fitted values, a non-randomized PIT histogram, a uniform Q-Q plot for non-randomized PIT, a histogram of the normal randomized residuals, a Q-Q plot of the normal randomized residuals, a Scale-Location plot of sqrt(| residuals |) against fitted values a plot of Cook's distances versus row labels a plot of pearson residuals against leverage. By default, four plots (number 1, 2, 6, and 8 from this list of plots) are provided.

Usage

```
## S3 method for class 'cmp'
plot(
  x,
  which = c(1L, 2L, 6L, 8L),
  ask = prod(par("mfcol")) < length(which) && dev.interactive(),
  bins = 10,
  ...
)
```

Arguments

x	an object class 'cmp' object, obtained from a call to glm.cmp
which	if a subset of plots is required, specify a subset of the numbers 1:8. See 'Details' below.
ask	logical; if TRUE, the user is asked before each plot.
bins	numeric; the number of bins shown in the PIT histogram or the PIT Q-Q plot.
...	other arguments passed to or from other methods (currently unused).

Details

The 'Scale-Location' plot, also called 'Spread-Location' plot, takes the square root of the absolute standardized deviance residuals (*sqrt*|*E*|) in order to diminish skewness is much less skewed than than |*E*| for Gaussian zero-mean E.

The 'Scale-Location' plot uses the standardized deviance residuals while the Residual-Leverage plot uses the standardized pearson residuals. They are given as $R_i/\sqrt{1-h_{ii}}$ where h_{ii} are the diagonal entries of the hat matrix.

The Residuals-Leverage plot shows contours of equal Cook's distance for values of 0.5 and 1.

There are two plots based on the non-randomized probability integral transformation (PIT) using compPIT. These are a histogram and a uniform Q-Q plot. If the model assumption is appropriate, these plots should reflect a sample obtained from a uniform distribution.

There are also two plots based on the normal randomized residuals calculated using compnormRandPIT. These are a histogram and a normal Q-Q plot. If the model assumption is appropriate, these plots should reflect a sample obtained from a normal distribution.

See Also

compPIT, compnormRandPIT, glm.cmp and autoplot.

Examples

```
data(takeoverbids)
M.bids <- glm.cmp(numbids ~ leglrest + rearest + finrest + whtknght
    + bidprem + insthold + size + sizesq + regulatn, data=takeoverbids)

## The default plots are shown
plot(M.bids)

## The plots for the non-randomized PIT
# plot(M.bids, which = c(2,3))
```

---

predict.cmp *Model Predictions for a glm.cmp Object*

Description

This is a function for obtaining predictions and optionally estimates standard errors of those prediction from a fitted COM-Poisson regression object.

Usage

```
## S3 method for class 'cmp'
predict(
  object,
  newdata = NULL,
  se.fit = FALSE,
  type = c("link", "response"),
  ...
)
```

Arguments

object	an object class ’cmp’, obtained from a call to glm.cmp.
newdata	optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
se.fit	logical; indicating if standard errors are required.
type	the type of prediction required. The default is ’link’ which is the scale of the linear predictor i.e., a log scale; the alternative ’response’ is on the scale of the response variable. The value of this argument can be abbreviated.
...	other arguments passed to or from other methods (currently unused).

Details

If newdata is omitted the predictions are based on the data used for the fit.

Value

If `se.fit` = `FALSE`, a vector of predictions.

If `se.fit` = `TRUE`, a list with components

`fit` Predictions, as for se.fit = FALSE.

`se.fit` Estimated standard errors.

Examples

```
data(takeoverbids)
M.bids <- glm.cmp(numbids ~ leglrest + rearest + finrest + whtknght
    + bidprem + insthold + size + sizesq + regulatn, data=takeoverbids)

predict(M.bids)
predict(M.bids, type= "response")
predict(M.bids, se.fit=TRUE, type="response")

newdataframe <- data.frame(bidprem = 1, finrest = 0, insthold = 0.05,
    leglrest = 0, rearest = 1, regulatn = 0, size = 0.1, whtknght = 1,
    sizesq = .1^2)
predict(M.bids, se.fit=TRUE, newdata = newdataframe, type="response")
```

`print.cmp` *Print Values of COM-Poisson Model*

Description

`print` method for class `cmp`.

Usage

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

Arguments

`x` an object class 'cmp', obtained from a call to `glm.cmp`.

`...` other arguments passed to or from other methods (currently unused).

Details

`print.cmp` can be used to print a short summary of object class 'cmp'.

See Also

summary.cmp, coef.cmp, fitted.cmp, glm.cmp.

Examples

```
## For examples see example(glm.cmp)
```

---

regression.diagnostic.cmp

CMP Regression Diagnostic

Description

This suite of functions provides the basic quantities which are used in forming a wide variety of diagnostics for checking the quality of regression fits.

Usage

```
## S3 method for class 'cmp'
influence(model, ...)

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

## S3 method for class 'cmp'
rstandard(
  model,
  infl = influence.cmp(model),
  type = c("deviance", "pearson"),
  ...
)

## S3 method for class 'cmp'
cooks.distance(
  model,
  infl = influence(model),
  res = infl$pear_res,
  hat = infl$hat,
  ...
)
```

Arguments

`model`	an object class 'cmp', obtained from a call to glm.cmp.
`...`	other arguments passed to or from other methods (currently unused).
`infl`	influence structure as returned by influence, only for rstudent and cooks.distance.

`type` type of residuals for `rstandard`. The alternatives are: ’deviance’ (default), and ’pearson’.

`res` residuals, with proper default.

`hat` hat values $H[i, i]$, see default.

Examples

```
data(attendance)
M.attendance <- glm.cmp(daysabs~ gender+math+prog, data=attendance)
influence(M.attendance)
hatvalues(M.attendance)
rstandard(M.attendance, type = "pearson")
cooks.distance(M.attendance)
```

`residuals.cmp` *Extract COM-Poisson Model Residuals*

Description

`residuals` is a generic function which extracts model residuals from objects returned by the modelling function `glm.comp`. `resid` is an alias for `residuals` .

Usage

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

Arguments

`object` an object class ’cmp’, obtained from a call to `glm.cmp`.

`type` the `type` of residuals which should be returned. The alternatives are: ’deviance’ (default), ’pearson’ and ’response’. Can be abbreviated.

`...` other arguments passed to or from other methods (currently unused).

Value

Residuals extracted from the object `object`.

See Also

`coef.cmp`, `fitted.cmp`, `glm.cmp`

rPIT *Random Normal Probability Integral Transform*

Description

A function to create the normal conditional (randomized) quantile residuals. The majority of the code and descriptions are taken from Dunsmuir and Scott (2015).

Usage

```
compnormRandPIT(object)
```

Arguments

`object`	an object class "cmp", obtained from a call to `glm.cmp`.

Details

The function `compPredProb` produces the non-randomized probability integral transform(PIT). It returns estimates of the cumulative predictive probabilities as upper and lower bounds of a collection of intervals. If the model is correct, a histogram drawn using these estimated probabilities should resemble a histogram obtained from a sample from the uniform distribution.

This function aims to produce observations which instead resemble a sample from a normal distribution. Such a sample can then be examined by the usual tools for checking normality, such as histograms and normal Q-Q plots.

For each of the intervals produced by `compPredProb`, a random uniform observation is generated, which is then converted to a normal observation by applying the inverse standard normal distribution function (using `qnorm`). The vector of these values is returned by the function in the list element `rt`. In addition non-random observations which should appear similar to a sample from a normal distribution are obtained by applying `qnorm` to the mid-points of the predictive distribution intervals. The vector of these values is returned by the function in the list element `rtMid`.

Value

A list consisting of two elements:

`rt`	the normal conditional randomized quantile residuals
`rdMid`	the midpoints of the predictive probability intervals

References

Berkowitz, J. (2001). Testing density forecasts, with applications to risk management. *Journal of Business \& Economic Statistics*, **19**, 465–474.

Dunn, P. K. and Smyth, G. K. (1996). Randomized quantile residuals. *Journal of Computational and Graphical Statistics*, **5**, 236–244.

Dunsmuir, W.T.M. and Scott, D.J. (2015). The `glarma` Package for Observation-Driven Time Series Regression of Counts. *Journal of Statistical Software*, **67**, 1–36.

Examples

```
data(takeoverbids)
M.bids <- glm.cmp(numbids ~ leglrest + rearest + finrest + whtknght
    + bidprem + insthold + size + sizesq + regulatn, data=takeoverbids)
compnormRandPIT(M.bids)
```

---

sitophilus *Sitophilus data set*

Description

Ribeiro et al. (2013) carried out an experiment to assess the bioactivity of extracts from different parts (seeds, leaves and branches) of Annona mucosa (Annonaceae) to control Sitophilus zeamais (Coleoptera: Curculionidae), a major pest of stored maize/corn in Brazil.

Usage

```
data(sitophilus)
```

Format

A data frame with 40 observations on 2 variables.

extract the treatment used

ninsect number emerged insects (progeny)

Details

10g of corn and 20 animals adults were placed in each Petri dish. Extracts prepared with different parts of mucosa or just water (control) were completely randomized with 10 replicates.

The numbers of emerged insects (progeny) after 60 days and their corresponding treatments were recorded in this dataset.

This dataset was obtained from the ‘cmpreg‘‘package of Ribeiro Jr, Zeviani & Demétrio (2019), which is based on the work of Ribeiro Junior et al. (2019).

This data set is also used to illustrate the syntax for regression on the dispersion.

References

Ribeiro Junior, E.E., Zeviani, W.M., Bonat, W.H., Demétrio, C.G., & Hinde, J. (2019). Reparametrization of COM–Poisson regression models with applications in the analysis of experimental data. Statistical Modelling. https://doi.org/10.1177%2F1471082X19838651.

Ribeiro, L.P., Vendramim, J.D., Bicalho, K.U., Andrade, M.S., Fernandes, J.B., Moral, R.A., & Demétrio C.G.B. (2013). Annona mucosa Jacq. (Annonaceae): A promising source of bioactive compounds against Sitophilus zeamais Mots. (Coleoptera: Curculionidae). *Journal of Stored Products Research*, **55**, 6-14.

Examples

```
## For examples see example(glm.cmp)
```

---

summary.cmp *Summarizing COM-Poisson Model Fit*

Description

`summary` method for class `cmp`.

Usage

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

## S3 method for class 'summary.cmp'
print(
  x,
  digits = max(3, getOption("digits") - 3),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

Arguments

`object`	an object class 'cmp', obtained from a call to `glm.cmp`.
`...`	other arguments passed to or from other methods (currently unused).
`x`	a result of the *default* method of `summary()`.
`digits`	numeric; minimum number of significant digits to be used for most numbers.
`signif.stars`	logical. If TRUE, 'significance stars' are printed for each coefficient.

Details

`print.summary.glm` tries to be smart about formatting the coefficients, standard errors and gives 'significance stars'. The `coefficients` component of the result gives the estimated coefficients and their estimated standard errors, together with their ratio. This third column is labelled as `Z value` as the dispersion is fixed for this family. A forth column gives the two-tailed p-value corresponding to `Z value` based on the asymptotic Normal reference distribution.

Value

`summary.cmp` returns an object of class "summary.cmp", a list containing at least the following components:

`call`	the component from object.
`family`	the component from object.

`deviance; residual_deviance`
the component from object.

`df_residual` the component from object.

`df_null` the component from object.

`null_deviance` the component from object.

`deviance_resid` the deviance residuals: see residuals.cmp.

`coefficients` the matrix of coefficients, standard errors, z-values and p-values.

`df` a 3-vector of the rank of the model and the number of residual degrees of freedom, plus number of coefficients.

See Also

`coef.cmp`, `fitted.cmp`, `glm.cmp`.

Examples

```
## For examples see example(glm.cmp)
```

`takeoverbids` *Takeover Bids data set*

Description

This data set gives the number of bids received by 126 US firms that were successful targets of tender offers during the period 1978–1985, along with some explanatory variables on the defensive actions taken by management of target firm, firm-specific characteristics and intervention taken by federal regulators. The `takeoverbids` data frame has 126 observations on 14 variables. The descriptions below are taken from Sáez-Castillo and Conde-Sánchez (2013).

Usage

```
data(takeoverbids)
```

Format

A data frame with 126 observations on 14 variables.

bidprem bid price divided by price 14 working days before bid

docno doc no

finrest indicator variable for proposed change in ownership structure

insthold percentage of stock held by institutions

leglrest indicator variable for legal defence by lawsuit

numbids number of bids received after the initial bid

obs Identifier

rearest indicator variable for proposed changes in asset structure

regulatn indicator variable for Department of Justice intervention

size total book value of assets in billions of dollars

takeover Indicator. 1 if the company was being taken over

weeks time in weeks between the initial and final offers

whtknght indicator variable for management invitation for friendly third-party bid

sizesq book value squared

Source

Journal of Applied Econometrics data archive: http://qed.econ.queensu.ca/jae/.

References

Cameron, A.C. and Johansson, P. (1997). Count Data Regression Models using Series Expansions: with Applications. *Journal of Applied Econometrics* **12** 203–223.

Cameron, A.C. and Trivedi P.K. (1998). Regression analysis of count data, Cambridge University Press, http://cameron.econ.ucdavis.edu/racd/racddata.html chapter 5.

Croissant Y (2011) Ecdat: Datasets for econometrics, R Package, version 0.1-6.1.

Jaggia, S. and Thosar, S. (1993). Multiple Bids as a Consequence of Target Management Resistance *Review of Quantitative Finance and Accounting* **3**, 447–457.

Examples

```
### Huang (2017) Page 371--372: Underdispersed Takeover Bids data
data(takeoverbids)
M.bids <- glm.cmp(numbids ~ leglrest + rearest + finrest + whtknght
    + bidprem + insthold + size + sizesq + regulatn, data=takeoverbids)
M.bids
summary(M.bids)
plot(M.bids) #or autoplot(M.bids)
```

`tidy.cmp` *Tidy a(n) CMP model object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'cmp'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

`x` an object class 'cmp' object, obtained from a call to `glm.cmp`

`conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.

`conf.level` The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

`exponentiate` Logical indicating whether or not to exponentiate the the coefficient estimates.

`...` other arguments passed to or from other methods (currently unused).

Value

A `tibble::tibble()` with columns:

`term` The name of the regression term.

`estimate` The estimated value of the regression term.

`std.error` The standard error of the regression term.

`statistic` The value of a test statistic to use in a hypothesis that the regression term is non-zero.

`p.value` The two-sided p-value associated with the observed statistic based on asymptotic normality.

`parameter` Only for varying dispersion models. Type of coefficient being estimated: 'mu', 'nu'

`conf.low` Lower bound on the confidence interval for the estimate.

`conf.high` Upper bound on the confidence interval for the estimate.

Examples

```
data(attendance)
M.attendance <- glm.cmp(daysabs~ gender+math+prog, data=attendance)
tidy(M.attendance)
```

update.cmp *Update and Re-fit a COM-Poisson Model*

Description

update (i.e., update.cmp) will update and (by-default) re-fit a model. It is identical to update in the stats package.

Usage

```
## S3 method for class 'cmp'
update(object, formula., formula_nu., ..., evaluate = TRUE)
```

Arguments

object	an object class 'cmp', obtained from a call to glm.cmp.
formula.	changes to the existing formula in object – see update.formula
formula_nu.	changes to the existing formula_nu in object – see update.formula for details. It also accepts NULL to not regressing on the dispersion.
...	other arguments passed to or from other methods (currently unused).
evaluate	logical; if TRUE evaluate the new call otherwise simply return the call

See Also

glm.cmp, update.formula, cmplrtest.

Examples

```
# To update the mean regression formula
data(takeoverbids)

## Fit full model
M.bids.full <- glm.cmp(numbids ~ leglrest + rearest + finrest + whtknght
    + bidprem + insthold + size + sizesq + regulatn, data=takeoverbids)
M.bids.full

## Dropping whtknght
M.bids.null <- update(M.bids.full, .~.-whtknght)
M.bids.null

## To update the dispersion regression formula
data(sitophilus)

## Fit full model
M.sit.full <- glm.cmp(formula = ninsect ~ extract, formula_nu = ~extract, data = sitophilus)
M.sit.full
```

```
## Dropping extract from the dispersion regression
M.sit.null1 <- update(M.sit.full, formula_nu =  ~.-extract)
M.sit.null1

## To not regress on the dispersion at all
M.sit.null2 <- update(M.sit.full, formula_nu = NULL)
M.sit.null2
```

vcov.cmp *Extracting the Variance-Covariance Matrix from a COM-Poisson Model Fit*

Description

Extracting the Variance-Covariance Matrix from a COM-Poisson Model Fit

Usage

```
## S3 method for class 'cmp'
vcov(object, ...)
```

Arguments

object	an object class ’cmp’ object, obtained from a call to glm.cmp
...	other arguments passed to or from other methods (currently unused).

Value

The method will return the estimated covariances between the parameter estimates of the fitted cmp model.

Examples

```
data(attendance)
M.attendance <- glm.cmp(daysabs~ gender+math+prog, data=attendance)
vcov(M.attendance)
```

Z *Calculate the Normalizing Constant for COM-Poisson distribution*

Description

A function to approximate the normalizing constant for COM-Poisson distributions via truncation. The standard COM-Poisson parametrization is being used here. Notice that the sum is hard coded to truncate at 100 so the approximation will be quite bad if the COM-Poisson has a large rate or mean.

Usage

```
Z(lambda, nu, log.z = FALSE, summax)
```

Arguments

`lambda`	rate parameter, straightly positive
`nu`	dispersion parameter, straightly positive
`log.z`	logical; if `TRUE`, normalising constant Z are returned as $log(Z)$.
`summax`	maximum number of terms to be considered in the truncated sum

Index