Package 'DGLMExtPois'

September 4, 2023

Type Package

Title Double Generalized Linear Models Extending Poisson Regression

Version 0.2.3

Maintainer Francisco Martinez <fmartin@ujaen.es>

Description Model estimation, dispersion testing and diagnosis of hyper-Poisson Saez-Castillo, A.J. and Conde-Sanchez, A. (2013) <doi:10.1016/j.csda.2012.12.009> and Conway-Maxwell-Poisson Huang, A. (2017) regression models.

Depends R (>= 3.5.0)

License GPL-2

Encoding UTF-8

LazyData true

Imports nloptr (>= 1.2.1), COMPoissonReg

RoxygenNote 7.2.3

URL https://github.com/franciscomartinezdelrio/DGLMExtPois

BugReports https://github.com/franciscomartinezdelrio/DGLMExtPois/issues

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Antonio Jose Saez-Castillo [aut],
Antonio Conde-Sanchez [aut],
Francisco Martinez [aut, cre]

Repository CRAN

Date/Publication 2023-09-04 09:50:05 UTC

R topics documented:

`AIC` *AIC and BIC for hyper-Poisson Fits*

Description

Computes the Akaike's information criterion or the Schwarz's Bayesian criterion for hyper-Poisson Fits

Usage

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

## S3 method for class 'glm_hP'
BIC(object, ...)
```

Arguments

`object`	an object of class `"glm_hP"`, typically the result of a call to `glm.hP`.
`...`	optionally more fitted model objects.
`k`	numeric, the *penalty* per parameter to be used; the default `k = 2` is the classical AIC.

Examples

```
## Fit a hyper-Poisson model
Bids$size.sq <- Bids$size ^ 2
fit <- glm.hP(formula.mu = numbids ~ leglrest + rearest + finrest +
              whtknght + bidprem + insthold + size + size.sq + regulatn,
              formula.gamma = numbids ~ 1, data = Bids)
```

```
## Obtain its AIC and BIC
AIC(fit)
BIC(fit)
```

AIC_CMP *AIC and BIC for COM-Poisson Fitted Models*

Description

Computes the Akaike's information criterion or the Schwarz's Bayesian criterion for COM-Poisson Fits

Usage

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

## S3 method for class 'glm_CMP'
BIC(object, ...)
```

Arguments

object	an object of class "glm_CMP", typically the result of a call to glm.CMP.
...	optionally more fitted model objects.
k	numeric, the *penalty* per parameter to be used; the default k = 2 is the classical AIC.

Examples

```
## Estimate a COM-Poisson model
Bids$size.sq <- Bids$size ^ 2
fit <- glm.CMP(formula.mu = numbids ~ leglrest + rearest + finrest +
              whtknght + bidprem + insthold + size + size.sq + regulatn,
              formula.nu = numbids ~ 1, data = Bids)

## Compute its AIC and BIC
AIC(fit)
BIC(fit)
```

`Bids` *Bids Received by U.S. Firms*

Description

A dataset with bids received by U.S. firms.

Usage

```
Bids
```

Format

A data frame with 126 rows and 13 variables:

docno doc no.

weeks weeks

numbids count

takeover delta(1 if taken over)

bidprem bid Premium

insthold institutional holdings

size size measured in billions

leglrest legal restructuring

rearest real restructuring

finrest financial restructuring

regulatn regulation

whtknght white knight

Source

Sanjiv Jaggia and Satish Thosar (1993) "Multiple Bids as a Consequence of Target Management Resistance", Review of Quantitative Finance and Accounting, 3(4), pp. 447-457.

A. Colin Cameron and Per Johansson (1997) "Count Data Regression Models using Series Expansions: with Applications", Journal of Applied Econometrics, 12, pp. 203-223.

confint.glm_CMP *Confidence Intervals for glm_hP Fits*

Description

Computes confidence intervals for one or more parameters in a glm_CMP object.

Usage

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

Arguments

object	a fitted object of class inheriting from "glm_CMP".
parm	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
level	the confidence level required.
...	additional argument(s) for methods.

Value

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

Examples

```
## Estimate the model
Bids$size.sq <- Bids$size ^ 2
fit <- glm.CMP(formula.mu = numbids ~ leglrest + rearest + finrest +
               whtknght + bidprem + insthold + size + size.sq + regulatn,
               formula.nu = numbids ~ 1, data = Bids)

## Compute confidence intervals
confint(fit)
```

confint.glm_hP *Confidence Intervals for glm_hP Fits*

Description

Computes confidence intervals for one or more parameters in a "glm_hP" object.

Usage

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

Arguments

object	a fitted object of class inheriting from "glm_hP".
parm	a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
level	the confidence level required.
...	additional argument(s) for methods.

Value

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

Examples

```
## Estimate the model
Bids$size.sq <- Bids$size ^ 2
fit <- glm.hP(formula.mu = numbids ~ leglrest + rearest + finrest +
         whtknght + bidprem + insthold + size + size.sq + regulatn,
              formula.gamma = numbids ~ 1, data = Bids)

## Compute confidence intervals for parameters
confint(fit)
```

CustomerProfile *Customer profile for a household supplies company*

Description

An observation corresponds to the census tracts within 10-mile radius around certain store.

Usage

```
CustomerProfile
```

Format

A data frame with 110 rows and 6 variables:

ncust number of customer of the census tracts who visit the store.

nhu number of housing units in the census tracts

aid average income in dollars

aha average housing unit in years

dnc distance to the nearest competitor in miles

ds distance to store in miles

Source

http://www.leg.ufpr.br/lib/exe/fetch.php/publications:papercompanions:ptwdataset4.txt

expected *Expected Probabilities and Frequencies for the hyper-Poisson and COM-Poisson Model*

Description

The `hP_expected` and `CMP_expected` functions calculate the probability distribution of the count response variable Y for each observation and obtain the corresponding expected frequencies. It is an informal way of assessing the fit of the hP or CMP model by comparing the predicted distribution of counts with the observed distribution.

Usage

```
hP_expected(object)

CMP_expected(object)
```

Arguments

object	a fitted object of class inheriting from "glm_hP" or "glm_CMP".

Details

The average expected probabilities are computed as

$$\bar{(}Pr)(y = k) = \frac{1}{n}\sum_{i=1}^{n}\widehat{Pr}(y_i = k|x_i)$$

The expected frequencies are obtained by multiplying by n.

Two measures are offered for summarizing the comparison between expected and observed frequencies: the sum of the absolute value of differences and the sum of the square of differences (similar to the Pearson statistic of goodness of fit).

Value

A list containing the following components:

frequencies	the expected counts for the hP or CMP fit.
observed_freq	the observed distribution.
probabilities	the expected distribution for the hP or CMP fit.
dif	sum of the absolute value of differences between frequencies and observed_freq.
chi2	sum of the square of differences between frequencies and observed_freq.

References

J. M. Hilbe (2011). Negative Binomial Regression. (2nd ed.). Cambridge University Press.

M. Scott Long and Jeremy Freese (2014). Regression Models for Categorical Dependent Variables using STATA. (3rd ed.). Stata Press.

Examples

```
## Fit a hyper-Poisson model

Bids$size.sq <- Bids$size ^ 2
hP.fit <- glm.hP(formula.mu = numbids ~ leglrest + rearest + finrest +
                 whtknght + bidprem + insthold + size + size.sq + regulatn,
                 formula.gamma = numbids ~ 1, data = Bids)

## Compute the expected probabilities and the frequencies

hP_expected(hP.fit)
## Estimate a COM-Poisson model

Bids$size.sq <- Bids$size ^ 2
CMP.fit <- glm.CMP(formula.mu = numbids ~ leglrest + rearest + finrest +
                   whtknght + bidprem + insthold + size + size.sq + regulatn,
```

```
                    formula.nu = numbids ~ 1, data = Bids)

## Compute the expected probabilities and the frequencies

CMP_expected(CMP.fit)
```

`glm.CMP` *Fit a COM-Poisson Double Generalized Linear Model*

Description

The `glm.CMP` function is used to fit a COM-Poisson double generalized linear model with a log-link for the mean (`mu`) and the dispersion parameter (`nu`).

Usage

```
glm.CMP(
  formula.mu,
  formula.nu,
  init.beta = NULL,
  init.delta = NULL,
  data,
  weights,
  subset,
  na.action,
  maxiter_series = 1000,
  tol = 0,
  offset,
  opts = NULL,
  model.mu = TRUE,
  model.nu = TRUE,
  x = FALSE,
  y = TRUE,
  z = FALSE
)
```

Arguments

`formula.mu`	regression formula linked to `log(mu)`
`formula.nu`	regression formula linked to `log(nu)`
`init.beta`	initial values for regression coefficients of `beta`.
`init.delta`	initial values for regression coefficients of `delta`.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `glm.CMP` is called.

weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
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.
maxiter_series	Maximum number of iterations to perform in the calculation of the normalizing constant.
tol	tolerance with default zero meaning to iterate until additional terms to not change the partial sum in the calculation of the normalizing constant.
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
opts	a list with options to the optimizer, nloptr, that fits the model. See, the opts parameter of nloptr for further details.
model.mu	a logical value indicating whether the *mu model frame* should be included as a component of the returned value.
model.nu	a logical value indicating whether the *nu model frame* should be included as a component of the returned value.
x	logical value indicating whether the mu model matrix used in the fitting process should be returned as a component of the returned value.
y	logical value indicating whether the response vector used in the fitting process should be returned as a component of the returned value.
z	logical value indicating whether the nu model matrix used in the fitting process should be returned as a component of the returned value.

Details

Fit a COM-Poisson double generalized linear model using as optimizer the NLOPT_LD_SLSQP algorithm of function nloptr.

Value

glm.CMP returns an object of class "glm_CMP". The function summary can be used to obtain or print a summary of the results. An object of class "glm_CMP" is a list containing at least the following components:

coefficients	a named vector of coefficients.
residuals	the residuals, that is response minus fitted values.
fitted.values	the fitted mean values.
linear.predictors	the linear fit on link scale.

call	the matched call.
offset	the offset vector used.
weights	the weights initially supplied, a vector of 1s if none were.
y	if requested (the default) the y vector used.
matrix.mu	if requested, the mu model matrix.
matrix.nu	if requested, the nu model matrix.
model.mu	if requested (the default) the mu model frame.
model.nu	if requested (the default) the nu model frame.
nloptr	an object of class "nloptr" with the result returned by the optimizer nloptr

References

Alan Huang (2017). "Mean-parametrized Conway–Maxwell–Poisson regression models for dispersed counts", Statistical Modelling, 17(6), pp. 359–380.

S. G. Johnson (2018). The nlopt nonlinear-optimization package

Examples

```
## Fit model
Bids$size.sq <- Bids$size^2
fit <- glm.CMP(formula.mu = numbids ~ leglrest + rearest + finrest +
               whtknght + bidprem + insthold + size + size.sq + regulatn,
               formula.nu = numbids ~ 1, data = Bids)

## Summary of the model
summary(fit)

## To see termination condition of the optimization process
fit$nloptr$message

## To see number of iterations of the optimization process
fit$nloptr$iterations
```

---

glm.hP *Fit a hyper-Poisson Double Generalized Linear Model*

Description

The glm.hP function is used to fit a hyper-Poisson double generalized linear model with a log-link for the mean (mu) and the dispersion parameter (gamma).

Usage

```
glm.hP(
  formula.mu,
  formula.gamma,
  init.beta = NULL,
  init.delta = NULL,
  data,
  weights,
  subset,
  na.action,
  maxiter_series = 1000,
  tol = 0,
  offset,
  opts = NULL,
  model.mu = TRUE,
  model.gamma = TRUE,
  x = FALSE,
  y = TRUE,
  z = FALSE
)
```

Arguments

formula.mu	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted.
formula.gamma	regression formula linked to log(gamma)
init.beta	initial values for regression coefficients of beta.
init.delta	initial values for regression coefficients of delta.
data	an optional data frame, list or environment (or object that can be coerced by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which glm.hP is called.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
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.
maxiter_series	Maximum number of iterations to perform in the calculation of the normalizing constant.
tol	tolerance with default zero meaning to iterate until additional terms to not change the partial sum in the calculation of the normalizing constant.

offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
opts	a list with options to the optimizer, nloptr, that fits the model. See, the opts parameter of nloptr for further details.
model.mu	a logical value indicating whether the *mu model frame* should be included as a component of the returned value.
model.gamma	a logical value indicating whether the *gamma model frame* should be included as a component of the returned value.
x	logical value indicating whether the mu model matrix used in the fitting process should be returned as a component of the returned value.
y	logical value indicating whether the response vector used in the fitting process should be returned as a component of the returned value.
z	logical value indicating whether the gamma model matrix used in the fitting process should be returned as a component of the returned value.

Details

Fit a hyper-Poisson double generalized linear model using as optimizer the NLOPT_LD_SLSQP algorithm of function nloptr.

Value

glm.hP returns an object of class "glm_hP". The function summary can be used to obtain or print a summary of the results.

The generic accessor functions coef, fitted.values and residuals can be used to extract various useful features of the value returned by glm.hP.

weights extracts a vector of weights, one for each case in the fit (after subsetting and na.action).

An object of class "glm_hP" is a list containing at least the following components:

coefficients	a named vector of coefficients.
residuals	the residuals, that is response minus fitted values.
fitted.values	the fitted mean values.
linear.predictors	the linear fit on link scale.
call	the matched call.
offset	the offset vector used.
weights	the weights initially supplied, a vector of 1s if none were.
df.residual	the residual degrees of freedom.
df.null	the residual degrees of freedom for the null model.
y	if requested (the default) the y vector used.

matrix.mu	if requested, the mu model matrix.
matrix.gamma	if requested, the gamma model matrix.
model.mu	if requested (the default) the mu model frame.
model.gamma	if requested (the default) the gamma model frame.
nloptr	an object of class "nloptr" with the result returned by the optimizer nloptr

References

Antonio J. Saez-Castillo and Antonio Conde-Sanchez (2013). "A hyper-Poisson regression model for overdispersed and underdispersed count data", Computational Statistics & Data Analysis, 61, pp. 148–157.

S. G. Johnson (2018). The nlopt nonlinear-optimization package

Examples

```
## Fit model
Bids$size.sq <- Bids$size ^ 2
fit <- glm.hP(formula.mu = numbids ~ leglrest + rearest + finrest +
              whtknght + bidprem + insthold + size + size.sq + regulatn,
              formula.gamma = numbids ~ 1, data = Bids)

## Summary of the model
summary(fit)

## To see the termination condition of the optimization process
fit$nloptr$message

## To see the number of iterations of the optimization process
fit$nloptr$iterations
```

---

hP *The hyper-Poisson Distribution*

Description

Density, distribution function and random generation for the hyper-Poisson distribution with parameters gamma and lambda.

Usage

```
dhP(x, gamma, lambda)

phP(q, gamma, lambda, lower.tail = TRUE)

rhP(n, gamma, lambda)
```

Arguments

x	vector of (non-negative integer) quantiles.
gamma	dispersion parameter. Must be strictly positive.
lambda	location parameter. Must be strictly positive.
q	vector of quantiles.
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
n	number of random values to return.

Value

dhP gives the density, phP gives the distribution function and rhP generates random deviates.

Invalid gamma or lambda will result in return value NaN, with a warning.

The length of the result is determined by n for rhP, 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 result. Only the first element of the logical arguments is used.

Examples

```
## density function for hyper-Poisson
dhP(3, 15, 2)
## distribution function for hyper-Poisson
phP(3, 15, 2)
## random generation for the hyper-Poisson
rhP(10, 15, 2)
```

---

lrtest *Likelihood Ratio Test for Nested glm_CMP and glm_hP Fits*

Description

Performs the likelihood ratio chi-squared test to compare nested models.

Usage

```
lrtest(object1, object2)
```

Arguments

object1, object2	fitted objects of classes inheriting from "glm_CMP" or "glm_hP"

Details

The test statistics is calculated as $2(llik - llik_0)$. The test statistics has a chi-squared distribution with r degrees of freedom, where r is the difference in the number of parameters between the full and null models.

Value

A list with class "lrtest" containing the following components:

statistics	the value of the statistic.
df	the degrees of freedom.
p-value	the p-value for the test.

Examples

```
Bids$size.sq <- Bids$size ^ 2

## Fit null model
fit0 <- glm.hP(formula.mu = numbids ~ leglrest + rearest + finrest +
              whtknght + bidprem + insthold + size + size.sq + regulatn,
              formula.gamma = numbids ~ 1, data = Bids)

## Fit full model
fit <- glm.hP(formula.mu = numbids ~ leglrest + rearest + finrest +
              whtknght + bidprem + insthold + size + size.sq + regulatn,
              formula.gamma = numbids ~ leglrest, data = Bids)

## Likelihood ratio test for the nested models
lrtest(fit,fit0)
```

---

plots *Plot Diagnostics for glm_hP and glm_CMP Objects*

Description

Two plots are currently available: a plot of residuals against fitted values and a Normal Q-Q plot.

Usage

```
## S3 method for class 'glm_hP'
plot(
  x,
  type = c("quantile", "pearson", "response"),
  ask = prod(graphics::par("mfcol")) < 2 && grDevices::dev.interactive(),
  ...
)

## S3 method for class 'glm_CMP'
plot(
  x,
  type = c("quantile", "pearson", "response"),
  ask = prod(graphics::par("mfcol")) < 2 && grDevices::dev.interactive(),
  ...
)
```

Arguments

x	glm_hP or glm_CMP object, typically the result of glm.hP or glm.CMP.
type	the type of residuals. The alternatives are: "quantile" (default), "pearson" and "response". Can be abbreviated.
ask	logical; if TRUE, the user is asked before each plot, see par(ask=.).
...	other parameters to be passed through to plotting functions.

Examples

```
## Fit the hyper-Poisson model
Bids$size.sq <- Bids$size ^ 2
hP.fit <- glm.hP(formula.mu = numbids ~ leglrest + rearest + finrest +
              whtknght + bidprem + insthold + size + size.sq + regulatn,
              formula.gamma = numbids ~ 1, data = Bids)
oldpar <- par(mfrow = c(1, 2))

## Plot diagnostics

plot(hP.fit)
par(oldpar)
## Fit the COM-Poisson model
Bids$size.sq <- Bids$size ^ 2
CMP.fit <- glm.CMP(formula.mu = numbids ~ leglrest + rearest + finrest +
              whtknght + bidprem + insthold + size + size.sq + regulatn,
              formula.nu = numbids ~ 1, data = Bids)
oldpar <- par(mfrow = c(1, 2))

## Plot diagnostics
plot(CMP.fit)
par(oldpar)
```

---

predict.glm_CMP *Predict Method for glm_CMP Fits*

Description

Obtains predictions from a fitted glm_CMP object.

Usage

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

Arguments

object	a fitted object of class inheriting from "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.
type	the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable.
...	further arguments passed to or from other methods.

Value

A vector with the prediction means.

Examples

```
## Fit a model
Bids$size.sq <- Bids$size ^ 2
fit <- glm.CMP(formula.mu = numbids ~ leglrest + rearest + finrest +
               whtknght + bidprem + insthold + size + size.sq + regulatn,
               formula.nu = numbids ~ 1, data = Bids)

## As the newdata parameter is not used the fitted values are obtained
predict(fit, type = "response")
```

---

predict.glm_hP *Predict Method for glm_hP Fits*

Description

Obtains predictions from a fitted "glm_hP" object.

Usage

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

Arguments

object	a fitted object of class inheriting from "glm_hP".
newdata	optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
type	the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable.
...	further arguments passed to or from other methods.

Value

A vector with the prediction means.

Examples

```
data(Bids)
## Fit a hype-Poisson model
Bids$size.sq <- Bids$size ^ 2
fit <- glm.hP(formula.mu = numbids ~ leglrest + rearest + finrest +
              whtknght + bidprem + insthold + size + size.sq + regulatn,
              formula.gamma = numbids ~ 1, data = Bids)

## As the newdata parameter is not used the fitted values are obtained
predict(fit, type = "response")
```

`residuals` *Extract and Visualize hyper-Poisson and COM-Poisson Model Residuals*

Description

residuals is a method which extracts model residuals from a "glm_hP" or "glm_CMP" object, commonly returned by glm.hP or glm.CMP. Optionally, it produces a half normal plot with a simulated envelope of the residuals.

Usage

```
## S3 method for class 'glm_hP'
residuals(
  object,
  type = c("pearson", "response", "quantile"),
  envelope = FALSE,
  rep = 19,
  title = "Simulated Envelope of Residuals",
  ...
)

## S3 method for class 'glm_CMP'
residuals(
  object,
  type = c("pearson", "response", "quantile"),
  envelope = FALSE,
  rep = 19,
  title = "Simulated Envelope of Residuals",
  ...
)
```

Arguments

object	an object of class "glm_hP" or "glm_CMP", typically the result of a call to glm.hP or glm.CMP.

type	the type of residuals which should be returned. The alternatives are: "pearson" (default), "response" and "quantile". Can be abbreviated.
envelope	a logical value indicating whether the envelope should be computed.
rep	number of replications for envelope construction. Default is 19, that is the smallest 95 percent band that can be built.
title	a string indicating the main title of the envelope.
...	further arguments passed to or from other methods.

Details

The response residuals ($r_i = y_i - \mu_i$), Pearson residuals ($r_i^P = r_i/\sigma_i$) or randomized quantile residuals are computed. The randomized quantile residuals are obtained computing the cumulative probabilities that the fitted model being less than y and less or equal than y. A random value from a uniform distribution between both probabilities is generated and the value of the residual is the standard normal variate with the same cumulative probability. Four replications of the quantile residuals are recommended because of the random component (see Dunn and Smyth, 1996 for more details).

The functions plot.glm_hP and plot.glm_CMP generate a residuals against fitted values plot and a Normal Q-Q plot.

The Normal Q-Q plot may show an unsatisfactory pattern of the Pearson residuals of a fitted model: then we are led to think that the model is incorrectly specified. The half normal plot with simulated envelope indicates that, under the distribution of the response variable, the model is fine when only a few points fall off the envelope.

Value

Residual values.

References

Peter K. Dunn and Gordon K. Smyth (1996). "Randomized quantile residuals". Journal of Computational and Graphical Statistics, 5(3), pp. 236-244.

A. C. Atkinson (1981). "Two graphical displays for outlying and influential observations in regression". Biometrika, 68(1), pp. 13–20.

See Also

plots

Examples

```
## Estimate a hyper-Poisson model
Bids$size.sq <- Bids$size ^ 2
hP.fit <- glm.hP(formula.mu = numbids ~ leglrest + rearest + finrest +
              whtknght + bidprem + insthold + size + size.sq + regulatn,
              formula.gamma = numbids ~ 1, data = Bids)

## Compute residuals
```

```
r <- residuals(hP.fit)
## Estimate a COM-Poisson model

Bids$size.sq <- Bids$size ^ 2
CMP.fit <- glm.CMP(formula.mu = numbids ~ leglrest + rearest + finrest +
              whtknght + bidprem + insthold + size + size.sq + regulatn,
              formula.nu = numbids ~ 1, data = Bids)

## Compute its residuals

r <- residuals(CMP.fit)
```

---

summary.glm_CMP *Summarizing COM-Poisson Fits*

Description

These functions are all methods for class "glm_CMP" or summary.glm_CMP objects.

Usage

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

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

Arguments

object	an object of class "glm_CMP", usually, a result of a call to glm.CMP.
...	further arguments passed to or from other methods.
x	an object of class "summary.glm_CMP", usually, a result of a call to summary.glm_CMP.
digits	the number of significant digits to use when printing.
signif.stars	logical. If TRUE, 'significance stars' are printed for each coefficient.

Examples

```
## Fit a COM-Poisson model
Bids$size.sq <- Bids$size^2
fit <- glm.CMP(formula.mu = numbids ~ leglrest + rearest + finrest +
               whtknght + bidprem + insthold + size + size.sq + regulatn,
```

```
                formula.nu = numbids ~ 1, data = Bids)

## Obtain a summary of the fitted model

summary(fit)
```

---

summary.glm_hP *Summarizing hyper-Poisson Fits*

Description

These functions are all methods for class "glm_hP" or summary.glm_hP objects.

Usage

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

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

Arguments

object	an object of class "glm_hP", usually, a result of a call to glm.hP.
...	further arguments passed to or from other methods.
x	an object of class "summary.glm_hP", usually, a result of a call to summary.glm_hP.
digits	the number of significant digits to use when printing.
signif.stars	logical. If TRUE, 'significance stars' are printed for each coefficient.

Examples

```
## Fit a hyper-Poisson model

Bids$size.sq <- Bids$size ^ 2
fit <- glm.hP(formula.mu = numbids ~ leglrest + rearest + finrest +
              whtknght + bidprem + insthold + size + size.sq + regulatn,
              formula.gamma = numbids ~ 1, data = Bids)

## Obtain a summary of the fitted model

summary(fit)
```

Index