Package 'COMPoissonReg'

January 20, 2025

Type Package

Title Conway-Maxwell Poisson (COM-Poisson) Regression

Version 0.8.1

Author Kimberly Sellers <kfs7@georgetown.edu>
Thomas Lotze <thomas.lotze@thomaslotze.com>
Andrew Raim <andrew.raim@gmail.com>

Maintainer Andrew Raim <andrew.raim@gmail.com>

URL https://github.com/lotze/COMPoissonReg

Description Fit Conway-Maxwell Poisson (COM-Poisson or CMP) regression models to count data (Sellers & Shmueli, 2010) <doi:10.1214/09-AOAS306>. The package provides functions for model estimation, dispersion testing, and diagnostics. Zero-inflated CMP regression (Sellers & Raim, 2016) <doi:10.1016/j.csda.2016.01.007> is also supported.

LazyLoad yes

Depends stats, Rcpp, numDeriv

LinkingTo Rcpp

RoxygenNote 7.2.3

Encoding UTF-8

Suggests testthat (>= 3.0.0), knitr, rmarkdown, ggplot2

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-11-29 13:30:05 UTC

Contents

`COMPoissonReg-package` *Estimate parameters for COM-Poisson regression*

Description

This package offers the ability to compute the parameter estimates for a COM-Poisson or zero-inflated (ZI) COM-Poisson regression and associated standard errors. This package also provides a hypothesis test for determining statistically significant data dispersion, and other model diagnostics.

Details

This package offers the ability to compute COM-Poisson parameter estimates and associated standard errors for a regular regression model or a zero-inflated regression model (via the `glm.cmp` function).

Further, the user can perform a hypothesis test to determine the statistically significant need for using COM-Poisson regression to model the data. The test addresses the matter of statistically significant dispersion.

The main order of functions for COM-Poisson regression is as follows:

1. Compute Poisson estimates (using `glm` for Poisson regression or `pscl` for ZIP regression).

2. Use Poisson estimates as starting values to determine COM-Poisson estimates (using `glm.cmp`).
3. Compute associated standard errors (using `sdev` function).

From here, there are many ways to proceed, so order is irrelevant:

- Perform a hypothesis test to assess for statistically significant dispersion (using `equitest` or `parametric.bootstrap`).
- Compute leverage (using leverage) and deviance (using deviance).
- Predict the outcome for new examples, using predict.

The package also supports fitting of the zero-inflated COM-Poisson model (ZICMP). Most of the tools available for COM-Poisson are also available for ZICMP.

As of version 0.5.0 of this package, a hybrid method is used to compute the normalizing constant $z(\lambda, \nu)$ for the COM-Poisson density. A closed-form approximation (Shmueli et al, 2005; Gillispie & Green, 2015) to the exact sum is used if the given λ is sufficiently large and ν is sufficiently small. Otherwise, an exact summation is used, except that the number of terms is truncated to meet a given accuracy. Previous versions of the package used simple truncation (defaulting to 100 terms), but this was found to be inaccurate in some settings.

See the package vignette for a more comprehensive guide on package use and explanations of the computations.

Author(s)

Kimberly Sellers, Thomas Lotze, Andrew M. Raim

References

Steven B. Gillispie & Christopher G. Green (2015) Approximating the Conway-Maxwell-Poisson distribution normalization constant, Statistics, 49:5, 1062-1073.

Kimberly F. Sellers & Galit Shmueli (2010). A Flexible Regression Model for Count Data. Annals of Applied Statistics, 4(2), 943-961.

Kimberly F. Sellers and Andrew M. Raim (2016). A Flexible Zero-Inflated Model to Address Data Dispersion. Computational Statistics and Data Analysis, 99, 68-80.

Galit Shmueli, Thomas P. Minka, Joseph B. Kadane, Sharad Borle, and Peter Boatwright (2005). A useful distribution for fitting discrete data: revival of the Conway-Maxwell-Poisson distribution. Journal of Royal Statistical Society C, 54, 127-142.

`CMP Distribution` *COM-Poisson Distribution*

Description

Functions for the COM-Poisson distribution.

Usage

```
dcmp(x, lambda, nu, log = FALSE, control = NULL)

rcmp(n, lambda, nu, control = NULL)

pcmp(x, lambda, nu, control = NULL)

qcmp(q, lambda, nu, log.p = FALSE, control = NULL)

ecmp(lambda, nu, control = NULL)

vcmp(lambda, nu, control = NULL)

ncmp(lambda, nu, log = FALSE, control = NULL)

tcmp(lambda, nu, control = NULL)
```

Arguments

`x`	vector of quantiles.
`lambda`	rate parameter.
`nu`	dispersion parameter.
`log`	logical; if TRUE, probabilities are returned on log-scale.
`control`	a `COMPoissonReg.control` object from `get.control` or `NULL` to use global default.
`n`	number of observations.
`q`	vector of probabilities.
`log.p`	logical; if TRUE, probabilities `p` are given as $\log(p)$.

Value

dcmp density,

pcmp cumulative probability,

qcmp quantiles,

rcmp generate random variates,

ecmp expected value,

vcmp variance,

ncmp value of the normalizing constant, and

tcmp upper value used to compute the normalizing constant under truncation method.

Author(s)

Kimberly Sellers

References

Kimberly F. Sellers & Galit Shmueli (2010). A Flexible Regression Model for Count Data. Annals of Applied Statistics, 4(2), 943-961.

`COMPoissonReg-options` *Package options*

Description

Global options used by the COMPoissonReg package.

Arguments

`COMPoissonReg.control`
: A default control data structure for the package. See the helper function get.control for a description of contents.

Details

- `getOption("COMPoissonReg.control")`

`couple` *Couple dataset*

Description

A dataset investigating the impact of education level and level of anxious attachment on unwanted pursuit behaviors in the context of couple separation.

Usage

```
data(couple)
```

Format

UPB number of unwanted pursuit behavior perpetrations.

EDUCATION 1 if at least bachelor's degree; 0 otherwise.

ANXIETY continuous measure of anxious attachment.

References

Loeys, T., Moerkerke, B., DeSmet, O., Buysse, A., 2012. The analysis of zero-inflated count data: Beyond zero-inflated Poisson regression. British J. Math. Statist. Psych. 65 (1), 163-180.

`equitest` *Equidispersion Test*

Description

Likelihood ratio test for equidispersion

Usage

```
equitest(object, ...)
```

Arguments

`object`	a model object
`...`	other parameters which might be required by the model

Details

A generic function for the likelihood ratio test for equidispersion using the output of a fitted mode. The function invokes particular methods which depend on the class of the first argument.

Value

Returns the test statistic and p-value determined from a χ^2 distribution with d_2 degrees of freedom.

Author(s)

Thomas Lotze

`freight` *Freight dataset*

Description

A set of data on airfreight breakage (breakage of ampules filled with some biological substance are shipped in cartons).

Usage

```
data(freight)
```

Format

broken number of ampules found broken upon arrival.

transfers number of times carton was transferred from one aircraft to another.

References

Kutner MH, Nachtsheim CJ, Neter J (2003). Applied Linear Regression Models, Fourth Edition. McGraw-Hill.

`get.control` *Construct a control object to pass additional arguments to a number of functions in the package.*

Description

Construct a control object to pass additional arguments to a number of functions in the package.

Usage

```
get.control(
  ymax = 1e+06,
  optim.method = "L-BFGS-B",
  optim.control = list(maxit = 150),
  hybrid.tol = 0.01,
  truncate.tol = 1e-06
)
```

Arguments

`ymax`	Truncate counts to maximum value of `y`.
`optim.method`	Optimization method for maximum likelihood. See the `method` argument in optim.
`optim.control`	`control` argument for optim.
`hybrid.tol`	Tolerance to decide when to use truncation method versus approximation method to compute quantities based on the normalizing constant. See details.
`truncate.tol`	Tolerance for truncation method. See details.

Details

A hybrid method is used throughout the package to compute the CMP normalizing constant and related quantities. When $\lambda^{-1/\nu}$ is smaller than `hybrid.tol`, an asymptotic approximation is used; otherwise, infinite series are truncated to finite summations. More information is given in the `COMPoissonReg` vignette.

The element `ymax` protects against very long computations. Users should beware when increasing this significantly beyond the default, as it may result in a session which needs to be terminated.

Value

List of controls.

`get.fixed` *Construct an object that specifies which indices of coefficients should remain fixed in maximum likelihood computation.*

Description

Construct an object that specifies which indices of coefficients should remain fixed in maximum likelihood computation.

Usage

```
get.fixed(beta = integer(0), gamma = integer(0), zeta = integer(0))
```

Arguments

`beta`	Vector of indices of `beta` to keep fixed.
`gamma`	Vector of indices of `gamma` to keep fixed.
`zeta`	Vector of indices of `zeta` to keep fixed.

Details

Arguments are expected to be vectors of integers. These are interpreted as the indices to keep fixed during optimization. For example, `beta = c(1L, 1L, 2L)` indicates that the first and second elements of `beta` should remain fixed. Note that duplicate indices are ignored. The default value is the empty vector `integer(0)`, which requests that no elements of the given coefficient vector should be fixed.

Value

List of vectors indicating fixed indices.

`get.init` *Construct initial values for coefficients.*

Description

Construct initial values for coefficients.

Usage

```
get.init(beta = NULL, gamma = NULL, zeta = NULL)
```

Arguments

`beta`	Vector for `beta`.
`gamma`	Vector for `gamma`.
`zeta`	Vector for `zeta`.

Details

The default value `NULL` is interpreted as an empty vector, so that the given component is absent from the model.

Value

List of initial value terms.

`get.init.zero` *Construct initial values for coefficients with zeros.*

Description

Construct initial values for coefficients with zeros.

Usage

```
get.init.zero(d1 = 0, d2 = 0, d3 = 0)
```

Arguments

`d1`	Dimension of `beta`.
`d2`	Dimension of `gamma`.
`d3`	Dimension of `zeta`.

Value

List of initial value terms containing all zeros.

get.modelmatrix — *Construct model matrices and offsets for CMP/ZICMP regression*

Description

Construct model matrices and offsets for CMP/ZICMP regression

Usage

```
get.modelmatrix(X = NULL, S = NULL, W = NULL, offset = NULL)
```

Arguments

`X`	An `X` matrix to use with `beta`.
`S`	An `S` matrix to use with `gamma`.
`W`	A `W` matrix to use with `zeta`.
`offset`	An offset object. See helper function get.offset.

Value

List of model matrix terms.

get.offset — *Construct values for offsets.*

Description

Construct values for offsets.

Usage

```
get.offset(x = NULL, s = NULL, w = NULL)
```

Arguments

`x`	Vector of offsets to go with `X` matrix.
`s`	Vector of offsets to go with `S` matrix.
`w`	Vector of offsets to go with `W` matrix.

Details

The default value `NULL` is interpreted as a vector of zeros. At least one component must be non-NULL so that the dimension can be determined.

Value

List of offset terms.

`get.offset.zero` *Construct zero values for offsets.*

Description

Construct zero values for offsets.

Usage

```
get.offset.zero(n)
```

Arguments

n	Number of observations.

Value

List of offset terms containing all zeros.

`glm.cmp` *COM-Poisson and Zero-Inflated COM-Poisson Regression*

Description

Fit COM-Poisson regression using maximum likelihood estimation. Zero-Inflated COM-Poisson can be fit by specifying a regression for the overdispersion parameter.

Usage

```
glm.cmp(
  formula.lambda,
  formula.nu = ~1,
  formula.p = NULL,
  data = NULL,
  init = NULL,
  fixed = NULL,
  control = NULL,
  ...
)
```

Arguments

`formula.lambda`	regression formula linked to `log(lambda)`. The response should be specified here.
`formula.nu`	regression formula linked to `log(nu)`. The default, is taken to be only an intercept.
`formula.p`	regression formula linked to `logit(p)`. If NULL (the default), zero-inflation term is excluded from the model.
`data`	An optional data.frame with variables to be used with regression formulas. Variables not found here are read from the envionment.
`init`	A data structure that specifies initial values. See the helper function get.init.
`fixed`	A data structure that specifies which coefficients should remain fixed in the maximum likelihood procedure. See the helper function get.fixed.
`control`	A control data structure. See the helper function get.control. If `NULL`, a global default will be used.
`...`	other arguments, such as `subset` and `na.action`.

Details

The COM-Poisson regression model is

$$y_i \sim \mathrm{CMP}(\lambda_\mathrm{i}, \nu_\mathrm{i}), \quad \log \lambda_\mathrm{i} = \mathrm{x_i}^\top \beta, \quad \log \nu_\mathrm{i} = \mathrm{s_i}^\top \gamma.$$

The Zero-Inflated COM-Poisson regression model assumes that y_i is 0 with probability p_i or y_i^* with probability $1 - p_i$, where

$$y_i^* \sim \mathrm{CMP}(\lambda_\mathrm{i}, \nu_\mathrm{i}), \quad \log \lambda_\mathrm{i} = \mathrm{x_i}^\top \beta, \quad \log \nu_\mathrm{i} = \mathrm{s_i}^\top \gamma, \quad \mathrm{logit}\, \mathrm{p_i} = \mathrm{w_i}^\top \zeta.$$

Value

`glm.cmp` produces an object of either class `cmpfit` or `zicmpfit`, depending on whether zero-inflation is used in the model. From this object, coefficients and other information can be extracted.

Author(s)

Kimberly Sellers, Thomas Lotze, Andrew Raim

References

Kimberly F. Sellers & Galit Shmueli (2010). A Flexible Regression Model for Count Data. Annals of Applied Statistics, 4(2), 943-961.

Kimberly F. Sellers and Andrew M. Raim (2016). A Flexible Zero-Inflated Model to Address Data Dispersion. Computational Statistics and Data Analysis, 99, 68-80.

glm.cmp, CMP support *Supporting Functions for COM-Poisson Regression*

Description

Supporting Functions for COM-Poisson Regression

Usage

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

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

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

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

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

## S3 method for class 'cmpfit'
coef(object, type = c("vector", "list"), ...)

## S3 method for class 'cmpfit'
nu(object, ...)

## S3 method for class 'cmpfit'
sdev(object, type = c("vector", "list"), ...)

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

## S3 method for class 'cmpfit'
equitest(object, ...)

## S3 method for class 'cmpfit'
leverage(object, ...)

## S3 method for class 'cmpfit'
deviance(object, ...)

## S3 method for class 'cmpfit'
residuals(object, type = c("raw", "quantile"), ...)
```

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

## S3 method for class 'cmpfit'
parametric.bootstrap(object, reps = 1000, report.period = reps + 1, ...)
```

Arguments

`object`	object of type `cmp`.
`...`	other arguments, such as `subset` and `na.action`.
`x`	object of type `cmp`.
`k`	Penalty per parameter to be used in AIC calculation.
`type`	Specifies quantity to be computed. See details.
`newdata`	New covariates to be used for prediction.
`reps`	Number of bootstrap repetitions.
`report.period`	Report progress every `report.period` iterations.

Details

The function `residuals` returns raw residuals when `type = "raw"` and quantile residuals when `type = "quantile"`.

The function `predict` returns expected values of the outcomes, eveluated at the computed estimates, when `type = "response"`. When `type = "link"`, a `data.frame` is instead returned with columns corresponding to estimates of `lambda` and `nu`.

The function `coef` returns a vector of coefficient estimates in the form `c(beta, gamma)` when `type = "vector"`. When `type = "list"`, the estimates are returned as a list with named elements `beta` and `gamma`.

The `type` argument behaves the same for the `sdev` function as it does for `coef`.

---

glm.cmp, ZICMP support

Supporting Functions for ZICMP Regression

Description

Supporting Functions for ZICMP Regression

Usage

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

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

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

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

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

## S3 method for class 'zicmpfit'
coef(object, type = c("vector", "list"), ...)

## S3 method for class 'zicmpfit'
nu(object, ...)

## S3 method for class 'zicmpfit'
sdev(object, type = c("vector", "list"), ...)

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

## S3 method for class 'zicmpfit'
equitest(object, ...)

## S3 method for class 'zicmpfit'
deviance(object, ...)

## S3 method for class 'zicmpfit'
residuals(object, type = c("raw", "quantile"), ...)

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

## S3 method for class 'zicmpfit'
parametric.bootstrap(object, reps = 1000, report.period = reps + 1, ...)
```

Arguments

`object`	object of type `zicmp`.
`...`	other arguments, such as `subset` and `na.action`.
`x`	object of type `zicmp`.

k	Penalty per parameter to be used in AIC calculation.
type	Specifies quantity to be computed. See details.
newdata	New covariates to be used for prediction.
reps	Number of bootstrap repetitions.
report.period	Report progress every report.period iterations.

Details

The function residuals returns raw residuals when type = "raw" and quantile residuals when type = "quantile".

The function predict returns expected values of the outcomes, eveluated at the computed estimates, when type = "response". When type = "link", a data.frame is instead returned with columns corresponding to estimates of lambda, nu, and p.

The function coef returns a vector of coefficient estimates in the form c(beta, gamma, zeta) when type = "vector". When type = "list", the estimates are returned as a list with named elements beta and gamma, and zeta.

The type argument behaves the same for the sdev function as it does for coef.

---

glm.cmp-raw *Raw Interface to COM-Poisson and Zero-Inflated COM-Poisson Regression*

Description

Fit COM-Poisson and Zero-Inflated COM-Poisson regression using a "raw" interface which bypasses the formula-driven interface of glm.cmp.

Usage

```
glm.cmp.raw(y, X, S, offset = NULL, init = NULL, fixed = NULL, control = NULL)

glm.zicmp.raw(
  y,
  X,
  S,
  W,
  offset = NULL,
  init = NULL,
  fixed = NULL,
  control = NULL
)
```

Arguments

`y`	A vector of counts which represent the response .
`X`	Design matrix for the ‘lambda‘ regression.
`S`	Design matrix for the ‘nu‘ regression.
`offset`	A data structure that specifies offsets. See the helper function get.offset.
`init`	A data structure that specifies initial values. See the helper function get.init.
`fixed`	A data structure that specifies which coefficients should remain fixed in the maximum likelihood procedure. See the helper function get.fixed.
`control`	A control data structure. See the helper function get.control.
`W`	Design matrix for the ‘p‘ regression.

Value

See the glm.cmp.

`leverage` *Leverage*

Description

A generic function for the leverage of points used in various model fitting functions. The function invokes particular methods which depend on the class of the first argument.

Usage

```
leverage(object, ...)
```

Arguments

`object`	a model object
`...`	other parameters which might be required by the model

Details

See the documentation of the particular methods for details.

Value

The form of the value returned depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method.

Author(s)

Thomas Lotze

nu *Estimate for dispersion parameter*

Description

(Deprecated) A generic function for the dispersion parameter estimate from the results of various model fitting functions. The function invokes particular methods which depend on the class of the first argument.

Usage

```
nu(object, ...)
```

Arguments

`object`	a model object
`...`	other parameters which might be required by the model

Details

See the documentation of the particular methods for details.

Value

The form of the value returned depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method.

See Also

predict

parametric.bootstrap *Parametric Bootstrap*

Description

A generic function for the parametric bootstrap from the results of various model fitting functions. The function invokes particular methods which depend on the class of the first argument.

Usage

```
parametric.bootstrap(object, reps = 1000, report.period = reps + 1, ...)
```

Arguments

`object`	a model object
`reps`	Number of bootstrap repetitions.
`report.period`	Report progress every `report.period` iterations.
`...`	other parameters which might be required by the model

Details

See the documentation of the particular methods for details.

Value

The form of the value returned depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method.

Author(s)

Thomas Lotze

`sdev` *Standard deviation*

Description

A generic function for standard deviation estimates from the results of various model fitting functions. The function invokes particular methods which depend on the class of the first argument.

Usage

```
sdev(object, ...)
```

Arguments

`object`	a model object
`...`	other parameters which might be required by the model

Details

See the documentation of the particular methods for details.

Value

The form of the value returned depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method.

Author(s)

Thomas Lotze

`ZICMP Distribution` *ZICMP Distribution*

Description

Computes the density, cumulative probability, quantiles, and random draws for the zero-inflated COM-Poisson distribution.

Usage

```
dzicmp(x, lambda, nu, p, log = FALSE, control = NULL)

rzicmp(n, lambda, nu, p, control = NULL)

pzicmp(x, lambda, nu, p, control = NULL)

qzicmp(q, lambda, nu, p, log.p = FALSE, control = NULL)

ezicmp(lambda, nu, p, control = NULL)

vzicmp(lambda, nu, p, control = NULL)
```

Arguments

`x`	vector of quantiles.
`lambda`	rate parameter.
`nu`	dispersion parameter.
`p`	zero-inflation probability parameter.
`log`	logical; if TRUE, probabilities are returned on log-scale.
`control`	a `COMPoissonReg.control` object from `get.control` or `NULL` to use global default.
`n`	number of observations.
`q`	vector of probabilities.
`log.p`	logical; if TRUE, probabilities p are given as $\log(p)$.

Value

dzicmp density,

pzicmp cumulative probability,

qzicmp quantiles,

rzicmp generate random variates,

ezicmp expected value. and

vzicmp variance.

Author(s)

Kimberly Sellers, Andrew Raim

References

Kimberly F. Sellers and Andrew M. Raim (2016). A Flexible Zero-Inflated Model to Address Data Dispersion. Computational Statistics and Data Analysis, 99, 68-80.

`ZIP Distribution` *COM-Poisson Distribution*

Description

Functions for the COM-Poisson distribution.

Usage

```
dzip(x, lambda, p, log = FALSE)

rzip(n, lambda, p)

pzip(x, lambda, p)

qzip(q, lambda, p, log.p = FALSE)

ezip(lambda, p)

vzip(lambda, p)
```

Arguments

`x`	vector of quantiles.
`lambda`	rate parameter.
`p`	zero-inflation probability parameter.
`log`	logical; if TRUE, probabilities are returned on log-scale.
`n`	number of observations.
`q`	vector of probabilities.
`log.p`	logical; if TRUE, probabilities `p` are given as $\log(p)$.

Value

dzip density,
pzip cumulative probability,
qzip quantiles,
rzip generate random variates,
ezip expected value,
vzip variance,

Author(s)

Kimberly Sellers

Index