Package ‘COMPoissonReg’

December 2, 2022

Type Package

Title Conway-Maxwell Poisson (COM-Poisson) Regression

Version 0.8.0

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


License GPL-2 | GPL-3

LazyLoad yes

Depends stats, Rcpp, numDeriv

LinkingTo Rcpp

RoxygenNote 7.2.2

Encoding UTF-8

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

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-12-02 10:40:17 UTC
**R topics documented:**

- COMPoissonReg-package .................................................. 2
- CMP Distribution ............................................................. 3
- COMPoissonReg-options .................................................... 5
- couple ................................................................. 5
- equitest .............................................................. 6
- freight ............................................................ 6
- get.control ............................................................... 7
- get.fixed .............................................................. 8
- get.init ............................................................... 8
- get.init.zero ............................................................ 9
- get.modelmatrix ............................................................ 10
- get.offset ............................................................. 10
- get.offset.zero ........................................................... 11
- glm.cmp .............................................................. 11
- glm.cmp, CMP support ......................................................... 13
- glm.cmp, ZICMP support ..................................................... 14
- glm.cmp-raw ............................................................ 16
- leverage .............................................................. 17
- nu ................................................................. 18
- parametric.bootstrap ....................................................... 18
- sdev ............................................................... 19
- ZICMP Distribution ....................................................... 20
- ZIP Distribution .......................................................... 21

**Index** ........................................................................ 23

---

**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 \( \lambda \) is sufficiently large and \( \nu \) 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


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

---

**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**
equitest \hspace{2cm} Equidispersion Test

Description

Likelihood ratio test for equidispersion

Usage

equitest(object, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>a model object</td>
</tr>
<tr>
<td>...</td>
<td>other parameters which might be required by the model</td>
</tr>
</tbody>
</table>

Details

A generic function for the likelihood ratio test for equidispersion using the output of a fitted model. 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 $\chi^2$ distribution with $d_2$ degrees of freedom.

Author(s)

Thomas Lotze

freight \hspace{2cm} 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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>broken</td>
<td>number of ampules found broken upon arrival.</td>
</tr>
<tr>
<td>transfers</td>
<td>number of times carton was transferred from one aircraft to another.</td>
</tr>
</tbody>
</table>
get.control

References

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

```r
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

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

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,  
    ...  
)
```
glm.cmp

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 environment.

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

$$y_i \sim CMP(\lambda_i, \nu_i), \quad \log \lambda_i = x_i^\top \beta, \quad \log \nu_i = s_i^\top \gamma.$$  

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

$$y_i^* \sim CMP(\lambda_i, \nu_i), \quad \log \lambda_i = x_i^\top \beta, \quad \log \nu_i = s_i^\top \gamma, \quad \logit p_i = w_i^\top \zeta.$$  

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


Description

Supporting Functions for COM-Poisson Regression

Usage

```r
## 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, evaluated 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`.

---

**Description**

Supporting Functions for ZICMP Regression
Usage

```r
## 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`. 
Penalty per parameter to be used in AIC calculation.

Specifies quantity to be computed. See details.

New covariates to be used for prediction.

Number of bootstrap repetitions.

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, evaluated 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**

```r
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)
```

leverage

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`.

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

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, ...)
sdev

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

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

Usage

\begin{itemize}
  \item \texttt{dzicmp(x, lambda, nu, p, log = FALSE, control = NULL)}
  \item \texttt{rzicmp(n, lambda, nu, p, control = NULL)}
  \item \texttt{pzicmp(x, lambda, nu, p, control = NULL)}
  \item \texttt{qzicmp(q, lambda, nu, p, log.p = FALSE, control = NULL)}
  \item \texttt{ezicmp(lambda, nu, p, control = NULL)}
  \item \texttt{vzicmp(lambda, nu, p, control = NULL)}
\end{itemize}

Arguments

- \texttt{x} \hspace{1cm} vector of quantiles.
- \texttt{lambda} \hspace{1cm} rate parameter.
- \texttt{nu} \hspace{1cm} dispersion parameter.
- \texttt{p} \hspace{1cm} zero-inflation probability parameter.
- \texttt{log} \hspace{1cm} logical; if TRUE, probabilities are returned on log-scale.
- \texttt{control} \hspace{1cm} a \texttt{COMPoissonReg.control} object from \texttt{get.control} or \texttt{NULL} to use global default.
- \texttt{n} \hspace{1cm} number of observations.
- \texttt{q} \hspace{1cm} vector of probabilities.
- \texttt{log.p} \hspace{1cm} logical; if TRUE, probabilities \texttt{p} are given as \texttt{log(p)}.

Value

- \texttt{dzicmp} density,
- \texttt{pzicmp} cumulative probability,
- \texttt{qzicmp} quantiles,
- \texttt{rzicmp} generate random variates,
- \texttt{ezicmp} expected value. and
- \texttt{vzicmp} variance.
ZIP Distribution

Author(s)
Kimberly Sellers, Andrew Raim

References

ZIP Distribution  COM-Poisson Distribution

Description
Functions for the COM-Poisson distribution.

Usage
\begin{verbatim}
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)
\end{verbatim}

Arguments
\begin{itemize}
  \item \texttt{x} vector of quantiles.
  \item \texttt{lambda} rate parameter.
  \item \texttt{p} zero-inflation probability parameter.
  \item \texttt{log} logical; if TRUE, probabilities are returned on log-scale.
  \item \texttt{n} number of observations.
  \item \texttt{q} vector of probabilities.
  \item \texttt{log.p} logical; if TRUE, probabilities \texttt{p} are given as \texttt{log(p)}.
\end{itemize}

Value
\begin{itemize}
  \item \texttt{dzip} density,
  \item \texttt{pzip} cumulative probability,
  \item \texttt{qzip} quantiles,
  \item \texttt{rzip} generate random variates,
  \item \texttt{ezip} expected value,
  \item \texttt{vzip} variance,
\end{itemize}
Author(s)
   Kimberly Sellers
## Index

* **COM-Poisson**  
  CMP Distribution, 3

* **Poisson**  
  ZIP Distribution, 21

* **Zero-Inflated**  
  ZIP Distribution, 21

* **datasets**  
  couple, 5  
  freight, 6

* **distribution**  
  CMP Distribution, 3  
  ZIP Distribution, 21

AIC.cmpfit (glm.cmp, CMP support), 13  
AIC.zicmpfit (glm.cmp, ZICMP support), 14

BIC.cmpfit (glm.cmp, CMP support), 13  
BIC.zicmpfit (glm.cmp, ZICMP support), 14

CMP Distribution, 3  
coef.cmpfit (glm.cmp, CMP support), 13  
coef.zicmpfit (glm.cmp, ZICMP support), 14

COMPoissonReg-options, 5  
COMPoissonReg-package, 2  
couple, 5

dcmp (CMP Distribution), 3  
deviance.cmpfit (glm.cmp, CMP support), 13  
deviance.zicmpfit (glm.cmp, ZICMP support), 14

dzicmp (ZICMP Distribution), 20  
dzip (ZIP Distribution), 21

ecmp (CMP Distribution), 3  
equitest, 6  
equitest.cmpfit (glm.cmp, CMP support), 13  
equitest.zicmpfit (glm.cmp, ZICMP support), 14

ezicmp (ZICMP Distribution), 20  
ezip (ZIP Distribution), 21

freight, 6

g.get.control, 5, 7, 12, 17  
g.get.fixed, 8, 12, 17  
g.get.init, 8, 12, 17  
g.get.init.zero, 9  
g.get.modelmatrix, 10  
g.get.offset, 10, 10, 17  
g.get.offset.zero, 11  
glm.cmp, 11, 17  
glm.cmp, CMP support, 13  
glm.cmp, ZICMP support, 14  
glm.cmp.raw, 16  
glm.cmp.raw (glm.cmp.raw), 16

leverage, 17  
leverage.cmpfit (glm.cmp, CMP support), 13  
logLik.cmpfit (glm.cmp, CMP support), 13  
logLik.zicmpfit (glm.cmp, ZICMP support), 14

ncmp (CMP Distribution), 3  
nu, 18  
nu.cmpfit (glm.cmp, CMP support), 13  
nu.zicmpfit (glm.cmp, ZICMP support), 14

optim, 7

parametric.bootstrap, 18  
parametric.bootstrap.cmpfit (glm.cmp, CMP support), 13  
parametric.bootstrap.zicmpfit (glm.cmp, ZICMP support), 14

pcmp (CMP Distribution), 3
predict.cmpfit(glm.cmp, CMP support), 13
predict.zicmpfit(glm.cmp, ZICMP support), 14
print.cmpfit(glm.cmp, CMP support), 13
print.zicmpfit(glm.cmp, ZICMP support), 14
pzicmp (ZICMP Distribution), 20
pzip (ZIP Distribution), 21
qcmp (CMP Distribution), 3
qzicmp (ZICMP Distribution), 20
qzip (ZIP Distribution), 21
rcmp (CMP Distribution), 3
residuals.cmpfit(glm.cmp, CMP support), 13
residuals.zicmpfit(glm.cmp, ZICMP support), 14
rzicmp (ZICMP Distribution), 20
rzip (ZIP Distribution), 21
sdev, 19
sdev.cmpfit(glm.cmp, CMP support), 13
sdev.zicmpfit(glm.cmp, ZICMP support), 14
summary.cmpfit(glm.cmp, CMP support), 13
summary.zicmpfit(glm.cmp, ZICMP support), 14
tcmp (CMP Distribution), 3
vcmp (CMP Distribution), 3
vcov.cmpfit(glm.cmp, CMP support), 13
vcov.zicmpfit(glm.cmp, ZICMP support), 14
vzicmp (ZICMP Distribution), 20
vzip (ZIP Distribution), 21
ZICMP Distribution, 20
ZIP Distribution, 21