Package ‘COMPoissonReg’

November 30, 2019

Type Package

Title Conway-Maxwell Poisson (COM-Poisson) Regression

Version 0.7.0

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

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.

License GPL-2 | GPL-3

LazyLoad yes

Depends stats, Rcpp

LinkingTo Rcpp

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-11-30 06:50:16 UTC

R topics documented:

`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.

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.

Examples

```
## load freight data
data(freight)

# Fit standard Poisson model
glm.out = glm(broken ~ transfers, data=freight,
  family=poisson, na.action=na.exclude)
print(glm.out)

# Fit COM-Poisson model (with intercept-only regression linked to the
# dispersion parameter)
cmp.out = glm.cmp(broken ~ transfers, data=freight)
print(cmp.out)
coef(cmp.out)
nu(cmp.out)[1]

# Compute associated standard errors
sdev(cmp.out)

# Get the full covariance matrix for the estimates
vcov(cmp.out)

# Likelihood ratio test for dispersion parameter
# Test for H_0: dispersion equal to 1 vs. H_1: not equal to 1
# (i.e. Poisson vs. COM-Poisson regression models)
lrt = equitest(cmp.out)

# Compute constant COM-Poisson leverage
lev = leverage(cmp.out)
```

```
## Not run:
# Compute constant COM-Poisson deviances
dev = deviance(cmp.out)

## End(Not run)

# Compute fitted values
y.hat = predict(cmp.out, newdata=freight)

# Compute residual values
res = residuals(cmp.out)
print(summary(res))

# Compute MSE
mean(res^2)

# Compute predictions on new data
new.data = data.frame(transfers=(0:10))
y.hat = predict(cmp.out, newdata=new.data)
plot(0:10, y.hat, type="l",
  xlab="number of transfers", ylab="predicted number broken")

## Not run:
# Compute parametric bootstrap results and use them to generate
# 0.95 confidence intervals for parameters.
cmp.boot = parametric.bootstrap(cmp.out, reps=1000)
print(apply(cmp.boot, 2, quantile, c(0.025,0.975)))

## End(Not run)

## Not run:
## load couple data
data(couple)

# Fit standard Poisson model
glm.out = glm(UPB ~ EDUCATION + ANXIETY, data=couple, family=poisson)
print(glm.out)

# Fit ZICMP model
zicmp.out = glm.cmp(UPB ~ EDUCATION + ANXIETY,
  formula.nu = ~ 1,
  formula.p = ~ EDUCATION + ANXIETY,
  data=couple)
print(zicmp.out)

# Compute standard errors for estimates of coefficients
sdev(zicmp.out)

# Get the full covariance matrix for the estimates
vcov(zicmp.out)

# Likelihood ratio test for equidispersion (H0: nu = 1 vs H1: not)
equitest(zicmp.out)
```

```
# Compute fitted values
y.hat = predict(zicmp.out)

# Compute residuals
res.raw = residuals(zicmp.out, type = "raw")
res.quan = residuals(zicmp.out, type = "quantile")
print(summary(res.raw))
print(summary(res.quan))

# Compute predictions on new data
new.data = data.frame(EDUCATION = round(1:20 / 20), ANXIETY = seq(-3,3, length.out = 20))
y.hat.new = predict(zicmp.out, newdata=new.data)
print(y.hat.new)

# Compute parametric bootstrap results and use them to generate
# 0.95 confidence intervals for parameters.
zicmp.boot = parametric.bootstrap(zicmp.out, reps=1000)
print(apply(zicmp.boot, 2, quantile, c(0.025,0.975)))

# A CMP example with offset terms.
cmp.out = glm.cmp(broken ~ transfers + offset(transfers), data=freight)
print(cmp.out)
coef(cmp.out)
nu(cmp.out)[1]

# A ZICMP example with offset terms.
zicmp.out = glm.cmp(UPB ~ EDUCATION + ANXIETY + offset(ANXIETY),
    formula.nu = ~ offset(ANXIETY),
    formula.p = ~ EDUCATION + ANXIETY + offset(ANXIETY),
    data=couple)
print(zicmp.out)

## End(Not run)
```

`CMP Distribution` *COM-Poisson Distribution*

Description

Functions for the COM-Poisson distribution.

Usage

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

rcmp(n, lambda, nu)

pcmp(x, lambda, nu)
```

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

Arguments

`x`	vector of quantiles.
`lambda`	rate parameter.
`nu`	dispersion 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

- `dcmp` gives the density,
- `pcmp` gives the cumulative probability,
- `qcmp` gives the quantile function, and
- `rcmp` generates random values.

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.optim.method`
: Optim method to use when computing maximum likelihood estimates.

`COMPoissonReg.optim.control`
: A list to be passed to `control` when calling `optim`. `fnscale` will be ignored if specified.

`COMPoissonReg.grad.eps`
: Distance to be used when finite differences are taken.

`COMPoissonReg.hess.eps`
: Distance to be used when finite second differences are taken.

`COMPoissonReg.ymax`
: Maximum count value to be considered. Larger values are truncated.

Details

- `options(COMPoissonReg.optim.method = 'L-BFGS-B')`
- `options(COMPoissonReg.optim.control = list(maxit = 150))`
- `options(COMPoissonReg.grad.eps = 1e-5)`
- `options(COMPoissonReg.hess.eps = 1e-2)`
- `options(COMPoissonReg.ymax = 1e6)`

`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, ...)

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

## S3 method for class 'zicmp'
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 the $\chi.1^2$ distribution.

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.

`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 = NULL, formula.p = NULL,
  beta.init = NULL, gamma.init = NULL, zeta.init = NULL, ...)
```

Arguments

`formula.lambda`	regression formula linked to `log(lambda)`.
`formula.nu`	regression formula linked to `log(nu)`. If NULL (the default), is taken to be intercept only.
`formula.p`	regression formula linked to `logit(p)`. If NULL (the default), zero-inflation term is excluded from the model.
`beta.init`	initial values for regression coefficients of `lambda`.
`gamma.init`	initial values for regression coefficients of `nu`.
`zeta.init`	initial values for regression coefficients of `p`.
`...`	other model parameters, such as data.

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

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

Value

`glm.cmp` produces an object of either class 'cmp' or 'zicmp', 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 'cmp'
summary(object, ...)

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

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

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

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

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

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

## S3 method for class 'cmp'
```

```
sdev(object, ...)

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

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

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

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

## S3 method for class 'cmp'
predict(object, newdata = NULL, ...)

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

Arguments

`object`	object of type `cmp`.
`...`	other model parameters, such as data.
`x`	object of type `cmp`.
`k`	Penalty per parameter to be used in AIC calculation.
`type`	Type of residual to be computed.
`newdata`	New covariates to be used for prediction.
`reps`	Number of bootstrap repetitions.
`report.period`	Report progress every `report.period` iterations.

`glm.cmp, ZICMP support`

Supporting Functions for ZICMP Regression

Description

Supporting Functions for ZICMP Regression

Usage

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

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

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

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

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

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

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

## S3 method for class 'zicmp'
sdev(object, ...)

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

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

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

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

## S3 method for class 'zicmp'
predict(object, newdata = NULL, ...)

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

Arguments

`object`	object of type `zicmp`.
`...`	other model parameters, such as data.

x	object of type zicmp.
k	Penalty per parameter to be used in AIC calculation.
type	Type of residual to be computed.
newdata	New covariates to be used for prediction.
reps	Number of bootstrap repetitions.
report.period	Report progress every report.period iterations.

---

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

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.

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)

rzicmp(n, lambda, nu, p)

pzicmp(x, lambda, nu, p)

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

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.
n	number of observations.
q	vector of probabilities.
log.p	logical; if TRUE, probabilities p are given as $\log(p)$.

Value

dzicmp gives the density,

pzicmp gives the cumulative probability,

qzicmp gives the quantile value, and

rzicmp generates random numbers.

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.

Index