

# Package ‘exactLoglinTest’

January 2, 2012

**Version** 1.3.7

**Title** Monte Carlo Exact Tests for Log-linear models

**Author** Brian Caffo <bcaffo@jhsph.edu>

**Maintainer** Brian Caffo <bcaffo@jhsph.edu>

**Depends** R (>= 1.5)

**Suggests** stats

**Description** Monte Carlo and MCMC goodness of fit tests for log-linear models

**License** GPL

**URL** <http://www.biostat.jhsph.edu/~bcaffo>

**Repository** CRAN

**Date/Publication** 2011-06-30 17:12:03

## R topics documented:

alligator.dat . . . . .	2
czech.dat . . . . .	2
gof . . . . .	3
hyper . . . . .	4
mcexact . . . . .	4
mcexact.internals . . . . .	6
pathologist.dat . . . . .	8
print.mcexact . . . . .	8
residence.dat . . . . .	9
simulateConditional . . . . .	10
titanic.dat . . . . .	11
update.bab . . . . .	12
update.cab . . . . .	13

<b>Index</b>	<b>15</b>
--------------	-----------

---

alligator.dat	<i>Data frame of alligator food choice by lake, gender and size.</i>
---------------	--

---

**Description**

A 5x4x2x2 table of alligators food choice, lake, gender and size stored in a data frame.

**Usage**

```
data(alligator.dat)
```

**Format**

allgiator.dat is a data frame

**References**

Agresti (1990). *Categorical Data Analysis*. Wiley-Interscience. Page 308.

---

czech.dat	<i>Czech auto workers data</i>
-----------	--------------------------------

---

**Description**

Czech auto workers data

**Usage**

```
data(czech.dat)
```

**Format**

A data frame with 64 observations on the following 7 variables.

**y** a numeric vector

**A** a factor with levels no yes

**B** a factor with levels no yes

**C** a factor with levels no yes

**D** a factor with levels small large

**E** a factor with levels small large

**F** a factor with levels neg pos

**Source**

Edwards and Havranek (1985)

**Examples**

```
data(czech.dat)
```

---

**gof***Goodness-of-fit function for Poisson log-linear models*

---

**Description**

A goodness-of-fit function for Poisson log-linear models as required by `mcexact`.

**Usage**

```
gof(y = NULL, mu = NULL, rowlabels = FALSE)
```

**Arguments**

<code>y</code>	counts
<code>mu</code>	mean
<code>rowlabels</code>	labels of returned goodness-of-fit statistics

**Details**

This function is a useful shell for writing alternative goodness-of-fit statistics for `mcexact`.

**Value**

If `rowlabels = TRUE`, then `gof` returns only the labels of the goodness of fit statistics. Otherwise it returns the actual values as a vector.

**Author(s)**

Brian S. Caffo

**See Also**

[mcexact](#)

**Examples**

```
#data(residence)
#get fitted values
#mu <- glm(residence$y ~ residence$x, family = poisson)$fit
#gof(residence$y, mu)
#gof(rowlabels = TRUE)
```

hyper

*Generalized hypergeometric distribution on the log scale*

---

**Description**

Returns the log hypergeometric conditional probability of a vector of Poisson counts up to an additive constant of proportionality.

**Usage**

```
hyper(y)
```

**Arguments**

y                      Counts

**Value**

A real-valued scalar.

**Author(s)**

Brian S. Caffo

**See Also**

[mcexact](#)

---

mcexact

*Computes Monte Carlo exact P-values for general log-linear models.*

---

**Description**

This function computes Monte Carlo estimates of conditional P-values for goodness of fit tests for general log-linear models.

**Usage**

```
mcexact(formula,  
        data,  
        stat = gof,  
        dens = hyper,  
        nosim = 10 ^ 3,  
        method = "bab",  
        savechain = FALSE,  
        tdf = 3,
```

```

      maxiter = nosim,
      p = NULL,
      batchsize = NULL)

build.mcx.obj(formula,
              data,
              stat = gof,
              dens = hyper,
              nosim = 10 ^ 3,
              method = "bab",
              savechain = FALSE,
              tdf = 3,
              maxiter = nosim,
              p = NULL,
              batchsize = NULL)

```

### Arguments

formula	Null model formula specified as in <code>glm</code>
data	Data frame
stat	The test statistic, a function of the form <code>function(y, mu.hat)</code> where <code>y</code> is the observed and <code>mu.hat</code> are the fitted values. Current default <code>gof</code> is a bivariate function of the deviance and the Pearson chi-squared.
dens	The target density on the log scale up to a constant of proportionality. A function of the form <code>function(y)</code> . Current default is (proportional to) the log of the generalized hypergeometric density.
nosim	Desired number of simulations.
method	Possibly two values, the importance sampling method of Booth and Butler, <code>method = "bab"</code> or the MCMC approach of Caffo and Booth <code>method = "cab"</code> .
savechain	If TRUE saves the values of the chain.
tdf	A tuning parameter
maxiter	For <code>method = "bab"</code> number of iterations is different from the number of simulations. <code>maxiter</code> is a bound on the total number of iterations.
p	A tuning parameter for <code>method = "cab"</code> .
batchsize	Required batchsizes for <code>method = "cab"</code> .

### Value

Returns a list of class either "bab" or "cab" depending on `method`. The list contains all of the inputs plus all required information to resume the simulation. Generic functions `print` and `summary` format the output while `update` can be used to resume simulations. `mcexact` is the front end while `build.mcx.obj` simply builds the basic object that `mcexact` applies to. `simulate.conditional` generates a matrix of simulated tables.

### Author(s)

Brian Caffo

## References

Booth and Butler (1999), "An importance sampling algorithm for exact conditional tests in log-linear models", *Biometrika* 86: 321-332.

Caffo and Booth (2001). "A Markov Chain Monte Carlo Algorithm for Approximating Exact Conditional Probabilities", *The Journal of Computational and Graphical Statistics* 10: 730-45.

<http://www.biostat.jhsph.edu/~bcaffo/downloads.htm>

## See Also

[fisher.test](#)

## Examples

```
#library(mcexact)
set.seed(1)

#importance sampling
data(residence.dat)
mcx <- mcexact(y ~ res.1985 + res.1980 + factor(sym.pair), data = residence.dat)
summary(mcx)

#mcmc
data(pathologist.dat)
mcx <- mcexact(y ~ factor(A) + factor(B) + I(A * B),
               data = pathologist.dat,
               method = "cab",
               p = .5,
               nosim = 10 ^ 4,
               batchsize = 100)
summary(mcx)
```

---

mcexact.internals      *Internal functions for mcexact*

---

## Description

Internal functions used for mcexact.

## Usage

```
rounded.tprob(y, m, s, df)
errorcheck(y,
           x,
           stat,
           dens,
           nosim,
           method,
           savechain,
```

```
tdf,  
maxiter,  
p,  
batchsize)
```

**Arguments**

y	counts
m	means
s	variances
df	degrees of freedom
x	model matrix
stat	function to be checked
dens	function to be checked
nosim	number of simulations to be checked
method	method to be checked
savechain	savechain flag to be checked
tdf	t degrees of freedom to be checked
maxiter	maximum number of iterations to be checked
p	prop of table entries left fixed to be checked
batchsize	batchsize to be checked

**Value**

rounded.tprob returns rounded student's t probabilities for integers y with means m and variances s. errorcheck is a function containing most of the error checking that mcexact performs.

**Author(s)**

Brian S. Caffo

**See Also**

[mcexact](#)

---

pathologist.dat	<i>Cross-classification of 118 tumor grades by two pathologists</i>
-----------------	---

---

**Description**

Data frame of a 5x5 table of pathologists ratings of 118 tumors. Each pathologist, A and B, rated the tumors on a scale of 1 to 5. y represents the counts of each combination of ratings.

**Usage**

```
data(pathologist.dat)
```

**Format**

A list containing y-the counts and x-the design matrix for the uniform association model.

**References**

Agresti (1990). *Categorical Data Analysis*. Wiley-Interscience. Page 368.

**See Also**

Page 263 of *Categorical Data Analysis* for a description of the uniform association model.

---

print.mcexact	<i>Print utilities for mcexact</i>
---------------	------------------------------------

---

**Description**

Generic print methods for objects of class cab and bab outputted from mcexact.

**Usage**

```
## S3 method for class 'bab'
print(x,...)
## S3 method for class 'bab'
summary(object,...)
```

**Arguments**

x	An object of class bab or cab
object	An object of class bab or cab
...	Unused, retained for generic method consistency

**Value**

`print.bab` and `print.cab` prints the P-value estimates obtained from `mcexact`. `summary.bab` and `summary.cab` prints extra information and returns a matrix of the P-value estimates.

**Author(s)**

Brian S. Caffo

**See Also**

[mcexact](#)

**Examples**

```
#data(residence)
#resid.mcx <- mcexact(residence$y ~ residence$x, nosim = 10 ^ 2, maxiter = 10 ^ 4)
#resid.mcx #calls print.bab
#print(resid.mcx) #calls print.bab
#summary(resid.mcx) #calls summary.bab
```

---

residence.dat

*Residence in 1980 by residence in 1985.*

---

**Description**

A data frame of a 4x4 cross-classification of residences in 1980 by residence in 1985. `residence$y` gives the counts, `res.1985` and `res.1980` give the 1980 and 1985 residences, `sym.pair` is used to fit the quasi-symmetry model.

**Usage**

```
data(residence.dat)
```

**Format**

A list containing y-the counts and x-the design matrix for the quasi-symmetry model.

**References**

Agresti (1990). *Categorical Data Analysis*. Wiley-Interscience. Page 357.

**See Also**

Page 354 of *Categorical Data Analysis* for a description of the quasi-symmetry model.

---

simulateConditional *Simulates from the conditional distribution of a log-linear model*

---

### Description

Simulates from the conditional distribution of log-linear models given the sufficient statistics.

### Usage

```
simulateConditional(formula,
                    data,
                    dens = hyper,
                    nosim = 10^3,
                    method = "bab",
                    tdf = 3,
                    maxiter = nosim,
                    p = NULL,
                    y.start = NULL)
simtable.bab(args, nosim = NULL, maxiter = NULL)
simtable.cab(args, nosim = NULL, p = NULL, y.start = NULL)
```

### Arguments

formula	A formula for the log-linear model
data	A data frame
dens	The target density on the log scale up to a constant of proportionality. A function of the form $f(y)$ . Current default is (proportional to) the log of the generalized hypergeometric density.
nosim	Desired number of simulations.
method	Possibly two values, the importance sampling method of Booth and Butler, <code>method = "bab"</code> or the MCMC approach of Caffo and Booth <code>method = "cab"</code> .
tdf	A tuning parameter
maxiter	For <code>method = "bab"</code> number of iterations is different from the number of simulations. <code>maxiter</code> is a bound on the total number of iterations.
p	A tuning parameter for <code>method = "cab"</code> .
y.start	An optional starting value when <code>method = "cab"</code>
args	An object of class "bab" or "cab"

### Value

A matrix where each simulated table is a row.

### Author(s)

Brian Caffo

**See Also**[fisher.test](#)**Examples**

```
data(czech.dat)
chain2 <- simulateConditional(y ~ (A + B + C + D + E + F) ^ 2,
                             data = czech.dat,
                             method = "cab",
                             nosim = 10 ^ 3,
                             p = .4,
                             dens = function(y) 0)
```

---

titanic.dat

*Titanic Survival Data*

---

**Description**

A data frame of counts of titanic passengers classified by class, age, sex and survival

**Usage**

```
data(titanic.dat)
```

**Format**

A data frame with 32 observations on the following 5 variables.

**y** a numeric vector

**class** a numeric vector

**age** a numeric vector

**sex** a numeric vector

**surv** a numeric vector

**Source**

Cytel web site <http://www.cytel.com>

**Examples**

```
data(titanic.dat)
```

---

`update.bab`*Update method for objects of class bab*

---

### Description

An update method for objects created by `mcexact` when `method = 'bab'`.

### Usage

```
## S3 method for class 'bab'
update(object, ...)
bab(args, nosim = NULL, maxiter = NULL, savechain = FALSE)
```

### Arguments

<code>object</code>	Output from <code>mcexact</code>
<code>...</code>	Alternative arguments for the update
<code>args</code>	Output from <code>mcexact</code>
<code>nosim</code>	The desired number of simulations to be performed in the update
<code>maxiter</code>	The maximum number of iterations allowed.
<code>savechain</code>	Saves the chain of goodness-of-fit statistics and their importance weights

### Details

The method `update.bab` calls the function `bab`, which is the engine for `mcexact` when `method = 'bab'`.

### Value

A list of the form outputted from `mcexact`

### Author(s)

Brian S. Caffo

### See Also

[mcexact](#)

### Examples

```
data(residence.dat)
mcx <- mcexact(y ~ res.1985 + res.1980 + factor(sym.pair), data = residence.dat)
summary(mcx)
mcx <- update(mcx, nosim = 10 ^ 4, maxiter = 10 ^ 6)
summary(mcx)
```

---

`update.cab`*Update method for objects of class cab*

---

**Description**

An update method for objects created by `mcexact` when `method = 'cab'`.

**Usage**

```
## S3 method for class 'cab'
update(object, ...)
cab(args,
     nosim = NULL,
     batchsize = NULL,
     savechain = FALSE,
     p = NULL,
     flush = FALSE)
```

**Arguments**

<code>object</code>	Output from <code>mcexact</code>
<code>...</code>	Alternative arguments for the update
<code>args</code>	Output from <code>mcexact</code>
<code>nosim</code>	The number of simulations to be performed in the update
<code>batchsize</code>	A new batchsize
<code>savechain</code>	Saves the chain of goodness-of-fit statistics
<code>p</code>	An updated proportion of simulated tables left fixed.
<code>flush</code>	Should the previous information be discarded? <code>flush</code> should be set to true if the batchsize is changed.

**Details**

The method `update.cab` calls the function `cab`, which is the engine for `mcexact` when `method = 'cab'`.

**Value**

A list of the form outputted from `mcexact`

**Author(s)**

Brian S. Caffo

**See Also**

[mcexact](#)

**Examples**

```
data(residence.dat)
mcx <- mcexact(y ~ res.1985 + res.1980 + factor(sym.pair),
              data = residence.dat,
              method = "cab",
              p = .5,
              batchsize = 100)
summary(mcx)
mcx <- update(mcx, nosim = 10 ^ 4)
summary(mcx)
```

# Index

## \*Topic **datasets**

- alligator.dat, 2
- czech.dat, 2
- pathologist.dat, 8
- residence.dat, 9
- titanic.dat, 11

## \*Topic **htest**

- gof, 3
- hyper, 4
- mcexact, 4
- mcexact.internals, 6
- print.mcexact, 8
- simulateConditional, 10
- update.bab, 12
- update.cab, 13

alligator.dat, 2

bab (update.bab), 12  
build.mcx.obj (mcexact), 4

cab (update.cab), 13  
czech.dat, 2

errorcheck (mcexact.internals), 6

fisher.test, 6, 11

gof, 3

hyper, 4

mcexact, 3, 4, 4, 7, 9, 12, 13  
mcexact.internals, 6

pathologist.dat, 8  
print.bab (print.mcexact), 8  
print.babSummary (print.mcexact), 8  
print.cab (print.mcexact), 8  
print.cabSummary (print.mcexact), 8  
print.mcexact, 8

residence.dat, 9

rounded.tprob (mcexact.internals), 6

simtable.bab (simulateConditional), 10  
simtable.cab (simulateConditional), 10  
simulateConditional, 10

summary.bab (print.mcexact), 8  
summary.cab (print.mcexact), 8

titanic.dat, 11

update.bab, 12  
update.cab, 13