Package: ltable (via r-universe)

September 10, 2024

Type Package

Title Easy to Make (Lazy) Tables

Version 2.0.4

Date 2024-07-09

Author Ocheredko Oleksandr

Description Constructs tables of counts and proportions out of data sets with possibility to insert tables to Excel, Word, HTML, and PDF documents. Transforms tables to data suitable for modelling. Features Gibbs sampling based log-linear (NB2) and power analyses (original by Oleksandr Ocheredko <doi:10.35566/isdsa2019c5>) for tabulated data.

Maintainer Ocheredko Oleksandr <Ocheredko@yahoo.com>

License GPL (>= 2)

Depends R (>= 4.3.0), methods, clipr

Imports graphics, stats, MASS, nloptr

Suggests knitr, datasets

Collate MCLogLin.R MCPower.R Plot.R PrintPower.R table_f.R tableToData.R Renewal.R

Repository CRAN

Date/Publication 2024-07-11 22:40:02 UTC

NeedsCompilation no

Contents

`ltable-package` *ltable*

Description

Constructs tables of counts and proportions out of data sets. Performs log-linear and power analyzes of tabulated data

Details

Gibbs sampling based log-linear analysis features some advantages against *glm {stats}*, first of all due to fixing overdispersion by NB2 posterior marginal distribution of counts that insures distinctly less biased covariance estimates, pivot issue for implemented power analysis. In some instances hypothesis testing of higher order effects disagrees with that of *glm {stats}* on account of larger NB2 model based errors estimates. Another though related enhancement is distinct better fit assessed by sum of squared differences between observed and expected counts. Results of power analysis backed up with MCMC BUGS delivered approach (reference 2).

Note

You can:

1. construct tables with data set fields of factor, character, logical, and numeric classes;
2. insert tables into Excel and Word documents using clipboard, into LaTeX, HTML, Markdown and reStructuredText documents by the knitr::kable agency;
3. perform Gibbs sampling based log-linear analysis;
4. perform power analysis of selected effect.

Author(s)

Ocheredko Oleksandr <Ocheredko@yahoo.com>

References

Ocheredko O.M. MCMC Bootstrap Based Approach to Power and Sample Size Evaluation. https://www.amazon.com/gp/product/1946728039/

Examples

```
require(ltable)
data(sdata, package="ltable")
table_f(sdata, "a")
table_f(sdata, "a", MV=TRUE, extended=TRUE)
table_f(sdata, "a,b,c")
knitr::kable(table_f(sdata, "a,b,c,d", type=2, digits=3))
table_f(sdata, "b,c,a,d", MV=TRUE, extended=TRUE, cb=TRUE)
```

`BCdata` *Breast cancer data to model risk*

Description

Breast cancer rates in Iceland by year of birth (11 cohorts from 1840-1849 to 1940-1949) and by age (13 groups from 20-24 to 80-84 years). Analysed by Breslow and Clayton (1993). Data is used also in BUGS Example "Ice: non-parametric smoothing in an age-cohort model". Data supplied with file *offsetdata.rda*.

Usage

```
data(BCdata)
```

Format

Description of data frames:

Fields:

1.	age	Age group: 1-13	numeric
2.	year	Birth cohort: 1-11	numeric
3.	cases	Breast cancer counts	numeric
4.	pyr	Person-years of risk	numeric

References

BUGS. Examples Volume 2. http://eio.usc.es/pub/mjginzo/descargas/leyenda/Documents/R/win-library/2.12/BRugs/OpenBUGS/Examples/Ice.html

Examples

```
require(ltable)
data(BCdata, package="ltable")
res1<-MCLogLin(cases~age*year, offset=pyr, data=BCdata, draw=1500, burnin=500)
```

`MCLogLin` *Function* MCLogLin

Description

Performs log-linear analyses for constructed tabulated data based on Gibbs sampler with NB2 posterior marginal distribution for counts

Usage

```
MCLogLin(formula, data, offset, contrasts=NULL, XLB=-100, XUB=100, a=0.1, b=0.1,
DIC=FALSE, pcov=FALSE, draw=10000, burnin=3000 )
```

Arguments

`formula`	a symbolic description of the model to be fit.
`data`	name of the data set; object of *data.frame* class
`offset`	variable in the data set to be used as offset.
`contrasts`	serves to choose types of contrasts to study effects of factors, the same with *glm{stats}*), orthogonal polynomials by default
`XLB`	the vector of smallest possible values of regression effects *betas*; can be number if pertains to all *betas*.
`XUB`	the vector of largest possible values of regression effects *betas*; can be number if pertains to all *betas*.
`a`	the value of shape parameter of gamma distributed inverce dispersion parameter (*phi*), i.e., *phi~Ga(a,b)*, so that mean(phi)=a/b and var(phi)=a/b^2
`b`	the value of rate (1/scale) parameter of gamma distributed inverce dispersion parameter (*phi*), i.e., *phi~Ga(a,b)*, so that mean(phi)=a/b and var(phi)=a/b^2
`DIC`	requests print of deviance information criteria and its components
`pcov`	requests print of covariance and correlation matricies of the model parameters
`draw`	indicates requested number of samples
`burnin`	indicates requested number of initial samples to discard

Details

- Performs log-linear modelling with supplied data by using Gibbs sampler.
- Printing output includes standard table of parameters estimates, goodness of fit indicators, analysis of residuals. On the prompt it prints the deviance information criteria with its components as well as covariance and correlation matricies of the model parameters. By using parameter (*offset*) one can model risks and relative risks instead of counts.

Value

returns a matrix with columns of chains of sampled values of model parameters (expected counts, regression coefficients, inverce dispersion parameter) to be studied by MCMC facilitating packages (e.g., coda, mcmc, mcmcplot, etc.)

Note

Function provides better conditioned variance matrix estimates against function *stats::glm*, which is particular important for high order effects and power analysis. Particularly suggestive is to check the model fit first. Jacobian reciprocal condition number near zero indicates solution instability. If *chisq/n* » 1, the error estimates obtained from the covariance matrix will be too small and should be multiplied by square root of *chisq/dof*. Poor fit will result from the use of an inappropriate model and jeopardizes the validity of power analysis.
It's recommended to keep difference between pars *draw* and *burnin* at least 3000.

Author(s)

Ocheredko Oleksandr <Ocheredko@yahoo.com>

See Also

glm

Examples

```
require(ltable)
data(tdata, package="ltable")
## For better illustration You should increase draw and burnin pars
res1<-MCLogLin(Counts~smoker +contraceptive +tromb +
contraceptive*tromb, data=tdata, draw=1500, burnin=500)


data(iris)
iriscut<-with(iris, data.frame(PL=cut(Petal.Length,3),
                               PW=cut(Petal.Width,3)))
irist<-table_f(iriscut,"PL,PW")
irisd<-tableToData(irist, ordered="PL,PW")
res2<-ltable::MCLogLin(Counts~PW+PL+PW*PL, DIC=TRUE, data=irisd,
draw=1500, burnin=500)
```

MCPower *Function* MCPower

Description

Performs power analyses for constructed tabulated data based on based on Gibbs sampler with NB2 posterior marginal distribution for counts

Usage

```
MCPower(formula, data, offset, contrasts=NULL, XLB=-100, XUB=100, a=0.1, b=0.1,
scale_min=1, scale_max=5, effect, p_alpha=0.05, draw=10000, burnin=3000)
```

Arguments

formula	a symbolic description of the model to be fit.
data	name of the data set; object of *data.frame* class
offset	variable in the data set to be used as offset.
contrasts	serves to choose types of contrasts to study effects of factors, same with *glm{stats}*), orthogonal polynomials by default
XLB	the vector of smallest possible values of regression effects *betas*; can be number if pertains to all *betas*.
XUB	the vector of largest possible values of regression effects *betas*; can be number if pertains to all *betas*.
a	the value of shape parameter of gamma distributed inverce dispersion parameter (*phi*), i.e., *phi~Ga(a,b)*, so that mean(phi)=a/b and var(phi)=a/b^2
b	the value of rate (1/scale) parameter of gamma distributed inverce dispersion parameter (*phi*), i.e., *phi~Ga(a,b)*, so that mean(phi)=a/b and var(phi)=a/b^2
scale_min	the smallest number of sample size scale range, 1 signifies the given data sample size (observed total counts).
scale_max	the max number of sample size considered in power analysis. 5 by default means 5 times augmented observed counts
effect	quoted effect tested by hypothesis; it should be one from the model formula, of second or higher order, introduced by * delimiter, i.e., "y*x", "y1*y2*x1*x2", etc.
p_alpha	serves to signify Z to check simulated z-scores against in power analysis, 0.05 by default
draw	indicates requested number of samples
burnin	indicates requested number of initial samples to discard

Details

- Performs power analysis in a given range of sample sizes (scale_min - scale_max).
- Creates object of S4 class *PowerClass* with accessing methods

Value

returns object of S4 class *PowerClass*

Note

The inspected sample size range defined by scale_min - scale_max automatically is divided into 11 consecutive values investigated by function. Given the results one can change sample size range, for example to scrutinize some particular interval to ensure power and p-value.
Function provides better conditioned variance matrix estimates against function *stats::glm* by the auspicity of NB2 dispersion parameter, coping with overdispersion in counts distribution, which is particular important for high order effects and power analysis. Particularly suggestive is to check the model fit first. Jacobian reciprocal condition number near zero indicates solution instability. If *chisq/n* » 1, the error estimates obtained from the covariance matrix will be too small and should be multiplied by square root of *chisq/dof*. Poor fit will result from the use of an inappropriate model and jeopardizes the validity of power analysis.
The drawback is failure to tackle singularity of order 5 or higher of Hessian matrix. Code returns error "Sorry, can't proceed with singular Hessian matrix." On such rare occasions please use ltable v.2.0.1 available for Unix (MacOS) machines. Function PowerPoisson performs log-linear and power analyses based on Levenberg-Marquardt algorithm which is distribution-free (so, Poisson in name of function is misleading). The only inconvenience is that GSL: GNU Scientific Library has to be installed first.
See-saw dynamic of either power or test curves is caused by Jacobian singularity, that indicates solution instability.
Flat profiles given low test or power values are indicative for insignificance of tested effect.
Flat profiles with z-values above 2 or power values that exceed 0.8 are indicative for significance of tested effect. On such occasions decrease both scale parameters to inspect smaller samples.

Author(s)

Ocheredko Oleksandr <Ocheredko@yahoo.com>

See Also

glm

Examples

```
require(ltable)
data(tdata, package="ltable")
## For better illustration You should increase draw and burnin pars
pres1<-MCPower(Counts~smoker +contraceptive +tromb +contraceptive*tromb,
scale_min=0.5, scale_max=1.5, effect="contraceptive*tromb", data=tdata,
draw=1000, burnin=300)
print(pres1, "model")
print(pres1)
plot(pres1, stencil=3)
```

plot,powerClass-method
Method for Function plot

Description

Method for function plot with
signature(x = "powerClass")

Usage

```
## S4 method for signature 'powerClass'
plot(x, stencil, ...)
```

Arguments

x	the name of *powerClass* object.
stencil	an optional arg containing 4 choices of print: missing(default), 1, 2, 3. See details.
...	not used

Details

The second argument *stencil* controls "what and how" to plot. *stencil=missing* (default) plots stand-alone images of z-score and power distributions along the range of sample sizes (see *print-method* for details on the range).
stencil=1 chooses z-score distributions to plot in stand-along fashion.
stencil=2 chooses power distributions to plot in stand-along fashion.
stencil=3 controls to plot z-score and power distributions paired alongside.
Also, Q0.05, Q01, Q0.5 (median) quantiles are graphed in lines.

Methods

signature(x = "powerClass") Method for function plot for object of S4 class *powerClass*.

Examples

```
require(ltable)
data(tdata, package="ltable")
## For better illustration You should increase draw and burnin pars
pres<-MCPower(Counts~smoker +contraceptive +tromb +
contraceptive*tromb, scale_max=1.5, effect="contraceptive*tromb",
data=tdata, draw=1000, burnin=300)
plot(pres)
plot(pres, stencil=3)
```

powerClass-class *Class* "powerClass"

Description

Objects of S4 class *powerClass* are exceptionally suitable for suggested approach to power analysis. Class serves a purpose of container of odds and ends of magnitude of information both on log-linear estimates and fit statistics as well as on the power analysis results, i.e., alpha and beta errors distributions across 11 sample sizes. Class also supported by getters and setters, text and graphic outputs.

Objects from the Class

Objects can be created by calls of the form `new("powerClass", ...)`.

Slots

`varnames`: Vector of mode `"character"` lists names of columns in design matrix.

`effectsname`: Vector of mode `"character"` lists names of columns in design matrix that constitute effect under study. Latter is given by arg *effect* in function *PowerPoisson.*

`cal`: Object of class `"call"` saves the function call.

`Ntotal`: Vector of mode `"numeric"`. Contains sample size of the data, scale_min, scale_max values

`estim`: Object of class `"list"` List of 11 lists of log-linear parameters estimates and model fit statistics across 11 sample sizes

`power1`: Object of class `"list"`. Contains lists for each column (contrast) of design matrix involved in effect under study. Each such list containes numeric vectors of values of simulated reg.coefficients, z-scores, power. Slot `power1` keeps the data pertaining to smallest sample size

`power2`: power2:power11 slots envelop the same structured information across consecutive sample sizes 2:11(largest).

`power3`: -//-

`power4`: -//-

`power5`: -//-

`power6`: -//-

`power7`: -//-

`power8`: -//-

`power9`: -//-

`power10`: -//-

`power11`: -//-

Methods

[`signature(x = "powerClass", i = "character", j = "integer", drop = "logical")`: getter, see Method for Function `[`

[<- `signature(x = "powerClass", i = "character", j = "integer", value)`: setter, see Method for Function `[<-`

plot `signature(x = "powerClass")`: plots images of z-score and power distributions along the range of sample sizes

print `signature(x = "powerClass")`: prints estimated log-linear model parameters and fit statistics as well as results of power analysis along the range of sample sizes

Author(s)

Ocheredko Oleksandr <Ocheredko@yahoo.com>

References

Ocheredko O.M. MCMC Bootstrap Based Approach to Power and Sample Size Evaluation. https://www.amazon.com/gp/product/1946728039/

Examples

```
require(ltable)
showClass("powerClass")
new("powerClass")
data(tdata, package="ltable")
## For better illustration You should increase draw and burnin pars
pres<-MCPower(Counts~smoker +contraceptive +tromb +
contraceptive*tromb, scale_max=1.5, effect="contraceptive*tromb",
draw=1000, burnin=300, data=tdata)
print(pres)
plot(pres,3)
pres["estim", 1]$betas
pres["power11", 1]$power
pres["power1", 1]$z
```

---

print,powerClass-method

Method for Function `print`

Description

Method for function `print` with
`signature(x = "powerClass")`

Usage

```
## S4 method for signature 'powerClass'
print(x, choice, ...)
```

Arguments

x	the name of *powerClass* object.
choice	an optional arg containing two choices of print: "power" (by default) prints the results of power analysis, while "model" prints estimated log-linear model parameters and fit statistics.
...	not used

Details

Fit statistic *Jacobian reciprocal condition number* measures the inverse sensitivity of the solution to small perturbations in the input data. It tends to zero as *J* tends to singularity indicating solution instability.
The value of ch-squared per degree of freedom *chisq/dof* approximately 1 indicates a good fit. If *chisq/dof* » 1 the error estimates obtained from the covariance matrix will be too small and should be multiplied by square root of *chisq/dof*.
Poor fit will result from the use of an inappropriate model.
BEWARE: Poor fit jeopardizes the validity of power analysis.

Methods

`signature(x = "powerClass")` Method for function `print` for object of S4 class *powerClass*.

The second argument *choice* controls information to print. It's advisable to start printing with arg *choice="model"*. Besides estimated log-linear model parameters, fit statistics printed for input data given arg *scale_min*=1 in function *PowerPoisson*. Otherwise, it prints results for augmented scale_min*data counts. Of particular importance is *Jacobian reciprocal condition number* and *chisq/dof*. See details.

Arg *choice="power"* prints results of power analysis in given range of sample size regulated by args *scale_min, scale_max* in function *PowerPoisson*. These are multiplyers for observed data counts. Range is divided into 11 even-spaced subsequent sample sizes. Each is described in printed quantiles (Q0.025, Q0.05, Q0.1, Q0.2, Q0.3, Q0.4, Q0.5) of power and z-score distributions. It's suggestive to use Q0.025 in making decision. Given the results one can change sample size range, for example to scrutinize some particular interval to ensure power and p-value.

Examples

```
require(ltable)
data(tdata, package="ltable")
## For better illustration You should increase draw and burnin pars
pres<-MCPower(Counts~smoker +contraceptive +tromb +
contraceptive*tromb, scale_min = 0.5, scale_max=1.5,
effect="contraceptive*tromb", data=tdata, draw=1000, burnin=300)
print(pres, "model")
print(pres, "power")
```

`renewal` *Function* renewal

Description

Estimates MCMC chain convergence based on renewal theory.

Usage

```
renewal(x, StatesNum=10, Astate=NULL, nForStart = 3000, epsilon1=0.05, epsilon2=0.05)
```

Arguments

`x`	name of the numeric vector of the chain; object of *numeric* class
`StatesNum`	positive integer that indicates the number of states to classify chain values into, max value is 100 s.t. values of chain classified into 100 ordered states; to choose StatesNum consider the adequate number of unique intervals chain values can be suitably represented by
`Astate`	positive integer from 1 to StateNum that defines state to base calculation on; there may be small differences in results under different base states, see help for details; default is median state value
`nForStart`	positive integer that indicates the maximum distance from beginning of chain to consider in finding Start value, 3000 by default
`epsilon1`	upper bound of closeness to stationary distribution π_X; it is used to calculate the starting position of the chain *nstat* $$\|P_{x0}^{nstat} - \pi_X\| \leqslant \epsilon_1$$
`epsilon2`	is upper bound of variance of estimator $$Var(\frac{1}{\ell} \sum_{nstat+1}^{nvar} X_k - E_{\pi_X}) \leqslant \epsilon_2$$

Details

- You can ascertaine if the sampled transition probability is close to the determined stationary probability of Markov Chain and how many iterations should be used in order to minimize the error of estimator.
- Algorithm is based on renewal theory and implements the concept of the so called "secondary chain". It is supported by strong mathematical reasoning and the obtained solutions are strict, i.e. they are not asymptotic. Hence, this method is not biased by additional error provided by limit theorems.
- If the calculated chain length surpasses length of supplied chain whatever chain length is provided it is the indication that chain based estimator deviates from stationary distribution based more than indicated by *epsilon2*.
- The larger the *epsilon1* the larger the starting value of the chain. The larger the *epsilon2* the lengthier is the chain.

Value

returns two positive integers, starting value of the chain and length of the chain to be used from starting value.

Note

Abstain from repeated supply of lengthier chain. It does not correct the problem of nonstationarity.

Author(s)

Ocheredko Oleksandr <Ocheredko@yahoo.com>

Examples

```
require(ltable)
data(tdata, package="ltable")
res1<-MCLogLin(Counts~smoker +contraceptive +tromb +
contraceptive*tromb, data=tdata, draw=5000, burnin=500)
renewal(res1[,1], Astate=5)
renewal(res1[,14], Astate=1)
```

---

sdata *Simple Data.*

Description

This data has no other meaning and purpose except for package functionality presentation.

Usage

```
data(sdata)
```

Format

A data frame with 22 observations (some values are purposely missing) on the following 4 variables.

- `a` a logical vector
- `b` a numeric vector
- `c` a factor with levels `female` and `male`
- `d` a character vector with variants "A" and "B"

Details

You can construct tables with data set fields of factor, character, logical, and numeric classes.

Examples

```
data(sdata, package="ltable")
```

SimData — *Simulated interval censored survival data to model hazard.*

Description

Simulated interval censored survival data. For details see user guide *ltable2.0.3.pdf*.

Usage

```
data(SimData)
```

Format

Description of data frames:

Fields:

1.	T	Treatment status: 0-1	numeric
2.	C	Comorbidity free status: 0-1	numeric
3.	Year	Year of event: 1-3	numeric
4.	Counts	Generated number of events	numeric
5.	Year2	Dummy var. of Year2: 0-1	numeric
6.	Year3	Dummy var. of Year3: 0-1	numeric
7.	offset	Person-years of risk	numeric

Examples

```
require(ltable)
data(SimData, package="ltable")
res3<-MCLogLin(Counts~Year2+Year3+T+C, offset=offset, data=SimData, draw=1500, burnin=500)
```

SMdata — *Heart surgery data to model standardized (mortality) ratio.*

Description

Counts of patient deaths following heart transplant surgery in 131 hospitals in the US between October 1987 and December 1989. These were analysed by Christiansen and Morris (1996, 1997). Data is also analysed by Peter D. Congdon. Data supplied with file *offsetdata.rda*.

Usage

```
data(SMdata)
```

Format

Description of data frames:

Fields:

1.	y	Number of Deaths	numeric
2.	o	Number of Expected Deaths	numeric

References

Peter D. Congdon. Bayesian Hierarchical Models With Applications Using R (2020) Second Edition. Example 4.5 Hospital Mortality, p.125-26.

Examples

```
require(ltable)
data(SMdata, package="ltable")
res2<-MCLogLin(y~1, offset=o, data=SMdata, draw=1500, burnin=500, DIC=TRUE)
```

`tableToData` *Function* tableToData

Description

Constructs *data.frames* that fit *glm{stats}* or *MCLogLin{ltable}*, *MCPower{ltable}* modelling out of tables created with function *table_f{ltable}*.

Usage

```
tableToData(tname, numerictype="", orderedtype="")
```

Arguments

`tname`	name of the tables created with function *table_f*; object of *data.frame* class
`numerictype`	the character string that lists variable names separated by comma to be transformed to numeric class. Variable "Counts" shouldn't be listed
`orderedtype`	the character string that lists variable names separated by comma to be transformed to ordered factor class. Variable "Counts" shouldn't be listed

Details

- Variables of character and logical classes shape the same design as does the factor class, therefore there is no need to change them to factors.

- Check the input and output. It's not a problem to have huge counts together with zero counts for NB2 model used in *ltable*. Still good practice to proceed with power analisis is to have data without zero counts. It's in no way detrimental as in the case of the Poisson GLM, having the mean and variance equality. The implication with Poisson GLM is that as the mean tends to zero, so must the variance. Still we do have some uncertainty about this fitted value. Of the same nature (but converse) problem is with cells of large counts.
- You can build the data by hand and skip this functionality.

Value

returns object of class *data.frame*

Author(s)

Ocheredko Oleksandr <Ocheredko@yahoo.com>

See Also

reshape

Examples

```
require(ltable)
data(iris)
iriscut<-with(iris, data.frame(PL=cut(Petal.Length,3),
                               PW=cut(Petal.Width,3)))
irist<-table_f(iriscut,"PL,PW")
irisd<-tableToData(irist, ordered="PL,PW")
```

---

table_f *Function* table_f

Description

Constructs tables of counts and proportions out of data sets.

Usage

```
table_f(data, datavars, type=1, digits=2, extended=FALSE, MV=FALSE, cb=FALSE)
```

Arguments

data	name of the data set; object of *data.frame* class
datavars	the character string that lists field names separated by comma in the order of presentation in the table: first has its sorted levels rolled out vertically leftmost, the last has its sorted levels spread by columns

type	the type of table: 1 (default) - count table; 2 - proportions by rows; 3 - proportions by columns; 4 - frequencies
digits	formats output digits number, applied only to proportions, default is 2
extended	TRUE adds margins of counts, applied only for proportions and frequencies, FALSE by default
MV	includes missing values into tabulation, operates with type=1 only, FALSE by default
cb	TRUE permits to copy the table to clipboard, FALSE by default

Details

- You can construct table with data set fields of factor, character, logical, and numeric classes.
- To insert table into Word document first open Excel, choose left high corner of placement by mouse click and use Ctrl+V combination or click on the Paste icon (the clipboard), then use Ctrl+C, open Word document, use Ctrl+V to place the table.
- If You want to use clipboard to insert table into Word document use option cb=TRUE. You will be asked to confirm, for previous information of clipboard would be lost.

Value

returns object of class *data.frame*

Note

Abstain from putting continuous variables or too many factor variables into *datavars* list to keep table legible. Put factor variable with numerous levels at the end of the list.

Author(s)

Ocheredko Oleksandr <Ocheredko@yahoo.com>

Examples

```
data(sdata, package="ltable")
table_f(sdata, "a")
table_f(sdata, "a", MV=TRUE, extended=TRUE)
knitr::kable(table_f(sdata, "a,b,c"))
table_f(sdata, "a,b,c,d", type=2, digits=3)
table_f(sdata, "b,c,a,d", MV=TRUE, extended=TRUE, cb=TRUE)
```

tdata *Tromboembolism Data.*

Description

Case-control data first considered by Worcester, J (1971). The data cross-classify tromboembolism and control patients by two risk factors: oral contraceptive user and smoking. Test quantifies boosting effect of contraceptive on odds of tromboembolism.
Data used in examples of power analysis.

Usage

```
data(tdata)
```

Format

A grouped data frame with 8 rows of factors' levels combinations. Factors are: smoking status (Yes, No), contraceptive usage (Yes, No), thromboembolism status (Trombol, Control).

`smoker` a character vector

`contraceptive` a character vector

`tromb` a character vector

`Counts` a numeric vector

Details

One can use tables created by function *table_f* transformed with function *tableToData* to appropriate data.frame format with fields of factor, character, logical, and numeric classes. Or one can build data by hand with *data.frame* facility.

References

Worcester, J (1971). The relative odds in the 2 by 3 contingency table. American Journal of Epidemiology, 93, 145-149.

Examples

```
data(tdata, package="ltable")
```

`[,powerClass-method` *Method for Function* `[`

Description

Method for function `[` with
`signature(x = "powerClass")`

Usage

```
## S4 method for signature 'powerClass'
x[i, j, drop]
```

Arguments

`x`	the name of *powerClass* object.
`i`	the name of the slot of the object
`j`	picks up j-th element of the list in slot with name &i.
`drop`	not used

Details

Method provides access to slots of *powerClass* object. Its structure delivered in *powerClass-class* index. Access to particular vectors of lists supplied with $ operator. For example, log-linear reg.coefficients estimates of smallest size data accessible by obj["estim", 1]$betas, errors can be obtained by analogue: obj["estim", 1]$errors. Power values extraction slightly differs: obj["power11", 1]&power extracts power values vector for 1st effect given 11th (largest) sample size. By analogue we get vector of z-scores for second effect given smallest sample size by obj["power1", 2]&z. See *powerClass-class* index.

Methods

`signature(x = "powerClass", i = "character", j = "integer", drop = "logical")` Method for function `[` for object of S4 class *powerClass*.

Examples

```
require(ltable)
require(ltable)
data(tdata, package="ltable")
## For better illustration You should increase draw and burnin pars
pres<-MCPower(Counts~smoker +contraceptive +tromb +
contraceptive*tromb, scale_max=1.5, effect="contraceptive*tromb",
data=tdata, draw=1000, burnin=300)
# get call
pres["cal"]
```

```
# get effect contrasts names
pres["effectsname"]
# get Jacobian reciprocal condition number in smallest sample
pres["estim",1]$Jacobian_rcnumber
# get chisq/n in smallest sample
pres["estim",1]$chi_sq
# get LogLikelihood
pres["estim",1]$LL
# get initial deviation between observed and expected counts
pres["estim",1]$dev0
# get final deviation between observed and expected counts
pres["estim",1]$dev
```

[<-,powerClass-method *Method for Function* [<-

Description

Method for function [<- with
signature(x = "powerClass")

Arguments

x	the name of *powerClass* object.
i	the name of the slot of the object
j	picks up j-th element of the list in slot with name &i.
value	values to set

Details

Set method provides access to slots of *powerClass* object. Its structure delivered in *powerClass-class* index. Access to particular vectors of lists supplied with $ operator. For example, log-linear reg.coefficients estimates of smallest size data accessible by obj["estim", 1]$betas, errors can be obtained by analogue: obj["estim", 1]$errors. Power values extraction slightly differs: obj["power11", 1]&power extracts power values vector for 1st effect given 11th (largest) sample size. By analogue we get vector of z-scores for second effect given smallest sample size by obj["power1", 2]&z. It's hardly matter of practicality to employ set method but for programming purpose. See *powerClass-class* index.

Methods

signature(x = "powerClass", i = "character", j = "integer", value = "ANY") Method for function [<- for object of S4 class *powerClass*.

Index