Package 'ltable'

November 22, 2021

Type Package

Title Easy to Make (Lazy) Tables

Version 2.0.1

Date 2021-11-12

Author Ocheredko Oleksandr

Description Constructs tables of counts and proportions out of data sets. It has simplified syntax appealing for novice and even for advanced user under time pressure. It is particularly suitable for exploratory data analysis or presentation to single out most appropriate pieces of tabulated information. The other important feature is possibility to insert tables to Excel and Word documents. This version also features capacity of log-linear and power analyses (original by Oleksandr Ocheredko <doi:10.35566/isdsa2019c5>) for tabulated data with GSL.

Maintainer Ocheredko Oleksandr <Ocheredko@yahoo.com>

License GPL (>= 2)

Depends R (>= 4.1.0), methods, clipr

Imports Rcpp (>= 1.0.5), RcppGSL (>= 0.3.9), graphics, stats

Suggests MCMCglmm, knitr, datasets

Collate PowerPoisson.R RcppExports.R Plot.R Print.R table_f.R tableToData.R

SystemRequirements 'GSL'

NeedsCompilation yes

OS_type unix

LinkingTo Rcpp, RcppGSL

Repository CRAN

Date/Publication 2021-11-22 11:40:05 UTC

R topics documented:

`ltable-package` *ltable*

Description

Constructs tables of counts and proportions out of data sets.

Details

In order to perform log-linear and power analyses for tabulated data GSL: GNU Scientific Library has to be installed first (reference 1). Log-linear analysis features some advantages against glm {stats}, first of all due to stability of GSL IWLS algorithms that insures distinctly less biased covariances estimates, pivot issue for implemented power analysis. In some instances hypothesis testing of higher order effects disagrees with that of glm on account of larger GSL estimated errors. 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 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.

Author(s)

Ocheredko Oleksandr <Ocheredko@yahoo.com>

References

1. GSL: GNU Scientific Library: http://www.gnu.org/software/gsl/,
2. 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)
```

`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")
pres<-PowerPoisson(Counts~smoker +contraceptive +tromb +
contraceptive*tromb, scale_max=1.5, effect="contraceptive*tromb", data=tdata)
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

1. GSL: GNU Scientific Library: http://www.gnu.org/software/gsl/,
2. 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")
pres<-PowerPoisson(Counts~smoker +contraceptive +tromb +
contraceptive*tromb, scale_max=1.5, effect="contraceptive*tromb", data=tdata)
print(pres)
plot(pres)
pres["estim", 1]$betas
pres["power11", 1]$power
pres["power1", 1]$z
```

powerpoisson *Function* PowerPoisson

Description

Performs log-linear and power analyses for constructed tabulated data based on GSL IWLS algorithms

Usage

```
PowerPoisson(formula, data, scale_min=1, scale_max=5,
effect, p_alpha=0.05, contrasts=NULL )
```

Arguments

formula	a symbolic description of the model to be fit.
data	name of the data set; object of *data.frame* class
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
contrasts	serves to choose types of contrasts to study effects of factors, same with glm), orthogonal polynomials by default

Details

- Performs log-linear modelling with supplied data by using GSL IWLS algorithms.
- 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 auspices of IWLS GSL algorithms, 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/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 and jeopardizes the validity of power analysis.

Author(s)

Ocheredko Oleksandr <Ocheredko@yahoo.com>

See Also

glm MCMCglmm

Examples

```
require(ltable)
data(tdata, package="ltable")
pres<-PowerPoisson(Counts~smoker +contraceptive +tromb +
contraceptive*tromb, scale_max=1.5, effect="contraceptive*tromb", data=tdata)
print(pres, "model")
print(pres)
plot(pres, stencil=3)
plot(pres)
```

---

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")
pres<-PowerPoisson(Counts~smoker +contraceptive +tromb +
contraceptive*tromb, scale_max=1.5, effect="contraceptive*tromb", data=tdata)
print(pres, "model")
print(pres)
```

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")
```

tableToData *Function* tableToData

Description

Constructs data.frames that fit glm or PowerPoisson modelling out of tables created with function *table_f*.

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. Good practice is to have data without zero counts. In the Poisson GLM, the mean and variance are the same thing. The implication of this 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(sdata, package="ltable")
stab<-table_f(sdata, "a,b,c")
sdat<-tableToData(stab, orderedtype="c")
res<-PowerPoisson(Counts~a+b+c+a*c, effect="a*c", data=sdat)
plot(res, st=3)
```

---

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.

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)
data(tdata, package="ltable")
pres<-PowerPoisson(Counts~smoker +contraceptive +tromb +
contraceptive*tromb, scale_max=1.5, effect="contraceptive*tromb", data=tdata)
# get call
pres["cal"]
# get effect contrasts names
pres["effectsname"]
# get Jacobian reciprocal condition number in smallest sample
```

```
pres["estim",1]$`Jacobian reciprocal condition number`
# get chisq/dof in smallest sample
pres["estim",1]$`chisq/dof`
# get reason for stopping iterations
pres["estim",1]$`reason for stopping`
# get initial sum of squared differences between observed and expected counts
pres["estim",1]$`initial |f(x)|`
# get final sum of squared differences between observed and expected counts
pres["estim",1]$`final |f(x)|`
# get iteration number of GSL non-linear LS fitting
pres["estim",1]$`number of iterations`
```

`[<-,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*.

Examples

```
require(ltable)
data(tdata, package="ltable")
pres<-PowerPoisson(Counts~smoker +contraceptive +tromb +
contraceptive*tromb, scale_max=1.5, effect="contraceptive*tromb", data=tdata)
# get call
pres["cal"]
# get effect contrasts names
pres["effectsname"]
# get Jacobian reciprocal condition number in smallest sample
pres["estim",1]$`Jacobian reciprocal condition number`
# get chisq/dof in smallest sample
pres["estim",1]$`chisq/dof`
# get reason for stopping iterations
pres["estim",1]$`reason for stopping`
# get initial sum of squared differences between observed and expected counts
pres["estim",1]$`initial |f(x)|`
# get final sum of squared differences between observed and expected counts
pres["estim",1]$`final |f(x)|`
# get iteration number of GSL non-linear LS fitting
pres["estim",1]$`number of iterations`
```

Index