Package 'RVAideMemoire'

August 12, 2014

Encoding latin1

Type Package

Title Diverse basic statistical and graphical functions

Version 0.9-38

Date 2014-08-12

Author Maxime Hervé

Maintainer Maxime Hervé <mx.herve@gmail.com>

Imports ade4, boot, car, lme4 (>= 1.0-4), lsmeans (>= 2.0.1), MASS,mixOmics (>= 5.0.2), multcomp, statmod, stats

Suggests ordinal, survival

Description This package contains diverse more or less complicated functions, written to simplify user's life: simplifications of existing functions, basic but not implemented tests, easy-to-use tools, bridges between functions of different packages... All functions are presented in the French book 'Aide-memoire de statistique appliquee a la biologie', written by the same author and available on CRAN.

License GPL-2

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-12 20:11:44

R topics documented:

`RVAideMemoire-package` *Diverse basic statistical and graphical functions*

Description

This package contains diverse more or less complicated functions, written to simplify user's life: simplifications of existing functions, basic but not implemented tests, easy-to-use tools, bridges between functions of different packages... All functions are presented in the French book 'Aide-memoire de statistique appliquee a la biologie', written by the same author and available on CRAN.

Details

Package:	RVAideMemoire
Type:	Package
Version:	0.9-38
Date:	2014-08-12
License:	GPL-2
LazyLoad:	yes

Author(s)

Maxime Hervé

Maintainer: Maxime Hervé <mx.herve@gmail.com>

References

Document : "Aide-memoire de statistique appliquee a la biologie - Construire son etude et analyser les resutats a l'aide du logiciel R" (available on CRAN)

`Anova.clm` *Anova Tables for Cumulative Link (Mixed) Models*

Description

These functions are methods for `Anova` to calculate type-II or type-III analysis-of-deviance tables for model objects produced by `clm` and `clmm`. Likelihood-ratio tests are calculated in both cases.

Usage

```
## S3 method for class 'clm'
Anova(mod, type = c("II", "III", 2, 3), ...)

## S3 method for class 'clmm'
Anova(mod, type = c("II", "III", 2, 3), ...)
```

Arguments

`mod`	`clm` or `clmm` object.
`type`	type of test, `"II"`, `"III"`, `2` or `3`.
`...`	additional arguments to `Anova`. Not usable here.

Details

See help of the `Anova` for a detailed explanation of what "type II" and "typ III" mean.

Value

See `Anova`.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`Anova`, `clm`, `clmm`

`bootstrap` *Bootstrap*

Description

Simplified version of the `boot` function.

Usage

```
bootstrap(x, fun, nrep = 1000, conf.level = 0.95, ...)
```

Arguments

`x`	numeric vector.
`fun`	function to be used for computation (`function(x,i) ...(x[i])`).
`nrep`	number of replicates.
`conf.level`	confidence level for confidence interval.
`...`	additional arguments to `boot`. See help of this function.

Details

See help of the `boot` function for more explanations.

Value

`method`	the character string `"Bootstrap"`
`data.name`	a character string giving the name of the data.
`estimate`	the estimated original value
`conf.level`	confidence level for confidence interval.
`rep`	number of replicates.
`conf.int`	limits of the confidence interval.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`boot`

Examples

```
# Confidence interval of a mean
samp <- sample(1:50,10,replace=TRUE)
bootstrap(samp,function(x,i) mean(x[i]))

# Confidence interval of the standard error of the mean
bootstrap(samp,function(x,i) sd(x[i])/sqrt(length(x[i])))
```

`byf.hist` *Histogram for factor levels*

Description

Draw a histogram of a numeric variable per level of a factor.

Usage

```
byf.hist(formula, data, ...)
```

Arguments

`formula`	a formula of the form `a ~ b` where `a` gives the data values and `b` a factor giving the corresponding groups.
`data`	an optional data frame containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`...`	other arguments to pass to `hist`.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`hist`

Examples

```
data(iris)
byf.hist(Sepal.Length~Species,data=iris)
```

byf.mqqnorm *QQ-plot for factor levels*

Description

Draw a multivariate QQ-plot of numeric variables per level of a factor.

Usage

```
byf.mqqnorm(formula, data)
```

Arguments

formula	a formula of the form `a ~ b`, where `a` is a matrix giving the dependent variables (each column giving a variable) and `b` a factor giving the corresponding groups.
data	an optional data frame containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`mqqnorm`, `byf.mshapiro`, `qqPlot`

Examples

```
data(iris)
byf.mqqnorm(as.matrix(iris[,1:4])~Species,data=iris)
```

byf.mshapiro *Shapiro-Wilk test for factor levels*

Description

Perform a multivariate Shapiro-Wilk test on numeric variables per level of a factor.

Usage

```
byf.mshapiro(formula, data)
```

Arguments

formula	a formula of the form a ~ b where a is a matrix giving the dependent variables (each column giving a variable) and b a factor giving the corresponding groups.
data	an optional data frame containing the variables in the formula formula. By default the variables are taken from environment(formula).

Value

method	name of the test.
data.name	a character string giving the names of the data.
tab	table of results.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

byf.mqqnorm, mshapiro.test, qqPlot

Examples

```
data(iris)
byf.mshapiro(as.matrix(iris[,1:4])~Species,data=iris)
```

---

byf.normhist *Histogram and normal distribution for factor levels*

Description

Draw on the same histogram of a numeric variable a histogram per level of a factor, and add the normal distribution line of each level.

Usage

```
byf.normhist(formula, data)
```

Arguments

formula	a formula of the form a ~ b where a gives the data values and b a factor giving the corresponding groups.
data	an optional data frame containing the variables in the formula formula. By default the variables are taken from environment(formula).

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
data(iris)
byf.normhist(Sepal.Length~Species,data=iris)
```

---

byf.qqnorm *QQ-plot for factor levels*

Description

Draw a QQ-plot of a numeric variable per level of a factor.

Usage

```
byf.qqnorm(formula, data, ...)
```

Arguments

formula	a formula of the form a ~ b where a gives the data values and b a factor giving the corresponding groups.
data	an optional data frame containing the variables in the formula formula. By default the variables are taken from environment(formula).
...	other arguments to pass to qqPlot.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

link[RVAideMemoire]{byf.shapiro}, qqPlot

Examples

```
data(iris)
byf.qqnorm(Sepal.Length~Species,data=iris)
```

byf.shapiro *Shapiro-Wilk test for factor levels*

Description

Perform a Shapiro-Wilk test on a numeric variable per level of a factor.

Usage

```
byf.shapiro(formula, data)
```

Arguments

formula	a formula of the form a ~ b where a gives the data values and b a factor giving the corresponding groups.
data	an optional data frame containing the variables in the formula formula. By default the variables are taken from environment(formula).

Value

method	name of the test.
data.name	a character string giving the names of the data.
tab	table of results.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

byf.qqnorm, shapiro.test

Examples

```
data(iris)
byf.shapiro(Sepal.Length~Species,data=iris)
```

`chisq.bintest` *Pearson's Chi-squared test or Fisher's exact test for binary variables*

Description

Perform a Pearson's Chi-squared test for comparing response probabilities (i.e. when the response variable is a binary variable), or a Fisher's exact test for count data if more than 20% of expected counts are <5 (Cochran's rule). The function is in fact a wrapper to the chi-squared test for comparison of proportions on a contingency table. If the p-value of the test is significative, the function performs pairwise comparisons, by using Pearson's Chi-squared test or Fisher's exact test depending on the respect to Cochran's rule.

Usage

```
chisq.bintest(formula, data, alpha = 0.05, p.method = "fdr")
```

Arguments

`formula`	a formula of the form `a ~ b`, where `a` and `b` give the data values and corresponding groups, respectively. `a` can be a numeric vector or a factor, with only two possible values (except NA).
`data`	an optional data frame containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`alpha`	significance level to compute pairwise comparisons.
`p.method`	method for p-values correction. See help of `p.adjust`.

Value

`method.test`	a character string giving the name of the global test computed.
`data.name`	a character string giving the name(s) of the data.
`alternative`	a character string describing the alternative hypothesis.
`estimate`	the estimated probabilities.
`null.value`	the value of the difference in probabilities under the null hypothesis, always 0.
`statistic`	test statistics (Pearson's Chi-squared test only).
`parameter`	test degrees of freedom (Pearson's Chi-squared test only).
`p.value`	p-value of the global test.
`alpha`	significance level.
`p.adjust.method`	method for p-values correction.
`p.value.multcomp`	data frame of pairwise comparisons result.
`method.multcomp`	a character string giving the name of the test computed for pairwise comparisons.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
response <- c(0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,1,1,0,0,1,1,1,1,1,1,0,0,1,1,1)
fact <- gl(3,10,labels=LETTERS[1:3])
chisq.bintest(response~fact)
```

---

chisq.exp *Expected counts for comparison of proportions to given values*

Description

Return expected counts before comparing proportions to given values by a chi-squared test.

Usage

```
chisq.exp(data, p, graph = FALSE)
```

Arguments

data	contingency table.
p	theoretical proportions.
graph	logical. If TRUE a mosaic plot of expected counts is drawn.

Details

The function returns how many counts can be < 5 to respect Cochran's rule (80% of counts must be >= 5).

Value

p.theo	theoretical proportions.
mat	contingency table of expected counts.
cochran	number of counts which can be < 5.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

prop.test, chisq.test, mosaicplot

Examples

```
proportions <- sample(c(0,1),60,replace=TRUE)
populations <- sample(LETTERS[1:3],60,replace=TRUE)
tab.cont <- table(populations,proportions)
p.theo <- c(0.2,0.5,0.7)
chisq.exp(tab.cont,p=p.theo)
```

`chisq.multcomp` *Pairwise comparisons after a chi-squared goodness-of-fit test*

Description

Perform pairwise comparisons after a global chi-squared goodness-of-fit test.

Usage

```
chisq.multcomp(x, p.method = "fdr")
```

Arguments

`x`	numeric vector (counts).
`p.method`	method for p-values correction. See help of `p.adjust`.

Value

`method`	name of the test.
`data.name`	a character string giving the name(s) of the data.
`p.adjust.method`	method for p-values correction.
`p.value`	table of results.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`chisq.test`

Examples

```
counts <- c(5,15,23,8,14)
chisq.test(counts)
chisq.multcomp(counts)
```

`chisq.theo.multcomp` *Pairwise comparisons after a chi-squared test for given probabilities*

Description

Perform pairwise comparisons after a global chi-squared test for given probabilities.

Usage

```
chisq.theo.multcomp(x, p = rep(1/length(x), length(x)), p.method = "fdr")
```

Arguments

`x`	numeric vector (counts).
`p`	theoretical proportions.
`p.method`	method for p-values correction. See help of `p.adjust`.

Value

`method`	name of the test.
`data.name`	a character string giving the name(s) of the data.
`observed`	observed counts.
`expected`	expected counts.
`p.adjust.method`	method for p-values correction.
`statistic`	statistics of each test.
`p.value2`	corrected p-values.
`p.value`	data frame of results.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`chisq.test`

Examples

```
counts <- c(5,15,23,8,14)
p.theo <- c(0.1,0.4,0.3,0.15,0.05)
chisq.test(counts,p=p.theo)
chisq.theo.multcomp(counts,p=p.theo)
```

`cochran.qtest` *Cochran's Q test*

Description

Perform the Cochran's Q test for unreplicated randomized block design experiments with a binary response variable and paired data. If the p-value of the test is significant, the function performs pairwise comparisons by using the Wilcoxon sign test.

Usage

```
cochran.qtest(formula, data, alpha = 0.05, p.method = "fdr")
```

Arguments

`formula`	a formula of the form `a ~ b	c`, where `a`, `b` and `c` give the data values and corresponding groups and blocks, respectively. `a` can be a numeric vector or a factor, with only two possible values.
`data`	an optional data frame containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`alpha`	significance level to compute pairwise comparisons.
`p.method`	method for p-values correction. See help of `p.adjust`.

Value

`method.test`	a character string giving the name of the global test computed.
`data.name`	a character string giving the name(s) of the data.
`alternative`	a character string describing the alternative hypothesis.
`estimate`	the estimated probabilities.
`null.value`	the value of the difference in probabilities under the null hypothesis, always 0.
`statistic`	test statistics (Pearson's Chi-squared test only).
`parameter`	test degrees of freedom (Pearson's Chi-squared test only).
`p.value`	p-value of the global test.
`alpha`	significance level.
`p.adjust.method`	method for p-values correction.
`p.value.multcomp`	data frame of pairwise comparisons result.
`method.multcomp`	a character string giving the name of the test computed for pairwise comparisons.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
response <- c(0,1,1,0,0,1,0,1,1,1,1,1,0,0,1,1,0,1,0,1,1,0,0,1,0,1,1,0,0,1)
fact <- gl(3,1,30,labels=LETTERS[1:3])
block <- gl(10,3,labels=letters[1:10])
cochran.qtest(response~fact|block)
```

---

cor.2comp *Comparison of 2 Pearson's linear correlation coefficients*

Description

Perform the test for equality of 2 Pearson's correlation coefficients. If the difference is not significative, the function returns the common coefficient, its confidence interval and performs the test for equality to a given value.

Usage

```
cor.2comp(var1, var2, var3, var4, alpha = 0.05, conf.level = 0.95, theo = 0)
```

Arguments

var1	numeric vector (first variable of the first correlation).
var2	numeric vector (second variable of the first correlation).
var3	numeric vector (first variable of the second correlation).
var4	numeric vector (second variable of the second correlation).
alpha	significance level.
conf.level	confidence level.
theo	theoretical coefficient.

Value

method.test	a character string giving the name of the global test computed.
data.name	a character string giving the name(s) of the data.
statistic	test statistics.
p.value	p-value for comparison of the 2 coefficients.
null.value	the value of the difference in coefficients under the null hypothesis, always 0.
alternative	a character string describing the alternative hypothesis.
estimate	the estimated correlation coefficients.
alpha	significance level.

conf.level	confidence level.
common.name	a character string explaining the elements of the table below.
common	data frame of results if the coefficients are not significantly different (common coefficient).

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

cor.test

Examples

```
cor1.var1 <- 1:30+rnorm(30,0,2)
cor1.var2 <- 1:30+rnorm(30,0,3)
cor2.var1 <- (-1):-30+rnorm(30,0,2)
cor2.var2 <- (-1):-30+rnorm(30,0,3)
cor.2comp(cor1.var1,cor1.var2,cor2.var1,cor2.var2)
```

---

cor.conf *Equality of a Pearson's linear correlation coefficient to a given value*

Description

Perform a test for equality of a Pearson's linear correlation coefficient to a given value.

Usage

```
cor.conf(var1, var2, theo)
```

Arguments

var1	numeric vector (first variable).
var2	numeric vector (second variable).
theo	theoretical value.

Value

method	a character string giving the name of the test.
data.name	a character string giving the name(s) of the data.
statistic	test statistics.
p.value	p-value of the test.
null.value	the value of the theoretical coefficient.
alternative	a character string describing the alternative hypothesis.
estimate	the estimated correlation coefficient.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

cor.test

Examples

```
var1 <- 1:30+rnorm(30,0,4)
var2 <- 1:30+rnorm(30,0,4)
cor.conf(var1,var2,theo=0.5)
```

---

cor.multcomp *Comparison of several Pearson's linear correlation coefficients*

Description

Perform comparisons of several Pearson's linear correlation coefficients. If no difference, the function returns the common correlation coefficient, its confidence interval and test for its equality to a given value. If difference is significative, the functions performs pairwise comparisons between coefficients.

Usage

```
cor.multcomp(var1, var2, fact, alpha = 0.05, conf.level = 0.95, theo = 0,
  p.method = "fdr")
```

Arguments

var1	numeric vector (first variable).
var2	numeric vector (second variable).
fact	factor (groups).
alpha	significance level.
conf.level	confidence level.
theo	theoretical coefficient.
p.method	method for p-values correction. See help ofp.adjust.

Value

method.test	a character string giving the name of the global test computed.
data.name	a character string giving the name(s) of the data.
statistic	test statistics.
parameter	test degrees of freedom.

p.value	p-value for comparison of the coefficients.
null.value	the value of the difference in coefficients under the null hypothesis, always 0.
alternative	a character string describing the alternative hypothesis.
estimate	the estimated correlation coefficients.
alpha	significance level.
conf.level	confidence level.
p.adjust.method	method for p-values correction.
p.value.multcomp	data frame of pairwise comparisons result.
common.name	a character string explaining the elements of the table below.
common	data frame of results if the coefficients are not significantly different (common coefficient).

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

cor.test

Examples

```
var1 <- c(1:15+rnorm(15,0,4),1:15+rnorm(15,0,1),1:15+rnorm(15,0,8))
var2 <- c(-1:-15+rnorm(15,0,4),1:15+rnorm(15,0,1),1:15+rnorm(15,0,8))
fact <- gl(3,15,labels=LETTERS[1:3])
cor.multcomp(var1,var2,fact)

var3 <- c(1:15+rnorm(15,0,1),1:15+rnorm(15,0,3),1:15+rnorm(15,0,2))
cor.multcomp(var1,var3,fact)
```

---

cox.resid *Martingale residuals of a Cox model*

Description

Plot martingale residuals of a Cox model against fitted values, to check for log-linearity of covariates.

Usage

```
cox.resid(model)
```

Arguments

model	a coxph model.

Author(s)

Maxime Hervé <mx.herve@gmail.com>, based on a John Fox idea.

References

Fox, J. 2002 Cox Proportional-Hazards Regression for Survival Data.

See Also

coxph

Examples

```
# 'kidney' dataset of package 'survival'
require(survival)
data(kidney)
model <- coxph(Surv(time,status)~age+factor(sex),data=kidney)
cox.resid(model)
```

---

cramer.test *Cramer's association coefficient*

Description

Compute the Cramer's association coefficient between 2 nominal variables, its confidence interval (by bootstraping) and tests for its significance.

Usage

```
cramer.test(x, y, nrep = 1000, conf.level = 0.95)
```

Arguments

x	a contingency table ('matrix' or 'table' object). x and y can also both be factors.
y	ignored if x is a contingency table. If not, y should be a vector of the same length.
nrep	number of replicates for bootstraping.
conf.level	confidence level.

Value

method	name of the test.
statistic	test statistics.
parameter	test degrees of freedom.
p.value	test p-value.
data.name	a character string giving the names of the data.

estimate	Cramer's coefficient.
conf.level	confidence level.
rep	number of replicates.
conf.int	confidence interval.
alternative	a character string giving the alternative hypothesis, always "two.sided"
null.value	the value of the association measure under the null hypothesis, always 0.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

boot

Examples

```
var1 <- sample(LETTERS[1:3],30,replace=TRUE)
var2 <- sample(letters[1:3],30,replace=TRUE)
cramer.test(var1,var2)
# or
cramer.test(table(var1,var2))
```

---

cv *Coefficient of variation*

Description

Compute the coefficient of variation of a vector.

Usage

```
cv(x, abs = TRUE, pc = TRUE)
```

Arguments

x	numeric vector.
abs	logical. If TRUE the coefficient is expressed in absolute value.
pc	logical. If TRUE the coefficient is expressed in percentage.

Details

The function deals with missing values.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
cv(rnorm(30))
```

`DA.confusion` *Classification error rate of a Discriminant Analysis*

Description

Compute the classification error rate of a Discriminant Analysis. The function divides the data in a training and a testing dataset, and predict the class of the individuals of the testing dataset by using the model fitted with the training dataset. Linear Discriminant Analysis (by using `lda`) and Partial Least Squares - Discriminant Analysis (by using `plsda`) are handled.

Usage

```
DA.confusion(model, train = 2/3, crit.lda = c("plug-in", "predictive","debiased"),
  crit.plsda = c("mahalanobis.dist", "centroids.dist", "max.dist"))
```

Arguments

`model`	object of class `"lda"` (from `lda`) or `"plsda"` (from `plsda`).
`train`	proportion of the total number of individuals to be used to build the training dataset.
`crit.lda`	this determines how the parameter estimation is handled in LDA cross-validation. With `"plug-in"` (the default) the usual unbiased parameter estimates are used and assumed to be correct. With `"debiased"` an unbiased estimator of the log posterior probabilities is used, and with `"predictive"` the parameter estimates are integrated out using a vague prior.
`crit.plsda`	prediction method to be applied for PLS-DA cross-validation. Should be a subset of `"mahalanobis.dist"` (default), `"centroids.dist"` or `"max.dist"`.

Details

When working on a LDA, the prior probabilities for the model fitted on the training dataset are extracted from `model` (they are not evaluated from the training dataset itself).

Value

`model`	type of discriminant model used.
`crit.lda`	method for parameter estimation when working on a LDA.
`crit.plsda`	distance used when working on a PLS-DA.
`prop.train`	a vector giving the number of individuals used to build the training dataset (`used`) and the total number of individuals (`total`).
`ind.for.train`	individuals used to build the training dataset.
`predicted`	vector of predicted classes.
`confusion`	confusion matrix.
`prop.confusion`	classification error rate.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

lda, predict.lda, plsda, predict.plsda

Examples

```
# LDA model
require(MASS)
data(iris)
model.LDA <- lda(iris[,1:4],iris$Species)
DA.confusion(model.LDA)

# PLS-DA model
require(mixOmics)
data(yeast)
model.PLSDA <- plsda(t(yeast$data),yeast$strain.cond)
DA.confusion(model.PLSDA)
```

DA.valid *Cross-validation in Discriminant Analysis*

Description

Perform a cross-validation to assess the prediction ability of a Discriminant Analysis. Linear Discriminant Analysis (by using lda) and Partial Least Squares - Discriminant Analysis (by using plsda) are handled. Two methods are implemented for cross-validation: leave-one-out and M-fold.

Usage

```
DA.valid(model, method = c("loo", "Mfold"), crit.lda = c("plug-in", "predictive",
  "debiased"), crit.plsda = c("mahalanobis.dist", "centroids.dist", "max.dist"),
  M = 10, nrep = 20)
```

Arguments

model	object of class "lda" (from lda) or "plsda" (from plsda).
method	what kind of validation to use, matching one of "Mfold" or "loo" (see below). Default is "Mfold".
crit.lda	this determines how the parameter estimation is handled in LDA cross-validation. With "plug-in" (the default) the usual unbiased parameter estimates are used and assumed to be correct. With "debiased" an unbiased estimator of the log posterior probabilities is used, and with "predictive" the parameter estimates are integrated out using a vague prior.

crit.plsda	prediction method to be applied for PLS-DA cross-validation. Should be a subset of "mahalanobis.dist" (default), "centroids.dist" or "max.dist".
M	the number of folds in the M-fold cross-validation.
nrep	the number of repetitions of the whole procedure in the M-fold cross-validation.

Details

Leave-one-out cross-validation is not recommended with PLS-DA models.

When working on a LDA model, the prior probabilities for the models fitted on the training datasets are extracted from model (they are not evaluated from the training dataset itself).

Value

model	type of discriminant model used.
method	method used for cross-validation.
crit.lda	method for parameter estimation in LDA cross-validation.
crit.plsda	distance used in PLS-DA cross-validation.
M	number of folds in the M-fold cross-validation.
nrep	number of repetitions in the M-fold cross-validation.
tab	results of cross-validation.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

lda, predict.lda, plsda, perf

Examples

```
# Cross-validation of a LDA model
require(MASS)
data(iris)
model.LDA <- lda(iris[,1:4],iris$Species)
DA.valid(model.LDA,M=5,nrep=10)

# Cross-validation of a PLS-DA model
require(mixOmics)
data(yeast)
model.PLSDA <- plsda(t(yeast$data),yeast$strain.cond)
DA.valid(model.PLSDA,M=5,nrep=10)
```

DA.var *Explained variance in Discriminant Analysis*

Description

Calculate intergroup variance explained by components of a Discriminant Analysis (either obtained from lda or plsda).

Usage

```
DA.var(model)
```

Arguments

model	object of class "lda" (from lda) or "plsda" (from plsda).

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

lda, plsda

Examples

```
# LDA model
require(MASS)
data(iris)
model.LDA <- lda(iris[,1:4],iris$Species)
DA.var(model.LDA)

# PLS-DA model
require(mixOmics)
data(yeast)
model.PLSDA <- plsda(t(yeast$data),yeast$strain.cond)
DA.var(model.PLSDA)
```

dendro.gp *Dendrogram and number of groups to be chosen*

Description

Draw a dendrogram and an additional bar plot helping to choose the number of groups to be retained (based on the dendrogram).

Usage

```
dendro.gp(dend)
```

Arguments

dend	a dendrogram obtained from hclust.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

hclust

Examples

```
data(iris)
distances <- dist(iris[,1:4],method="euclidian")
dendro <- hclust(distances,method="ward")
dendro.gp(dendro)
```

---

fc.multcomp *Pairwise comparisons for the interaction between a factor and a covariate*

Description

Perform pairwise comparisons between groups, based on a model, for one factor or the interaction between a factor and a covariate. The function is based on glht, so deals with any model accepted by this function.

Usage

```
fc.multcomp(model, term = NULL, mat = NULL, p.method = "fdr")
```

Arguments

model	any model accepted by glht.
term	character string giving the term on which comparisons are based (a factor or the interaction between a factor and a covariate).
mat	matrix of contrasts (see 'Details'), either a matrix or a data frame.
p.method	method for p-values correction. See help of p.adjust.

Details

In matrices of contrasts, each line is a comparison (= contrast) and each colum is a level of the factor. Rules for writing contrasts are:

- levels not involved in the comparison must have a null value

- levels to be compared must have opposite signs

- levels can be grouped (for example 2 -1 -1 give a comparison of the first level against the group composed by the two others)

- the sum of all values of a contrast must be null.

Value

`method`	a character string giving the name of the test computed.
`model`	model call.
`p.value`	table of results.
`p.adjust.method`	method for p-values correction.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`glht`

Examples

```
# 'kidney' dataset of package 'survival'
require(survival)
require(multcomp)

model <- coxph(Surv(time,status)~age*disease+factor(sex),data=kidney)

# Default: all pairwise comparisons
fc.multcomp(model,"disease")
fc.multcomp(model,"age:disease")

# Dunnett contrasts
n <- n <- 1:nlevels(kidney$disease)
names(n) <- levels(kidney$disease)
mat <- contrMat(n)
fc.multcomp(model,"disease",mat)
```

`fisher.multcomp` *Pairwise comparisons after a test for independence of 2 categorical variables*

Description

Perform pairwise comparisons after a test for independence of 2 categorical variables, by using a Fisher's exact test on each possible 2x2 table.

Usage

```
fisher.multcomp(tab.cont, p.method = "fdr")
```

Arguments

`tab.cont`	contingency table.
`p.method`	method for p-values correction. See help of `p.adjust`.

Value

`method`	name of the test.
`data.name`	a character string giving the name(s) of the data.
`p.adjust.method`	method for p-values correction.
`p.value`	table of results of pairwise comparisons.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`chisq.test`, `fisher.test`

Examples

```
tab.cont <- as.table(matrix(c(25,10,12,6,15,14,9,16,9),ncol=3,dimnames=list(c("fair",
  "dark","russet"),c("blue","brown","green"))))
chisq.test(tab.cont)
fisher.multcomp(tab.cont)
```

`friedman.rating.test` *Friedman rank sum test for ratings*

Description

Wrapper for `friedman.test` with ratings (ordinal response variables).

Usage

```
friedman.rating.test(y, ...)

## Default S3 method:
friedman.rating.test(y, groups, blocks, ...)

## S3 method for class 'formula'
friedman.rating.test(formula, data, subset, na.action, ...)
```

Arguments

`y`	response variable (preferably an ordered factor).
`groups`	a vector giving the group for the corresponding element of `y`.
`blocks`	a vector giving the block for the corresponding element of `y`.
`formula`	a formula of the form `a ~ b | c`, where `a`, `b` and `c` give the data values and corresponding groups and blocks, respectively. It is preferable that `a` is an ordered factor.
`data`	an optional data frame containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used.
`na.action`	a function which indicates what should happen when the data contain `NA`s. Defaults to `getOption("na.action")`.
`...`	further arguments to be passed to or from other methods.

Value

See `friedman.test`

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
set.seed(1608)
response <- factor(sample(1:6,30,replace=TRUE),levels=c("1","2","3","4","5","6"),ordered=TRUE)
fact <- gl(3,1,30,labels=LETTERS[1:3])
block <- gl(10,3,labels=letters[1:10])
friedman.rating.test(response~fact|block)
```

`G.multcomp` *Pairwise comparisons after a G-test*

Description

Perform pairwise comparisons after a global G-test.

Usage

```
G.multcomp(x, p.method = "fdr")
```

Arguments

`x`	numeric vector (counts).
`p.method`	method for p-values correction. See help of `p.adjust`.

Value

`method`	name of the test.
`data.name`	a character string giving the name(s) of the data.
`p.adjust.method`	method for p-values correction.
`p.value`	table of results.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`G.test`

Examples

```
counts <- c(5,15,23,8,14)
G.test(counts)
G.multcomp(counts)
```

`G.test` *G-test*

Description

Perfom a G-test on a contingency table or a vector of counts.

Usage

```
G.test(x, p = rep(1/length(x), length(x)))
```

Arguments

`x`	a numeric vector or matrix (see Details).
`p`	theoretical proportions (optional).

Details

If `x` is matrix, it must be constructed like this:

- 2 columns giving number of successes (left) and fails (right)

- 1 row per population.

The function works as `chisq.test` :

- if `x` is a vector and theoretical proportions are not given, equality of counts is tested

- if `x` is a vector and theoretical proportions are given, equality of counts to theoretical counts (given by theoretical proportions) is tested

- if `x` is a matrix, equality of proportion of successes between populations is tested.

Value

`method`	name of the test.
`statistic`	test statistics.
`parameter`	test degrees of freedom
`p.value`	p-value.
`data.name`	a character string giving the name(s) of the data.
`observed`	the observed counts.
`expected`	the expected counts under the null hypothesis.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`chisq.test`, `G.multcomp`, `G.theo.multcomp`, `pairwise.G.test`

Examples

```
counts <- c(5,15,23,8,14)
G.test(counts)
```

`G.theo.multcomp` *Pairwise comparisons after a G-test for given probabilities*

Description

Perform pairwise comparisons after a global G-test for given probabilities.

Usage

```
G.theo.multcomp(x, p = rep(1/length(x), length(x)), p.method = "fdr")
```

Arguments

`x`	numeric vector (counts).
`p`	theoretical proportions.
`p.method`	method for p-values correction. See help of `p.adjust`.

Value

`method`	name of the test.
`data.name`	a character string giving the name(s) of the data.
`observed`	observed counts.
`expected`	expected counts.
`p.adjust.method`	method for p-values correction.
`statistic`	statistics of each test.
`p.value2`	corrected p-values.
`p.value`	data frame of results.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`G.test`

Examples

```
counts <- c(5,15,23,8,14)
p.theo <- c(0.1,0.4,0.3,0.15,0.05)
G.test(counts,p=p.theo)
G.theo.multcomp(counts,p=p.theo)
```

`ind.contrib` *Individual contributions in regression*

Description

Compute difference in regression parameters when each individual is dropped, expressed in proportion of the whole regression coefficients. The function deals with `lm` (including glm) and `least.rect` models.

Usage

```
ind.contrib(model, print.diff = FALSE, graph = TRUE, warning=25)
```

Arguments

`model`	model (of class `"lm"` or `"least.rect"`).
`print.diff`	logical. If `TRUE` results are printed.
`graph`	logical. If `TRUE` results are returned in a graphical way.
`warning`	level of graphical warning.

Value

`coefficients`	coefficients of each computed regression.
`coefficients.diff`	difference in coefficients between each computed regression and the whole regression.
`coefficients.prop`	difference in coefficients expressed in proportion of the whole regression coefficients.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`lm.influence`, `least.rect`

Examples

```
x <- 1:30
y <- 1:30+rnorm(30,0,4)
model1 <- lm(y~x)
model2 <- least.rect(y~x)
ind.contrib(model1)
ind.contrib(model2)
```

`kruskal.rating.test` *Kruskal-Wallis rank sum test for ratings*

Description

Wrapper for `kruskal.test` with ratings (ordinal response variables).

Usage

```
kruskal.rating.test(x, ...)

## Default S3 method:
kruskal.rating.test(x, g, ...)

## S3 method for class 'formula'
kruskal.rating.test(formula, data, subset, na.action, ...)
```

Arguments

`x`	response variable (preferably an ordered factor).
`g`	a factor giving the group for the corresponding element of `x`.
`formula`	a formula of the form `a ~ b`, where `a` and `b` give the data values and corresponding groups. It is preferable that `a` is an ordered factor.
`data`	an optional data frame containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used.
`na.action`	a function which indicates what should happen when the data contain `NA`s. Defaults to `getOption("na.action")`.
`...`	further arguments to be passed to or from other methods.

Value

See `kruskal.test`

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
set.seed(1609)
response <- factor(sample(1:6,30,replace=TRUE),levels=c("1","2","3","4","5","6"),ordered=TRUE)
fact <- gl(3,1,30,labels=LETTERS[1:3])
kruskal.rating.test(response~fact)
```

LDA.format *Re-formating LDA data*

Description

Re-format data obtained from lda, especially to get coordinates of individuals on factorial plans and correlation between original variables and LDA components.

Usage

```
LDA.format(model)
```

Arguments

model	object of class "lda" (from lda).

Value

x	original data (X variables only).
grouping	class of each individual.
li	coordinates of each individual on the LDA components.
co	correlation between original variables and LDA components.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

lda, s.class, s.corcircle

Examples

```
require(MASS)
require(ade4)
data(iris)
model.LDA <- lda(iris[,1:4],iris$Species)
coord <- LDA.format(model.LDA)
s.class(coord$li,fac=coord$grouping,col=rainbow(nlevels(coord$grouping)),cellipse=0)
s.corcircle(coord$co,label=abbreviate(rownames(coord$co),3))
```

`least.rect` *Least rectangles linear regression*

Description

Fit a least rectangle linear regression, possibly for each level of a factor.

Usage

```
least.rect(formula, data, conf.level = 0.95, theo = 1)
```

Arguments

`formula`	a formula of the form `y ~ x`, where `y` and `x` give the y and x variable, respectively. The formula can also be `y ~ x | f` to fit a (separate) regression for each level of the factor `f`.
`data`	an optional data frame containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`conf.level`	confidence level.
`theo`	theoretical value of the slope. If several regression are fitted, the same value is used for all comparisons of slope vs. theoretical value.

Value

`coefficients`	regression parameters.
`residuals`	residuals.
`fitted.values`	fitted values.
`call`	the matched call.
`model`	the model frame used.
`conf.level`	confidence level.
`conf.int`	confidence interval of regression parameters.
`theo`	theoretical value of the slope.
`comp`	data frame of results for equality of the slope(s) to the theoretical value.
`corr`	data frame of results for significativity of the correlation coefficient(s).
`multiple`	logical, `TRUE` if several regressions are fitted.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
x <- 1:30+rnorm(30,0,3)
y <- 1:30+rnorm(30,0,3)
regression1 <- least.rect(y~x)
summary(regression1)

x2 <- c(1:30,1:30)
y2 <- c(1:30+rnorm(30,0,3),seq(10,22,12/29)+rnorm(30,0,3))
fact <- gl(2,30,labels=LETTERS[1:2])
regression2 <- least.rect(y2~x2|fact)
summary(regression2)
```

`logis.fit` *Graphical adujstment of a simple binary logistic regression to data*

Description

Cut the data into intervals, compute the response probability and its standard error for each interval and add the results to the regression curve. No test is performed but this permits to have a graphical idea of the adjustment of the model to the data.

Usage

```
logis.fit(model, int = 5, ...)
```

Arguments

`model`	`glm` model.
`int`	number of intervals.
`...`	other arguments. See help of `points` and `segments`.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`glm`

Examples

```
x <- 1:50
y <- c(rep(0,18),sample(0:1,14,replace=TRUE),rep(1,18))
model <- glm(y~x,family=binomial)
plot(x,y)
lines(x,model$fitted)
logis.fit(model)
```

logis.noise — *Creating a nls model for logistic regression from fitted values of a glm model*

Description

Add some noise to the fitted values of a glm model to create a nls model for logistic regression (creating a nls model from exact fitted values can not be done, see help of nls).

Usage

```
logis.noise(model, intensity = 25)
```

Arguments

model	glm model.
intensity	intensity of the noise: lower the value, bigger the noise.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

glm, nls

Examples

```
x <- 1:50
y <- c(rep(0,18),sample(0:1,14,replace=TRUE),rep(1,18))
model <- glm(y~x,family=binomial)
y2 <- logis.noise(model)
# Then model2 <- nls(y2~SSlogis(...))
```

mod — *Mode*

Description

Compute the mode of a vector. The function makes the difference between continuous and discontinuous variables (which are made up of integers only). By extention, it also gives the most frequent value in a character vector or a factor.

Usage

```
mod(x)
```

Arguments

x	numeric vector.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

density

Examples

```
# Continuous variable
x <- rnorm(100)
mod(x)

# Discontinuous variable
y <- rpois(100,2)
mod(y)

# Character vector
z <- sample(LETTERS[1:3],20,replace=TRUE)
mod(z)
```

---

mqqnorm *Multivariate normality QQ-Plot*

Description

Draw a QQ-plot to assess multivariate normality.

Usage

```
mqqnorm(x, main = "Multi-normal Q-Q Plot")
```

Arguments

x	a data frame or a matrix of numeric variables (each column giving a variable).
main	title of the graph.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

mshapiro.test, qqPlot

Examples

```
x <- 1:30+rnorm(30)
y <- 1:30+rnorm(30,1,3)
mqqnorm(cbind(x,y))
```

`mshapiro.test` *Shapiro-Wilk multivariate normality test*

Description

Performs a Shapiro-Wilk test to asses multivariate normality. This is a slightly modified copy of the `mshapiro.test` function of the package mvnormtest, for internal convenience.

Usage

```
mshapiro.test(x)
```

Arguments

`x`	a data frame or a matrix of numeric variables (each column giving a variable).

Value

`method`	name of the test.
`data.name`	a character string giving the names of the data.
`statistic`	test statistics of the test.
`p.value`	p-value of the test.

Author(s)

Maxime Hervé <mx.herve@gmail.com> from the work of Slawomir Jarek

See Also

`shapiro.test`, `mshapiro.test`

Examples

```
x <- 1:30+rnorm(30)
y <- 1:30+rnorm(30,1,3)
mshapiro.test(cbind(x,y))
```

overdisp.glmer *Estimation of overdispersion with* glmer *models*

Description

Estimate residual deviance and residual degrees of freedom to check for overdispersion with glmer models. This function is directly comming from http://glmm.wikidot.com/faq.

Usage

```
overdisp.glmer(model)
```

Arguments

model	a model fitted by glmer.

Author(s)

Ben Bolker

See Also

glmer

Examples

```
require(lme4)

# Example from the 'glmer' function
gm1 <- glmer(cbind(incidence,size-incidence)~period+(1|herd),
 family="binomial",data=cbpp)
overdisp.glmer(gm1)
```

pairwise.G.test *Pairwise comparisons for proportions using G-tests*

Description

Perform pairwise comparisons between pairs of proportions with correction for multiple testing.

Usage

```
pairwise.G.test(x, p.method = "fdr")
```

Arguments

x	matrix with 2 columns giving the counts of successes and failures, respectively.
p.method	method for p-values correction. See help of p.adjust.

Value

method	name of the test.
data.name	a character string giving the name(s) of the data.
p.adjust.method	method for p-values correction.
p.value	table of results.

See Also

G.test

Examples

```
x <- matrix(c(22,28,13,37,35,15),ncol=2,dimnames=list(c("Control","Treatment1","Treatment2"),
  c("Alive","Dead")),byrow=TRUE)
G.test(x)
pairwise.G.test(x)
```

---

pairwise.manova *Pairwise MANOVAs*

Description

Perform pairwise comparisons between group levels with corrections for multiple testing.

Usage

```
pairwise.manova(resp, fact, p.method = "fdr")
```

Arguments

resp	response matrix (one column per variable). Object of class "data.frame" are accepted and internally converted into matrices.
fact	grouping factor.
p.method	method for p-values correction. See help of p.adjust.

Value

method	a character string giving the name of the test.
data.name	a character string giving the name(s) of the data.
p.value	table of results.
p.adjust.method	method for p-values correction.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

manova, summary.manova

Examples

```
data(iris)

# MANOVA
summary(manova(as.matrix(iris[,1:4])~iris$Species))

# Pairwise comparisons
pairwise.manova(iris[,1:4],iris$Species)
```

---

pairwise.perm.t.test *Pairwise permutational t tests*

Description

Perform pairwise comparisons between group levels with corrections for multiple testing.

Usage

```
pairwise.perm.t.test(resp, fact, p.method = "fdr", paired = FALSE,
  alternative = c("two.sided","less", "greater"), nperm = 999)
```

Arguments

resp	response vector.
fact	grouping factor.
p.method	method for p-values correction. See help of p.adjust.
paired	a logical indicating whether you want paired (permutational) t-tests.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
nperm	number of permutations.

Value

method	a character string indicating what type of t-tests were performed.
data.name	a character string giving the name(s) of the data.
p.value	table of results.
p.adjust.method	method for p-values correction.
permutations	number of permutations.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

pairwise.t.test

Examples

```
set.seed(1203)
response <- c(rnorm(5),rpois(5,0.5),rnorm(5,2,1))
fact <- gl(3,5,labels=LETTERS[1:3])

# Not enough permutations here but it runs faster

# Permutational ANOVA
perm.anova(response~fact,nperm=49)

# Pairwise comparisons
pairwise.perm.t.test(response,fact,nperm=49)
```

---

pairwise.wilcox.rating.test

Pairwise comparisons for ratings using Wilcoxon rank sum tests

---

Description

Perform pairwise comparisons for ratings (ordinal response variables) between group levels with correction for multiple testing.

Usage

```
pairwise.wilcox.rating.test(x, g, p.adjust.method = "fdr",
  paired = FALSE, ...)
```

Arguments

x	response vector (preferably an ordered factor).
g	grouping factor.
p.adjust.method	method for p-values correction. See help of p.adjust.
paired	a logical indicating whether you want a paired test.
...	additional arguments to be passed to wilcox.rating.test

Value

Object of class "pairwise.htest"

See Also

wilcox.rating.test

Examples

```
set.seed(1609)
response <- factor(sample(1:6,30,replace=TRUE),levels=c("1","2","3","4","5","6"),ordered=TRUE)
fact <- gl(3,1,30,labels=LETTERS[1:3])
kruskal.rating.test(response~fact)
pairwise.wilcox.rating.test(response,fact)
```

---

perm.anova *Permutational Analysis of Variance*

Description

Perform a permutational analysis of variance for 1 to 3 factors. For 2 and 3 factors, experiment design must be balanced. For 2 factors, the factors can be crossed with or without interaction, or nested. The second factor can be a blocking (random) factor. For 3 factors, design is restricted to 2 fixed factors crossed (with or without interaction) inside blocks (third factor).

Usage

```
perm.anova(formula, nest.f2 = c("fixed", "random"), data, nperm = 999)
```

Arguments

formula	a formula of the form `response ~ factor(s)` (see Details).
nest.f2	in case of 2 nested factors, precision is needed if the nested factor (factor2) is `"fixed"` (default) or `"random"`.
data	an optional data frame containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
nperm	number of permutations.

Details

For 2 factors, the formula can be:

`response ~ factor1 + factor2` for 2 fixed factors without interaction

`response ~ factor1 * factor2` for 2 fixed factors with interaction

`response ~ factor1 / factor2` for 2 fixed factors with factor2 nested into factor1 (if factor2 is a random factor, argument `nest.f2` must be changed from `"fixed"` (default) to `"random"`)

`response ~ factor1 | factor2` for 1 fixed factor (factor1) and 1 blocking (random) factor (factor2).

For 3 factors, the formula can only be:

`response ~ factor1 + factor2 | factor3` or

`response ~ factor1 * factor2 | factor3`. The 2 factors are here fixed and crossed inside each level of the third, blocking (random), factor.

Value

a data frame of class `"anova"`.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
set.seed(1203)
response <- c(rnorm(12),rpois(12,0.5),rnorm(12,2,1))
fact1 <- gl(3,12,labels=LETTERS[1:3])
fact2 <- gl(3,1,36,labels=letters[1:3])
fact3 <- gl(6,6,labels=letters[1:6])
block <- gl(2,6,36,labels=letters[1:2])

# Not enough permutations here but faster to run

# 2 crossed fixed factors with interaction
perm.anova(response~fact1*fact2,nperm=49)

# 2 nested fixed factors
perm.anova(response~fact1/fact2,nperm=49)

# 2 nested factors, fact2 being random
perm.anova(response~fact1/fact3,nest.f2="random",nperm=49)

# 1 fixed factor and 1 blocking (random) factor
perm.anova(response~fact1|block,nperm=49)
```

---

perm.bartlett.test *Permutational Bartlett's test of homogeneity of variances*

Description

Perform a permutational Bartlett's test of homogeneity of k variances.

Usage

```
perm.bartlett.test(formula, data, nperm = 999)
```

Arguments

formula	a formula of the form a ~ b where a gives the data values and b the corresponding groups.
data	an optional data frame containing the variables in the formula formula. By default the variables are taken from environment(formula).
nperm	number of permutations.

Value

method	name of the test.
data.name	a character string giving the name(s) of the data.
statistic	test statistics of the parametric test.
permutations	number of permutations.
p.value	p-value of the permutational test.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

bartlett.test

Examples

```
response <- c(rnorm(12),rpois(12,1),rnorm(12,2,1))
fact <- gl(3,12,labels=LETTERS[1:3])
perm.bartlett.test(response~fact)
```

---

perm.cor.test *Permutational Pearson's correlation test*

Description

Perform a permutational Pearson's product-moment correlation test.

Usage

```
perm.cor.test(x, y, alternative = c("two.sided", "less", "greater"), nperm = 999)
```

Arguments

x, y	numeric vectors of data values. x and y must have the same length.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
nperm	number of permutations.

Value

method	name of the test.
data.name	a character string giving the name(s) of the data.
statistic	test statistics of the parametric test.
permutations	number of permutations.
p.value	p-value of the permutational test.
estimate	the estimated correlation coefficient.
alternative	a character string describing the alternative hypothesis.
null.value	the value of the association measure under the null hypothesis, always 0.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

cor.test

Examples

```
x <- rnorm(50)
y <- runif(50)
perm.cor.test(x,y)
```

---

perm.t.test *Permutational Student's t-test*

Description

Perform a permutational Student's t-test.

Usage

```
perm.t.test(formula, data, alternative = c("two.sided", "less", "greater"),
  paired = FALSE, nperm = 999)
```

Arguments

formula	a formula of the form a ~ b where a gives the data values and b a factor with 2 levels giving the corresponding groups.
data	an optional data frame containing the variables in the formula formula. By default the variables are taken from environment(formula).
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
paired	a logical indicating whether you want a paired t-test.
nperm	number of permutations.

Value

statistic	test statistics of the parametric test.
permutations	number of permutations.
p.value	p-value of the permutational test.
estimate	the estimated mean or difference in means depending on whether it was a paired or not paired test.
alternative	a character string describing the alternative hypothesis.
method	a character string indicating what type of t-test was performed.
data.name	a character string giving the name(s) of the data.
null.value	the specified hypothesized value of the mean difference, always 0.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

t.test

Examples

```
response <- c(rnorm(5),rnorm(5,2,1))
fact <- gl(2,5,labels=LETTERS[1:2])

# Unpaired test
perm.t.test(response~fact)

# Paired test
perm.t.test(response~fact,paired=TRUE)
```

---

perm.var.test *Permutational F test to compare two variances*

Description

Perform a permutational F test to compare two variances.

Usage

```
perm.var.test(formula, data, ratio = 1, alternative = c("two.sided", "less",
  "greater"), nperm = 999)
```

Arguments

formula	a formula of the form a ~ b where a gives the data values and b a factor with 2 levels giving the corresponding groups.
data	an optional data frame containing the variables in the formula formula. By default the variables are taken from environment(formula).
ratio	the hypothesized ratio of the two population variances.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
nperm	number of permutations.

Value

method	name of the test.
statistic	test statistics of the parametric test.
permutations	number of permutations.
p.value	p-value of the permutational test.
estimate	the ratio of the two variances.
alternative	a character string describing the alternative hypothesis.
data.name	a character string giving the name(s) of the data.
null.value	the ratio of population variances under the null hypothesis.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

var.test

Examples

```
response <- c(rpois(8,1),rpois(8,3))
fact <- gl(2,8,labels=LETTERS[1:2])
perm.var.test(response~fact)
```

plot1comp.ind *Plot of indiviuals and their class on a single axis (component)*

Description

Represent indivual points on a single axis (usually a factorial component) and add their class, respecting to a given factor.

Usage

```
plot1comp.ind(dfxy, fac, axis = 1, col = 1:nlevels(fac), cex = 1, box = TRUE)
```

Arguments

dfxy	a data frame containing coordinates of individuals on the axi(e)s.
fac	a factor giving the class of each individual.
axis	axis to be represented (the first by default).
col	a vector of colors (one value per class of the factor).
cex	size of class names.
box	logical, should a box to be drawn around class names?

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

s.class

Examples

```
require(MASS)
data(iris)
LDA <- lda(iris[,1:4],iris$Species)
LDA.form <- LDA.format(LDA)
plot1comp.ind(LDA.form$li,LDA.form$grouping,col=1:3)
```

plot1comp.var — *Plot of correlation between predictor variables and a single component*

Description

Represent correlations between predictor variables and a single axis (usually a factorial component).

Usage

```
plot1comp.var(dfxy, axis = 1, cex = 1)
```

Arguments

dfxy	a data frame containing correlation coefficients between predictor variables and axi(e)s.
axis	axis to be represented (the first by default).
cex	size of variables names.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

s.corcircle

Examples

```
require(MASS)
data(iris)
LDA <- lda(iris[,1:4],iris$Species)
LDA.form <- LDA.format(LDA)
plot1comp.var(LDA.form$co)
```

plotresid — *Simple analysis of model residuals*

Description

Plot residuals of a model against fitted values and a QQ-plot of these residuals. Optionally, a Shapiro-Wilk test can be performed on residuals. The function deals with lm (including glm, glm.nb and manova), lmer, glmer, glmmadmb, lme, nls, nlsList, survreg and least.rect models.

Usage

```
plotresid(model, shapiro = FALSE)
```

Arguments

model	an object of class "lm", "merMod", "glmmadmb", "lme", "nls", "nlsList", "survreg" or "least.rect".
shapiro	logical. If TRUE a Shapiro-Wilk test is performed on residuals.

Details

Externally studentized resisuals are used for lm, glm (except with a poisson, quasipoisson, binomial or quasibinomial family). Standardized residuals are used for nlme, nls and nlsList models. Quantile residuals (function qresiduals) are used for glm.nb, glm (with a poisson, quasipoisson, binomial or quasibinomial family), glmer (with a poisson or binomial family) and glmer.nb models. In all other cases raw residuals are used.

With a manova model, only a multivariate QQ-plot is drawn. The test performed when shapiro=TRUE is a Shapiro-Wilk test for multivariate normality.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

lm, glm, glm.nb, manova, lmer, glmer, lmer, glmer.nb, lme, nls, nlsList, survreg, least.rect, qresiduals, qqPlot, shapiro.test, mqqnorm, mshapiro.test

---

plotsurvivors *Survivor curve*

Description

Plot the survivor curve (log(survivors) against time) of a dataset to check for constancy of hazard.

Usage

```
plotsurvivors(x, status = rep(1, length(x)))
```

Arguments

x	time to event.
status	status (1: event observed, 0: event not observed).

Value

n	initial number of individuals.
time	time of events.
alive	number of survivors at each time.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
# 'kidney' dataset of package 'survival'
require(survival)
data(kidney)
plotsurvivors(kidney$time,kidney$status)
```

PLSDA.ncomp *Graphical help to number of components selection in PLS-DA*

Description

Draw a graph with classification error rate (obtained from cross-validation with the M-fold method) and proportion of intergroup variance explained from one to any possible components included in a given PLS-DA model. This helps to choose the number of components to be kept in the model.

Usage

```
PLSDA.ncomp(X, Y, pred.method = c("mahalanobis.dist", "centroids.dist", "max.dist"),
  M = 10, nrep = 10)
```

Arguments

X	numeric matrix of predictors.
Y	a factor giving the class of each individual.
pred.method	prediction method to be applied for PLS-DA cross-validation. Should be a subset of "mahalanobis.dist" (default), "centroids.dist" or "max.dist".
M	the number of folds in the M-fold cross-validation.
nrep	the number of repetitions of the whole procedure in the M-fold cross-validation.

Details

The function emphasizes values obtained with nlev - 1 groups, with nlev being the number of levels of the factor, which is often the optimal number of components to be kept.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

plsda, perf, DA.valid, DA.var

Examples

```
require(mixOmics)
data(yeast)
X <- t(yeast$data)
Y <- yeast$strain.cond
PLSDA.ncomp(X,Y,M=5,nrep=3)
```

PLSDA.test *Permutational test for the discriminant ability of the factor in PLS-DA*

Description

Perform a permutational test based on the classification error rate of a PLS-DA model, to test for the discriminant ability of the factor.

Usage

```
PLSDA.test(model, pred.method = c("mahalanobis.dist", "centroids.dist", "max.dist"),
  M = 10, nperm = 999)
```

Arguments

model	object of class "plsda" (from plsda).
pred.method	prediction method to be applied for PLS-DA cross-validation. Should be a subset of "mahalanobis.dist" (default), "centroids.dist" or "max.dist".
M	the number of folds in the M-fold cross-validation.
nperm	number of permutations.

Details

Data have to be balanced to perform the test. The function automatically takes care of it by taking equal number of individuals in each class of the factor, randomly chosen for each class and in each permutation. The number of individuals kept is equal to the number of individuals in the smallest class.

To compare classification error rates of the original and permuted models, classification error rate of the original model must be obtained with balanced data. The function does this by randomly choosing equal number of individuals in each class (as in the permutation process). However, to avoid basing the test on an original classification error rate obtained from non representative individuals, the process is repeated 30 times and the mean of these 30 values is considered as the classification error rate of the original model.

Value

method	a character string giving the name of the test.
data.name	a character string giving the name(s) of the data and parameters of the cross-validation.
p.value	p-value of the permutational test.
alternative	a character string describing the alternative hypothesis, always "greater".
null.value	the classification error rate obtained from the original model.
estimate	the classification error rate obtained from the original model.
pred.method	distance criterion used in cross-validation.
M	number of folds in the M-fold cross-validation.
permutations	number of permutations.

Warning

The process is quite long and takes several minutes to be achieved. Just don't worry.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

plsda, perf

Examples

```
# Not enough permutations here but it runs faster.

require(mixOmics)
data(yeast)
X <- t(yeast$data)
Y <- yeast$strain.cond
model <- plsda(X,Y)
## Not run: PLSDA.test(model,M=3,nperm=29)
```

PLSDA.VIP *Variable Importance in the Projection (VIP)*

Description

Return VIP score of each X-variable in a PLS-DA model (obtained from plsda).

Usage

```
PLSDA.VIP(model, graph = FALSE)
```

Arguments

model	object of class "plsda" (from plsda).
graph	logical: should VIP scores be printed?

Value

tab	table of results.
sup1	name of X-variables having a VIP score > 1.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

plsda

Examples

```
require(mixOmics)
data(yeast)
model.PLSDA <- plsda(t(yeast$data),yeast$strain.cond)
PLSDA.VIP(model.PLSDA)
```

---

prop.multcomp *Pairwise comparisons after a test for given proportions*

Description

Perform pairwise comparisons after a global test for given proportions, by using exact binomial tests.

Usage

```
prop.multcomp(x, p, p.method = "fdr")
```

Arguments

x	conitngency table.
p	theoretical proportions.
p.method	method for p-values correction. See help of p.adjust.

Value

method	name of the test.
data.name	a character string giving the name(s) of the data.
observed	observed proportions.
expected	expected proportions.
p.adjust.method	method for p-values correction.
p.value2	corrected p-values.
p.value	table or results of pairwise comparisons.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

prop.test

Examples

```
proportions <- sample(c(0,1),60,replace=TRUE)
populations <- sample(LETTERS[1:3],60,replace=TRUE)
tab.cont <- table(populations,proportions)
p.theo <- c(0.2,0.5,0.7)
prop.test(tab.cont,p=p.theo)
prop.multcomp(tab.cont,p=p.theo)
```

---

rating.lsmeans *LSMeans for Cumulative Link (Mixed) Models*

Description

Extract LSMeans (produced by lsmeans) from Cumulative Link (Mixed) Models (produced by clm or clmm), with different possible formats.

Usage

```
rating.lsmeans(lsm, type = c("prob", "cumprob", "class1", "class2"), level = 0.9)
```

Arguments

lsm	object returned by lsmeans applied on a clm or clmm object. A factor named cut must have been called in lsmeans, to compute LSMeans per cut point (i.e. rating). This is typically done by using lsmeans(model,~factor|cut) where factor is the factor (or interaction) giving levels for which LSMeans have to be computed.
type	type of output to be returned: "prob" (default) gives probability of each rating, "cumprob" gives cumulative probabilities (Pi is probability to be <= to rating i), "class1" gives the most probable rating and "class2" gives the first rating for which the cumulative probability is >= to level.
level	used only for type "class2" (see type).

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

lsmeans, clm, clmm

Examples

```
require(ordinal)
require(lsmeans)

model <- clm(rating~contact*temp,data=wine)
LSM <- lsmeans(model,~contact:temp|cut)

# Probabilities
rating.lsmeans(LSM)

# Cumulative probabilities
rating.lsmeans(LSM,type="cumprob")

# Most probable rating
rating.lsmeans(LSM,type="class1")
```

---

rating.prob *Observed rating frequencies*

Description

Compute observed rating frequencies per level of a factor, in various formats.

Usage

```
rating.prob(x, g, type = c("prob", "cumprob", "class"))
```

Arguments

x	ordered factor (ratings).
g	factor giving groups to be compared.
type	type of output to be returned: "prob" (default) gives frequency of each rating, "cumprob" gives cumulative frequencies (Fi is frequence of ratings <= i) and "class" gives the most frequent rating.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
require(ordinal)
data(wine)

# Frequencies
rating.prob(wine$rating,wine$contact:wine$temp)

# Cumulative frequencies
rating.prob(wine$rating,wine$contact:wine$temp,type="cumprob")

# Most frequent rating
rating.prob(wine$rating,wine$contact:wine$temp,type="class")
```

---

reg.ci *Confidence intervals of a simple linear regression*

Description

Compute and add to a graph the confidence interval of a simple regression line or of individual values.

Usage

```
reg.ci(model, conf.level = 0.95, type = c("mean", "ind"), ...)
```

Arguments

model	lm model.
conf.level	confidence level.
type	interval type : "mean" for the interval of the regression line (default), "ind" for the interval of individual values (also called "prediction interval").
...	other agruments. See help of lines.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

lm

Examples

```
x <- 1:50
y <- 1:50+rnorm(50,0,4)
regression <- lm(y~x)
plot(x,y)
abline(regression)
reg.ci(regression,type="mean",col="red")
reg.ci(regression,type="ind",col="blue")
```

---

s.corcircle2 *Plot of the factorial maps of a correlation circle*

Description

Performs the scatter diagram of a correlation circle. The function is a copy of s.corcircle including 3 new arguments: boxes, col and lwd.

Usage

```
s.corcircle2(dfxy, xax = 1, yax = 2, label = row.names(df),
  clabel = rep(1,nrow(dfxy)), grid = TRUE, sub = "", csub = 1,
  possub = "bottomleft", cgrid = 0, fullcircle = TRUE, box = FALSE,
  add.plot = FALSE, boxes = FALSE, col = rep("black", nrow(dfxy)),
 lwd = rep(1,nrow(dfxy)))
```

Arguments

dfxy	a data frame with two coordinates.
xax	the column number for the x-axis (horizontal).
yax	the column number for the y-axis (vertical).
label	a vector of strings of characters for the point labels.
clabel	if not NULL, a character size for the labels, used with par("cex")*clabel. Can be a unique value or one value per label.
grid	a logical value indicating whether a grid in the background of the plot should be drawn.
sub	a string of characters to be inserted as legend.
csub	a character size for the legend, used with par("cex")*csub.

possub	a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright").
cgrid	a character size, parameter used with par("cex")*cgrid to indicate the mesh of the grid.
fullcircle	a logical value indicating whether the complete circle sould be drawn.
box	a logical value indcating whether a box should be drawn.
add.plot	if TRUE uses the current graphics window.
boxes	a logical value indcating whether a box should be drawn around labels.
col	a vector of colors (unique value or one value per arrow).
lwd	a vector of line widths (unique value or one value per arrow).

Author(s)

Maxime Hervé <mx.herve@gmail.com>, based on a function of Stéphane Champely

See Also

s.corcircle

Examples

```
require(ade4)
data(olympic)

pca <- dudi.pca(olympic$tab,scannf=FALSE,nf=2)
col <- c("red","red","blue","grey","red","red","blue","grey","blue","grey")
s.corcircle2(pca$co,col=col)
```

---

scat.mix.categorical *Representation of qualitative variables in Hill and Smith mix analysis*

Description

Represent qualitative variables of a Hill and Smith mix analysis as in MCA. The function is based on s.class.

Usage

```
scat.mix.categorical(dudi.obj, xax = 1, yax = 2, csub = 2, possub = "topleft", ...)
```

Arguments

dudi.obj	dudi.mix object.
xax	horizontal axis.
yax	vertical axis.
csub	title size.
possub	title position ("topleft", "topright", "bottomleft" or "bottomright").
...	additional arguments to s.class. See help of this function.

Author(s)

Maxime Hervé <mx.herve@gmail.com>, based on an idea of Stéphane Champely.

References

Champely, S. 2005 Introduction à l'analyse multivariée (factorielle) sous R.

See Also

dudi.mix, s.class

Examples

```
require(ade4)
# Fictive dataset
age <- sample(15:60,50,replace=TRUE)
sex <- sample(c("M","F"),50,replace=TRUE)
size <- sample(155:190,50,replace=TRUE)
hair <- sample(c("Fair","Dark","Russet"),50,replace=TRUE)
eyes <- sample(c("Blue","Green","Brown"),50,replace=TRUE)
weight <- sample(50:85,50,replace=TRUE)
hand <- sample(c("Left.handed","Right.handed"),50,replace=TRUE)
tab <- data.frame(age,sex,size,weight,hand,eyes,hair)
amix <- dudi.hillsmith(tab,scannf=FALSE,nf=2)
scat.mix.categorical(amix)
```

---

scat.mix.numeric *Representation of numeric variables in Hill and Smith mix analysis*

Description

Represent numeric variables of a Hill and Smith mix analysis as in PCA. The function is based on s.corcircle.

Usage

```
scat.mix.numeric(obj.dudi, xax = 1, yax = 2, ...)
```

Arguments

obj.dudi	dudi.mix object.
xax	horizontal axis.
yax	vertical axis.
...	additional arguments to s.corcircle. See help of this function.

Author(s)

Stéphane Champely

References

Champely, S. 2005 Introduction à l'analyse multivariée (factorielle) sous R.

See Also

dudi.mix, s.corcircle

Examples

```
require(ade4)
# Fictive dataset
age <- sample(15:60,50,replace=TRUE)
sex <- sample(c("M","F"),50,replace=TRUE)
size <- sample(155:190,50,replace=TRUE)
hair <- sample(c("Fair","Dark","Russet"),50,replace=TRUE)
eyes <- sample(c("Blue","Green","Brown"),50,replace=TRUE)
weight <- sample(50:85,50,replace=TRUE)
hand <- sample(c("Left.handed","Right.handed"),50,replace=TRUE)
tab <- data.frame(age,sex,size,weight,hand,eyes,hair)
amix <- dudi.hillsmith(tab,scannf=FALSE,nf=2)
scat.mix.numeric(amix)
```

---

se *Standard error of the mean*

Description

Compute the standard error of the mean of a vector.

Usage

```
se(x)
```

Arguments

x	numeric vector.

Details

The function deals with missing values.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
se(rnorm(30))
```

seq2 *Sequence generation*

Description

Generate a regular sequence from the minimum to the maximum of a vector.

Usage

```
seq2(x, int = 999)
```

Arguments

x	numeric vector.
int	number of values to be generated (int breaks).

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

seq

Examples

```
seq2(rnorm(30))
```

spearman.ci *Confidence interval of a Spearman's rank correlation coefficient*

Description

Compute the confidence interval of a Spearman's rank correlation coefficient by bootstraping.

Usage

```
spearman.ci(var1, var2, nrep = 1000, conf.level = 0.95)
```

Arguments

var1	numeric vector (first variable).
var2	nuermic verctor (second variable).
nrep	number of replicates for bootstraping.
conf.level	confidence level.

Value

method	name of the test.
data.name	a character string giving the name(s) of the data.
conf.level	confidence level.
rep	number of replicates.
estimate	Spearman's rank correlation coefficient.
conf.int	confidence interval.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

cor.test, boot

Examples

```
var1 <- sample(1:50,15,replace=TRUE)
var2 <- sample(1:50,15,replace=TRUE)
spearman.ci(var1,var2)
```

---

user.cont *User defined contrasts for LSMeans*

Description

Return a function usable by lsmeans for user defined contrasts.

Usage

```
user.cont(cont)
```

Arguments

cont	any matrix of contrasts (see 'Details').

Details

In these matrices, each line is a comparison (= contrast) and each colum is a level of the factor. Rules for writing contrasts are:

- levels not involved in the comparison must have a null value

- levels to be compared must have opposite signs

- levels can be grouped (for example 2 -1 -1 give a comparison of the first level against the group composed by the two others)

- the sum of all values of a contrast must be null.

Value

`user.cont.lsmc` the function to be called by `lsmeans`

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`lsmeans`

Examples

```
require(car)
require(lsmeans)

response <- c(rpois(30,1),rpois(30,3),rpois(30,10))
fact <- gl(3,30,labels=LETTERS[1:3])
model <- glm(response~fact,family="poisson")
Anova(model)
mat <- matrix(c(1,-1,0,0,1,-1,2,-1,-1),nrow=3,byrow=TRUE,dimnames=list(levels(fact),1:3))
mat
cont.lsmc <- user.cont(mat)
lsmeans(model,cont~fact)
```

---

wilcox.paired.multcomp

Non parametric pairwise comparisons for paired data

Description

Perform non parametric pairwise comparisons of paired samples by Wilcoxon signed rank tests for paired data.

Usage

```
wilcox.paired.multcomp(formula, data, p.method = "fdr")
```

Arguments

`formula`	a formula of the form `a ~ b | c`, where `a`, `b` and `c` give the data values and corresponding groups and blocks, respectively.
`data`	an optional data frame containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`p.method`	method for p-values correction. See help of `p.adjust`.

Value

method	name of the test.
data.name	a character string giving the name(s) of the data.
method	a character string indicating the name of the test.
p.adjust.method	method for p-values correction.
p.value	table of results of pairwise comparisons.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

pairwise.wilcox.test, wilcox.test

Examples

```
response <- c(rnorm(10,0,3),rnorm(10,5,3),rnorm(10,8,2))
fact <- gl(3,10,labels=LETTERS[1:3])
block <- gl(10,1,30,labels=letters[1:10])
friedman.test(response~fact|block)
wilcox.paired.multcomp(response~fact|block)
```

---

wilcox.paired.rating.multcomp

Non parametric pairwise comparisons for paired ratings

Description

Perform non parametric pairwise comparisons of paired ratings (ordinal response variables) by Wilcoxon signed rank tests for paired data.

Usage

```
wilcox.paired.rating.multcomp(formula, data, p.method = "fdr")
```

Arguments

formula	a formula of the form `a ~ b	c`, where `a`, `b` and `c` give the data values and corresponding groups and blocks, respectively. It is preferable that `a` is an ordered factor.
data	an optional data frame containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
p.method	method for p-values correction. See help of `p.adjust`.

Value

`method`	name of the test.
`data.name`	a character string giving the name(s) of the data.
`method`	a character string indicating the name of the test.
`p.adjust.method`	method for p-values correction.
`p.value`	table of results of pairwise comparisons.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`pairwise.wilcox.rating.test`, `wilcox.rating.test`

Examples

```
set.seed(1630)
response <- factor(c(sample(1:3,10,replace=TRUE),
  sample(2:5,10,replace=TRUE),
  sample(3:6,10,replace=TRUE)),levels=c("1","2","3","4","5","6"),ordered=TRUE)
fact <- gl(3,10,labels=LETTERS[1:3])
block <- gl(10,1,30,labels=letters[1:10])
friedman.rating.test(response~fact|block)
wilcox.paired.rating.multcomp(response~fact|block)
```

`wilcox.rating.signtest` *Wilcoxon sign test for ratings*

Description

Wrapper for `wilcox.signtest` with ratings (ordinal response variables).

Usage

```
wilcox.rating.signtest(x, data, mu = NULL)
```

Arguments

`x`	either a formula of the form `a ~ b` where `a` gives the data values and `b` a factor with 2 levels giving the corresponding groups (comparison of two median ratings); or a numeric vector (comparison of one median rating to a given value). It is preferable that `a` (or `x`) is an ordered factor.

data	an optional data frame containing the variables in the formula x (if x is a formula). By default the variables are taken from environment(x). Not used if x is a numeric vector.
mu	theoretical median rating.

Value

See wilcox.signtest

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

wilcox.rating.test

Examples

```
set.seed(1706)
response <- factor(c(sample(1:4,7,replace=TRUE),sample(3:6,7,replace=TRUE)),
  levels=c("1","2","3","4","5","6"),ordered=TRUE)

# Comparison of 2 samples
fact <- gl(2,7,labels=LETTERS[1:2])
wilcox.rating.signtest(response~fact)

# Comparison to a given value
theo <- "4"
wilcox.rating.signtest(response,mu=theo)
```

---

wilcox.rating.test *Wilcoxon rank sum and signed rank tests for ratings*

Description

Wrapper for wilcox.test with ratings (ordinal response variables).

Usage

```
wilcox.rating.test(x, ...)

## Default S3 method:
wilcox.rating.test(x, y = NULL, mu = NULL, ...)

## S3 method for class 'formula'
wilcox.rating.test(formula, data, subset, na.action, ...)
```

Arguments

x	response variable (preferably an ordered factor).
y	an optional second response variable (preferably an ordered factor) to be compared with x.
mu	a character string giving a theoretical rating to be compared with x in one-sample tests.
formula	a formula of the form a ~ b, where a and b give the data values and corresponding groups. It is preferable that a is an ordered factor.
data	an optional data frame containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
...	further arguments to be passed to or from other methods.

Value

See wilcox.test

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
set.seed(1719)
response <- factor(sample(1:6,30,replace=TRUE),levels=c("1","2","3","4","5","6"),ordered=TRUE)
fact <- gl(2,15,labels=LETTERS[1:2])
wilcox.rating.test(response~fact)
```

---

wilcox.signtest *Wilcoxon sign test*

Description

Perform a Wilcoxon sign test to compare medians of two small paired samples or one median to a given value.

Usage

```
wilcox.signtest(x, data, mu = 0)
```

Arguments

x	either a formula of the form a ~ b where a gives the data values and b a factor with 2 levels giving the corresponding groups (comparison of two medians); or a numeric vector (comparison of one median to a given value).
data	an optional data frame containing the variables in the formula x (if x is a formula). By default the variables are taken from environment(x). Not used if x is a numeric vector.
mu	theoretical median.

Value

method	a character string indicating what type of test was performed.
data.name	a character string giving the name(s) of the data.
null.value	the specified hypothesized value of the median or median difference depending on the test performed.
p.value	p-value of the test.
estimate	the estimated median or difference in medians depending on the test performed.
alternative	a character string giving the alternative hypothesis, always "two.sided"

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

wilcox.test

Examples

```
set.seed(1706)
response <- c(rnorm(7,3,1.5),rnorm(7,5.5,2))

# Comparison of 2 samples
fact <- gl(2,7,labels=LETTERS[1:2])
wilcox.signtest(response~fact)

# Comparison to a given value
theo <- 4
wilcox.signtest(response,mu=theo)
```

Index