Package 'RVAideMemoire'

November 15, 2015

Encoding latin1

Type Package

Title Diverse Basic Statistical and Graphical Functions

Version 0.9-52

Date 2015-11-15

Author Maxime Hervé

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

Imports ade4, boot, car, cramer, graphics, grDevices, lme4 (>= 1.0-4), MASS, mixOmics (>= 5.0.2), multcompView, nnet, pls, pspearman, statmod, stats, utils, vegan

Suggests EMT, lsmeans, npsm, ordinal, survival

Description 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 2015-11-15 22:26:58

R topics documented:

`RVAideMemoire-package` *Diverse Basic Statistical and Graphical Functions*

Description

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:	0.9-52
Date:	2015-11-15
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)

`adonis.II` *Type II permutational MANOVA using distance matrices*

Description

This function is a wrapper to `adonis` but performs type II tests (whereas `adonis` performs type I).

Usage

```
adonis.II(formula, data, ...)
```

Arguments

`formula`	a typical model formula such as `Y~A+B*C`, but where `Y` is either a dissimilarity object (inheriting from class `"dist"`) or data frame or a matrix; `A`, `B`, and `C` may be factors or continuous variables.
`data`	the data frame from which `A`, `B` and `C` would be drawn.
`...`	additional arguments to `adonis`. See help of this function.

Details

See `adonis` for detailed explanation of what is done. The only difference with `adonis` is that `adonis.II` performs type II tests instead of type I.

Value

a data frame of class "anova".

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
require(vegan)
data(dune)
data(dune.env)

# Compare:
adonis(dune~Management*A1,data=dune.env,permutations=99)
adonis(dune~A1*Management,data=dune.env,permutations=99)

# With:
adonis.II(dune~Management*A1,data=dune.env,permutations=99)
adonis.II(dune~A1*Management,data=dune.env,permutations=99)
```

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

`back.lsmeans` *Back-transformation of LSMeans*

Description

Back-transforms LSMeans (produced by lsmeans) when the model was built on a transformed response variable. This is typically the case when a LM(M) with *log(x+1)* as response variable gives a better fitting than a GLM(M) for count data.

Usage

```
back.lsmeans(lsm, transform = c("log", "logit", "sqrt", "inverse"), base = exp(1),
  add = 0, ord = FALSE, decreasing = TRUE)
```

Arguments

`lsm`	object returned by lsmeans.
`transform`	transformation applied to the response variable before building the model on which `lsm` is based.
`base`	the base with respect to which the logarithm transformation was computed (if `transform="log"`). Defaults to *e*=`exp(1)`.
`add`	value to be added to *x* before computing the transformation, if needed (*e.g.* `1` if the initial transformation was *log(x+1)*).
`ord`	logical indicating if back-transformed LSMeans should be ordered.
`decreasing`	logical indicating in which order back-transformed LSMeans should be ordered, if `order=TRUE`.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

lsmeans

Examples

```
require(lsmeans)

set.seed(1149)
response <- c(rpois(30,0),rpois(30,2),rpois(30,4))
fact <- gl(3,30,labels=LETTERS[1:3])

model <- lm(log(response+1)~fact)
LSM <- lsmeans(model,~fact)
back.lsmeans(LSM,transform="log",add=1)
```

---

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

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

Usage

```
byf.hist(formula, data, sep = FALSE, density = TRUE, xlab = NULL, ylab = NULL)
```

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).
sep	logical. If TRUE a histogram is displayed per level of the factor. If FALSE all levels are displayed on the same histogram.
density	logical. If TRUE density polygons are displayed, if FALSE classical counts are displayed.
xlab	label for x-axis (name of the response variable as default).
ylab	label for y-axis ("Density" or "Frequency" as default, depending on the type of histogram).

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

Draws 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

Performs 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.qqnorm *QQ-plot for factor levels*

Description

Draws 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

Performs 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.bin.exp *Expected counts for comparison of response probabilities to given values*

Description

Returns expected counts before comparing response probabilities (i.e. when the response variable is a binary variable) to given values by a chi-squared test. The function is in fact a wrapper to the chi-squared test for comparison of proportions to given values on a contingency table.

Usage

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

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).
p	theoretical probabilities.
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 probabilities.
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.theo.bintest, mosaicplot

Examples

```
response <- c(rep(0:1,c(40,60)),rep(0:1,c(55,45)),rep(0:1,c(65,35)))
fact <- gl(3,100,labels=LETTERS[1:3])
p.theo <- c(0.5,0.45,0.2)
chisq.bin.exp(response~fact,p=p.theo)
```

---

chisq.bintest *Pearson's Chi-squared test for binary variables*

Description

Performs a Pearson's Chi-squared test for comparing response probabilities (i.e. when the response variable is a binary variable). 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 significant, the function performs pairwise comparisons by using Pearson's Chi-squared tests.

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.

Details

If the response is a 0/1 variable, the probability of the '1' group is tested. In any other cases, the response is transformed into a factor and the probability of the second level is tested.

Since a chi-squared test is an approximate test, an exact test is preferable when the number of individuals is small (200 is a reasonable minimum). See `fisher.bintest` in that case.

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.
`parameter`	test degrees of freedom.
`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>

See Also

`G.bintest`, `fisher.bintest`

Examples

```
response <- c(rep(0:1,c(40,60)),rep(0:1,c(55,45)),rep(0:1,c(65,35)))
fact <- gl(3,100,labels=LETTERS[1:3])
chisq.bintest(response~fact)
```

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

Description

Returns 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),200,replace=TRUE)
populations <- sample(LETTERS[1:3],200,replace=TRUE)
tab.cont <- table(populations,proportions)
p.theo <- c(0.4,0.5,0.7)
chisq.exp(tab.cont,p=p.theo)
```

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

Description

Performs 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`.

Details

Since a chi-squared test is an approximate test, an exact test is preferable when the number of individuals is small (200 is a reasonable minimum). See `multinomial.multcomp` in that case.

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`, `multinomial.test`, `multinomial.multcomp`

Examples

```
counts <- c(49,30,63,59)
chisq.test(counts)
chisq.multcomp(counts)
```

`chisq.theo.bintest` *Pearson's Chi-squared test for comparison of response probabilities to given values*

Description

Performs a Pearson's Chi-squared test for comparing response probabilities (i.e. when the response variable is a binary variable) to given values. The function is in fact a wrapper to the chi-squared test for comparison of proportions to given values on a contingency table.

Usage

```
chisq.theo.bintest(formula, data, p)
```

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)`.
`p`	theoretical probabilities.

Details

If the response is a 0/1 variable, the probability of the '1' group is tested. In any other cases, the response is transformed into a factor and the probability of the second level is tested.

Value

`method.test`	a character string giving the name of the test.
`data.name`	a character string giving the name(s) of the data.
`alternative`	a character string describing the alternative hypothesis, always two-sided.
`estimate`	the estimated probabilities.
`null.value`	the theoretical probabilities.
`statistic`	test statistics.
`parameter`	test degrees of freedom.
`p.value`	p-value of the test.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`prop.test`, `chisq.bin.exp`, `prop.bin.multcomp`

Examples

```
response <- c(rep(0:1,c(40,60)),rep(0:1,c(55,45)),rep(0:1,c(65,35)))
fact <- gl(3,100,labels=LETTERS[1:3])
p.theo <- c(0.5,0.45,0.2)
chisq.theo.bintest(response~fact,p=p.theo)
```

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

Description

Performs 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`.

Details

Since a chi-squared test is an approximate test, an exact test is preferable when the number of individuals is small (200 is a reasonable minimum). See `multinomial.theo.multcomp` in that case.

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, multinomial.test, multinomial.theo.multcomp

Examples

```
counts <- c(49,30,63,59)
p.theo <- c(0.2,0.1,0.45,0.25)
chisq.test(counts,p=p.theo)
chisq.theo.multcomp(counts,p=p.theo)
```

---

cochran.qtest *Cochran's Q test*

Description

Performs 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.

Details

If the response is a 0/1 variable, the probability of the '1' group is tested. In any other cases, the response is transformed into a factor and the probability of the second level is tested.

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

Performs 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

Performs 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

Performs 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

Plots 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 an idea of John Fox.

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

Computes the Cramér'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	Cramér'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

Computes 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))
```

CvM.test *Two-sample Cramer-von Mises test*

Description

Wrapper for cramer.test, for more convenient result printing.

Usage

```
CvM.test(x, y, ...)
```

Arguments

x	a numeric vector of data values.
y	a numeric vector of data values.
...	additional arguments to cramer.test. See help of this function.

Details

See help of the `cramer.test` function for more explanations.

Value

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

Author(s)

Maxime Hervé <mx.herve@gmail.com> based on the function of Carsten Franz

Examples

```
set.seed(1109)
x <- rpois(30,2)
y <- rpois(30,3)
CvM.test(x,y)
```

---

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

Description

Draws 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)
```

---

deprecated *Deprecated functions in RVAideMemoire package*

Description

Functions that are not usable anymore, and will be entirely removed from the package in future versions.

Usage

```
byf.normhist(...)
cor.sparse(...)
DA.confusion(...)
DA.valid(...)
DA.var(...)
fc.multcomp(...)
friedman.rating.test(...)
kruskal.rating.test(...)
pairwise.manova(...)
pairwise.wilcox.rating.test(...)
plot1comp.ind(...)
plot1comp.var(...)
PLSDA.ncomp(...)
PLSDA.test(...)
s.corcircle2(...)
scat.mix.categorical(...)
scat.mix.numeric(...)
scatter.coa2(...)
wilcox.paired.rating.multcomp(...)
wilcox.rating.signtest(...)
wilcox.rating.test(...)
```

Arguments

...	previous arguments.

Details

`byf.normhist` was not very useful and `byf.hist` does nearly the same job.

`cor.sparse` is replaced by the more generic `MVA.plot`.

`DA.confusion` and `DA.valid` are replaced by the more generic `MVA.cmv` and `MVA.cv`.

DA.var is replaced by the more generic MVA.synt.

fc.multcomp is not useful anymore since lstrends (package lsmeans) does the same job in a much more powerful manner (see argument var of lstrends).

friedman.rating.test, kruskal.rating.test, wilcox.rating.test, wilcox.rating.signtest, pairwise.wilcox.rating.test and wilcox.paired.rating.multcomp can be problematic with ratings (in which ties and zeroes are very frequent). The use of CLM(M)s (*via* clm and clmm) is recommended.

pairwise.manova is not useful anymore since lsmeans (package lsmeans) does the same job in a much more powerful manner (on "mlm" objects, created by lm and not manova)

plot1comp.ind, plot1comp.var, s.corcircle2, scat.mix.categorical, scat.mix.numeric and scatter.coa2 are replaced by the more generic MVA.plot.

PLSDA.ncomp was not really useful and mvr does nearly the same job.

PLSDA.test is replaced by the more generic MVA.test.

---

fisher.bintest *Fisher's exact test for binary variables*

Description

Performs a Fisher's exact test for comparing response probabilities (i.e. when the response variable is a binary variable). The function is in fact a wrapper to the Fisher's exact test for count data. If the p-value of the test is significant, the function performs pairwise comparisons by using Fisher's exact tests.

Usage

```
fisher.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.

Details

If the response is a 0/1 variable, the probability of the '1' group is tested. In any other cases, the response is transformed into a factor and the probability of the second level is tested.

Since chi-squared and G tests are approximate tests, exact tests are preferable when the number of individuals is small (200 is a reasonable minimum).

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.
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>

See Also

chisq.bintest, G.bintest

Examples

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

---

fisher.multcomp *Pairwise comparisons using Fisher's exact test*

Description

Performs pairwise comparisons after a comparison of proportions or after a test for independence of 2 categorical variables, by using a Fisher's exact test.

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.

Details

Since chi-squared and G tests are approximate tests, exact tests are preferable when the number of individuals is small (200 is a reasonable minimum).

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`, `prop.test`, `fisher.test`

Examples

```
# 2-column contingency table: comparison of proportions
tab.cont1 <- matrix(c(17,23,12,24,20,10),ncol=2,dimnames=list(c("Control",
  "Treatment1","Treatment2"),c("Alive","Dead")),byrow=TRUE)
fisher.test(tab.cont1)
fisher.multcomp(tab.cont1)

# 3-column contingency table: independence test
tab.cont2 <- 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"))))
fisher.test(tab.cont2)
fisher.multcomp(tab.cont2)
```

`fp.test` *Fligner-Policello test*

Description

Performs a Fligner-Policello test of the null that the medians in the two groups (samples) are the same.

Usage

```
fp.test(x, ...)

## Default S3 method:
fp.test(x, y, delta = 0, alternative = "two.sided", ...)

## S3 method for class 'formula'
fp.test(formula, data, subset, ...)
```

Arguments

x	a numeric vector of data values.
y	a numeric vector of data values.
delta	null difference in medians tested.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
formula	a formula of the form a ~ b, where a and b give the data values and corresponding groups.
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.
...	further arguments to be passed to or from other methods.

Details

The Fligner-Policello test does not assume that the shape of the distribution is similar in two groups, contrary to the Mann-Whitney-Wilcoxon test. However, it assumes that the the distributions are symmetric.

Value

statistic	test statistics.
p.value	p-value of the test.
alternative	a character string describing the alternative hypothesis.
method	a character string indicating the name of the test.
data.name	a character string giving the names of the data.
null.value	the specified hypothesized value of the median difference.

Author(s)

Maxime Hervé <mx.herve@gmail.com> based on fp.test

See Also

fp.test, wilcox.test

Examples

```
x <- rpois(20,3)
y <- rpois(20,5)
fp.test(x,y)
```

`G.bintest` *G-test for binary variables*

Description

Performs a G-test for comparing response probabilities (i.e. when the response variable is a binary variable). The function is in fact a wrapper to the G-test for comparison of proportions on a contingency table. If the p-value of the test is significant, the function performs pairwise comparisons by using G-tests.

Usage

```
G.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`.

Details

If the response is a 0/1 variable, the probability of the '1' group is tested. In any other cases, the response is transformed into a factor and the probability of the second level is tested.

Since a G-test is an approximate test, an exact test is preferable when the number of individuals is small (200 is a reasonable minimum). See `fisher.bintest` in that case.

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.

parameter	test degrees of freedom.
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>

See Also

chisq.bintest, fisher.bintest

Examples

```
response <- c(rep(0:1,c(40,60)),rep(0:1,c(55,45)),rep(0:1,c(65,35)))
fact <- gl(3,100,labels=LETTERS[1:3])
G.bintest(response~fact)
```

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

Description

Performs 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.

Details

Since a G-test is an approximate test, an exact test is preferable when the number of individuals is small (200 is a reasonable minimum). See multinomial.multcomp in that case.

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, multinomial.test, multinomial.multcomp

Examples

```
counts <- c(49,30,63,59)
G.test(counts)
G.multcomp(counts)
```

G.test *G-test*

Description

Perfoms 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 with two coloums, equality of proportion of successes between populations is tested.

- if x is a matrix with more than two columns, independence of rows and columns is tested.

Since a G-test is an approximate test, an exact test is preferable when the number of individuals is small (200 is a reasonable minimum). See `multinomial.test` in that case with a vector, `fisher.test` with a matrix.

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`, `multinomial.test`, `fisher.test` `G.multcomp`, `G.theo.multcomp`, `pairwise.G.test`

Examples

```
counts <- c(49,30,63,59)
G.test(counts)
```

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

Description

Performs 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.

Details

Since a G-test is an approximate test, an exact test is preferable when the number of individuals is small (200 is a reasonable minimum). See multinomial.theo.multcomp in that case.

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, multinomial.test, multinomial.theo.multcomp

Examples

```
counts <- c(49,30,63,59)
p.theo <- c(0.2,0.1,0.45,0.25)
G.test(counts,p=p.theo)
G.theo.multcomp(counts,p=p.theo)
```

`ind.contrib` *Individual contributions in regression*

Description

Computes 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)
```

`least.rect` *Least rectangles linear regression*

Description

Fits 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

Cuts 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

Adds 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

Computes 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)
```

---

mood.medtest *Mood's median test*

Description

Performs a Mood's median test to compare medians of independent samples.

Usage

```
mood.medtest(x, ...)

## Default S3 method:
mood.medtest(x, g, exact = NULL, ...)

## S3 method for class 'formula'
mood.medtest(formula, data, subset, ...)
```

Arguments

x	a numeric vector of data values.
g	a vector or factor object giving the group for the corresponding elements of x.
exact	a logical indicating whether an exact p-value should be computed.
formula	a formula of the form a ~ b, where a and b give the data values and corresponding groups.
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.
...	further arguments to be passed to or from other methods.

Details

If exact=NULL, a Fisher's exact test is used if the number of data values is < 200; otherwise a chi-square test is used, with Yates continuity correction if necessary.

Value

method	a character string indicating the name of the test.
data.name	a character string giving the name(s) of the data.
statistic	the value the chi-squared test statistic (in case of a chis-square test).
parameter	the degrees of freedom of the approximate chi-squared distribution of the test statistic (in case of a chis-square test).
p.value	the p-value of the test.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
set.seed(1716)
response <- c(rnorm(10,3,1.5),rnorm(10,5.5,2))
fact <- gl(2,10,labels=LETTERS[1:2])
mood.medtest(response~fact)
```

mqqnorm *Multivariate normality QQ-Plot*

Description

Draws 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))
```

---

multinomial.multcomp *Pairwise comparisons after an exact multinomial test*

Description

Performs pairwise comparisons after a global exact multinomial test. These comparisons are performed using exact binomial tests.

Usage

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

Arguments

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

Details

Since chi-squared and G tests are approximate tests, exact tests are preferable when the number of individuals is small (200 is a reasonable minimum).

An exact multinomial test with two groups is strictly the same than an exact binomial test.

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

multinomial.test, binom.test

Examples

```
counts <- c(5,15,23)
multinomial.test(counts)
multinomial.multcomp(counts)
```

---

multinomial.test *Exact multinomial test*

Description

Perfoms an exact multinomial test on a vector of counts.

Usage

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

Arguments

x	numeric vector (counts).
p	theoretical proportions (optional).

Details

The function works as chisq.test or G.test :

- if theoretical proportions are not given, equality of counts is tested

- if theoretical proportions are given, equality of counts to theoretical counts (given by theoretical proportions) is tested.

Since chi-squared and G tests are approximate tests, exact tests are preferable when the number of individuals is small (200 is a reasonable minimum).

Be aware that the calculation time increases with the number of individuals (i.e. the sum of x) and the number of groups (i.e. the length of x).

An exact multinomial test with two groups is strictly the same as an exact binomial test.

Value

method	name of the test.
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> based on multinomial.test

See Also

chisq.test, G.test, binom.test, multinomial.multcomp, multinomial.theo.multcomp

Examples

```
counts <- c(5,15,23)
multinomial.test(counts)
```

---

multinomial.theo.multcomp

Pairwise comparisons after an exact multinomial test for given probabilities

---

Description

Performs pairwise comparisons after a global exact multinomial test for given probabilities. These comparisons are performed using exact binomial tests.

Usage

```
multinomial.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.

Details

Since chi-squared and G tests are approximate tests, exact tests are preferable when the number of individuals is small (200 is a reasonable minimum).

An exact multinomial test with two groups is strictly the same than an exact binomial test.

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.
p.value2	corrected p-values.
p.value	data frame of results.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

multinomial.test, binom.test

Examples

```
counts <- c(5,15,23)
p.theo <- c(0.2,0.5,0.3)
multinomial.test(counts,p=p.theo)
multinomial.theo.multcomp(counts,p=p.theo)
```

MVA.anova *Type II permutational test for constrained multivariate analyses*

Description

This function is a wrapper to anova.cca(...,by="terms") but performs type II tests (whereas anova.cca performs type I).

Usage

```
MVA.anova(object, ...)
```

Arguments

object	a result object from cca, rda or capscale.
...	additional arguments to anova.cca (can be permutations, model, parallel and/or strata). See help of this function.

Details

See anova.cca for detailed explanation of what is done. The only difference with anova.cca is that MVA.anova performs type II tests instead of type I.

See example of adonis.II for the difference between type I (sequential) and type II tests.

Value

a data frame of class "anova".

Author(s)

Maxime Hervé <mx.herve@gmail.com>

MVA.biplot *Biplot of multivariate analyses*

Description

Displays a biplot of a multivariate analysis. This just consists in superimposing a score plot and a correlation circle (plus centroids of factor levels in constrained analyses, RDA or CCA). The correlation circle is adjusted to fit the size of the score plot.

Usage

```
MVA.biplot(x, xax = 1, yax = 2, scaling = 2, sco.set = c(12, 1, 2),
  cor.set = c(12, 1, 2), space = 1, ratio = 0.9, weights = 1,
  constraints = c("nf", "n", "f", NULL), sco.args = list(),
  cor.args = list(), f.col = 1, f.cex = 1)
```

Arguments

x	a multivariate analysis (see Details).
xax	the horizontal axis.
yax	the vertical axis.
scaling	type of scaling (see MVA.scoreplot).
sco.set	scores to be displayed, when several sets are available (see MVA.scoreplot).
cor.set	correlations to be displayed, when several sets are available (see MVA.scoreplot).
space	space to use, when several are available (see MVA.scoreplot and MVA.corplot).
ratio	constant for adjustement of correlations to the size of the score plot (0.9 means the longest arrows is 90% of the corresponding axis).
weights	only used with constrained analyses (RDA or CCA) where some constraints are factors. Individual weights, used to calculate barycenter positions.
constraints	only used with constrained analyses (RDA or CCA). Type of constraints to display: quantitative ("n"), factors ("f"), both ("nf", default) or none ("NULL").

sco.args	list containing optional arguments to pass to MVA.scoreplot. All arguments are accepted.
cor.args	list containing optional arguments to pass to MVA.corplot. All arguments are accepted except xlab, ylab, circle, intcircle, drawintaxes, add and add.const.
f.col	color(s) used for barycenters in case of a constraint analysis (RDA or CCA) containing factor constraint(s). Can be a unique value, a vector giving one color per constraint or a vector giving one color per barycenter (all factors confounded).
f.cex	size(s) used for barycenters in case of a constraint analysis (RDA or CCA) containing factor constraint(s). Can be a unique value, a vector giving one size per constraint or a vector giving one size per barycenter (all factors confounded).

Details

This function should not be use directly. Prefer the general MVA.plot, to which all arguments can be passed.

All multivariate analyses covered by MVA.corplot can be used for biplots.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
require(vegan)
data(iris)
RDA <- rda(iris[,1:4]~Species,data=iris)
MVA.plot(RDA,"biplot",cor.args=list(col="purple"),ratio=0.8,f.col=c("red","green","blue"))
```

MVA.cmv *Cross model validation*

Description

Performs cross model validation (2CV) with different PLS analyses.

Usage

```
MVA.cmv(X, Y, repet = 10, kout = 7, kinn = 8, ncomp = 8, model = c("PLSR",
  "CPPLS", "PLS-DA", "PPLS-DA", "PLS-DA/LDA", "PLS-DA/QDA", "PPLS-DA/LDA",
  "PPLS-DA/QDA"), crit.inn = c("RMSEP", "Q2", "NMC"), Q2diff = 0.05,
  lower = 0.5, upper = 0.5, Y.add = NULL, weights = rep(1, nrow(X)),
  set.prior = FALSE, crit.DA = c("plug-in", "predictive", "debiased"), ...)
```

Arguments

X	a data frame of independent variables.
Y	the dependent variable(s): numeric vector, data frame of quantitative variables or factor.
repet	an integer giving the number of times the whole 2CV procedure has to be repeated.
kout	an integer giving the number of folds in the outer loop.
kinn	an integer giving the number of folds in the inner loop.
ncomp	an integer giving the maximal number of components to be tested in the inner loop (can be re-set depending on the size of the train sets).
model	the model to be fitted (see Details).
crit.inn	the criterion to be used to choose the number of components in the inner loop. Root Mean Square Error of Prediction ("RMSEP", default) and Q2 ("Q2") are only used for PLSR and CPPLS, whereas the Number of MisClassifications ("NMC") is only used for discriminant analyses.
Q2diff	the threshold to be used if the number of components is chosen according to Q2. The next component is added only if it makes the Q2 increase more than Q2diff (5% by default).
lower	a vector of lower limits for power optimisation in CPPLS or PPLS-DA (see cppls.fit).
upper	a vector of upper limits for power optimisation in CPPLS or PPLS-DA (see cppls.fit).
Y.add	a vector or matrix of additional responses containing relevant information about the observations, in CPPLS or PPLS-DA (see cppls.fit).
weights	a vector of individual weights for the observations, in CPPLS or PPLS-DA (see cppls.fit).
set.prior	only used when a second analysis (LDA or QDA) is performed. If TRUE, the prior probabilities of class membership are defined according to the mean weight of individuals belonging to each class. If FALSE, prior probabilities are obtained from the data sets on which LDA/QDA models are built.
crit.DA	criterion used to predict class membership when a second analysis (LDA or QDA) is used. See predict.lda.
...	other arguments to pass to plsr (PLSR, PLS-DA) or cppls (CPPLS, PPLS-DA).

Details

Cross model validation is detailed is Szymanska et al (2012). Some more details about how this function works:

- when a discriminant analysis is used ("PLS-DA", "PPLS-DA", "PLS-DA/LDA", "PLS-DA/QDA", "PPLS-DA/LDA" or "PPLS-DA/QDA"), the training sets (test set itself in the inner loop, test+validation sets in the outer loop) are generated in respect to the relative proportions of the levels of Y in the original data set (see splitf).

- "PLS-DA" is considered as PLS2 on a dummy-coded response. For a PLS-DA based on the CPPLS algorithm, use "PPLS-DA" with lower and upper limits of the power parameters set to 0.5.

- if a second analysis is used ("PLS-DA/LDA", "PLS-DA/QDA", "PPLS-DA/LDA" or "PPLS-DA/QDA"), a LDA or QDA is built on scores of the first analysis (PLS-DA or PPLS-DA) also in the inner loop. The number of misclassifications, based on this second analysis, is used to choose the number of components.

Value

model	model used.
type	type of model used.
repet	number of times the whole 2CV procedure was repeated.
kout	number of folds in the outer loop.
kinn	number of folds in the inner loop.
crit.inn	criterion used to choose the number of components in the inner loop.
crit.DA	criterion used to classify individuals of the test and validation sets.
Q2diff	threshold used if the number of components is chosen according to Q2.
models.list	list of of models generated (repet*kout models), for PLSR, CPPLS, PLS-DA and PPLS-DA.
models1.list	list of of (P)PLS-DA models generated (repet*kout models), for PLS-DA/LDA, PLS-DA/QDA, PPLS-DA/LDA and PPLS-DA/QDA.
models2.list	list of of LDA/QDA models generated (repet*kout models), for PLS-DA/LDA, PLS-DA/QDA, PPLS-DA/LDA and PPLS-DA/QDA.
RMSEP	RMSEP computed from the models used in the outer loops (repet values).
Q2	Q2 computed from the models used in the outer loops (repet values).
NMC	NMC computed from the models used in the outer loops (repet values).

Author(s)

Maxime Hervé <mx.herve@gmail.com>

References

Szymanska E, Saccenti E, Smilde AK and Westerhuis J (2012) Double-check: validation of diagnostic statistics for PLS-DA models in metabolomics studies. Metabolomics (2012) 8:S3-S16.

See Also

predict.MVA.cmv, mvr, lda, qda

Examples

```
require(pls)
require(MASS)

# PLSR
data(yarn)
## Not run: MVA.cmv(yarn$NIR,yarn$density,model="PLSR")

# PPLS-DA coupled to LDA
data(mayonnaise)
## Not run: MVA.cmv(mayonnaise$NIR,factor(mayonnaise$oil.type),model="PPLS-DA/LDA",crit.inn="NMC")
```

MVA.corplot *Correlation circle of multivariate analyses*

Description

Displays a correlation circle of a multivariate analysis.

Usage

```
MVA.corplot(x, xax = 1, yax = 2, thresh = 0, fac = NULL, set = c(12, 1, 2), space = 1,
  xlab = NULL, ylab = NULL, main = NULL, circle = TRUE, intcircle = 0.5, points = TRUE,
  ident = TRUE, arrows = TRUE, labels = NULL, main.pos = c("bottomleft", "topleft",
  "bottomright", "topright"), main.cex = 1.3, legend = FALSE, legend.pos = c("topleft",
  "topright", "bottomleft", "bottomright"), legend.title = NULL, legend.lab = NULL,
  pch = 16, cex = 1, col = 1, lwd = 1, drawintaxes = TRUE, add = FALSE, add.const = 1,
   keepmar = FALSE)
```

Arguments

x	a multivariate analysis (see Details).
xax	the horizontal axis.
yax	the vertical axis. This can be set to NULL for a one-dimensional graph, which is a dotchart.
thresh	threshold (in absolute value of the correlation coefficient) of variables to be plotted.
fac	an optional factor defining groups of variables.
set	variables to be displayed, when several sets are available (see Details). 12 (default) for both sets, 1 for X or constraints, 2 for Y or constrained variables.
space	variables to be displayed, when several spaces are available (see Details). space is the number of the space to be plotted.
xlab	legend of the horizontal axis. If NULL (default), automatic labels are used depending on the multivariate analysis.

ylab	only used for two-dimensional graphs. Legend of the vertical axis. If NULL (default), automatic labels are used depending on the multivariate analysis.
main	optional title of the graph.
circle	only used for two-dimensional graphs. Logical indicating if the circle of radius 1 should be plotted.
intcircle	only used for two-dimensional graphs. Vector of one or several values indicating radii of circles to be plotted inside the main circle. Can be set to NULL.
points	only used for two-dimensional graphs. If FALSE, arrows or points (see arrows) are replaced with their corresponding label (defined by labels).
ident	only used for two-dimensional graphs when points=TRUE. A logical indicating if variable names should be displayed.
arrows	only used if points=TRUE. Logical indicating if arrows should be plotted. If FALSE, points are displayed at the extremity of the arrows.
labels	names of the variables. If NULL (default), labels correspond to variable names found in the data used in the multivariate analysis. For two-dimensional graphs, only used if ident=TRUE.
main.pos	position of the title, if main is not NULL. Default to "bottomleft".
main.cex	size of the title, if main is not NULL.
legend	only used for two-dimensional graphs. Logical indicating if a legend should be added to the graph.
legend.pos	position of the legend, if legend is TRUE. Default to "topleft".
legend.title	optional title of the legend, if legend is TRUE.
legend.lab	legend labels, if legend is TRUE. If NULL, levels of the factor defined by fac are used.
pch	symbol(s) used for points, when points are displayed (see arrows). If fac is not NULL, can be a vector of length one or a vector giving one value per group. Otherwise a vector of any length can be defined, which is recycled if necessary.
cex	size of the points and/or of the variable names. For two-dimensional graphs: if fac is not NULL, can be a vector of length one or a vector giving one value per group; otherwise a vector of any length can be defined, which is recycled if necessary. For dotcharts, gives the size used for points and all labels (see dotchart).
col	color(s) used for points and/or variable names. If fac is not NULL, can be a vector of length one or a vector giving one value per group. Otherwise a vector of any length can be defined, which is recycled if necessary (not available for density histograms, see dhist).
lwd	only used if arrows are displayed. Width of arrows. If fac is not NULL, can be a vector of length one or a vector giving one value per group. Otherwise a vector of any length can be defined, which is recycled if necessary.
drawintaxes	logical indicating if internal axes should be drawn.
add	only used for two-dimensional graphs. Logical indicating if the correlation circle should be added to an existing graph.

add.const	only used for two-dimensional graphs and if add is TRUE. Constant by which correlations are multiplied to fit onto the original graph.
keepmar	only used for two-dimensional graphs. Logical indicating if margins defined by MVA.corplot should be kept after plotting (necessary in some cases when add=TRUE).

Details

This function should not be use directly. Prefer the general MVA.plot, to which all arguments can be passed.

Many multivariate analyses are supported, from various packages:

- PCA: dudi.pca, rda.

- sPCA: spca.

- IPCA: ipca.

- sIPCA: sipca.

- LDA: lda, discrimin.

- PLS-DA (PLS2 on a dummy-coded factor): plsda. X space only.

- sPLS-DA (sPLS2 on a dummy-coded factor): splsda. X space only.

- Multilevel (s)PLS-DA ((s)PLS-DA on a dummy-coded factor): multilevel. X space only. Package 'mixOmics' version >= 5.0.4.

- CPPLS: mvr. Set 1 is X, set 2 is Y. If set=12 (default), fac is not available and pch,cex, col, lwd can be defined differently for each set. X space only.

- PLSR: mvr, pls, plsR. Set 1 is X, set 2 is Y. If set=12 (default), fac is not available and pch,cex, col, lwd can be defined differently for each set. X space only.

- sPLSR: pls. Set 1 is X, set 2 is Y. If set=12 (default), fac is not available and pch,cex, col, lwd can be defined differently for each set. X space only.

- Multilevel (s)PLSR: multilevel. Set 1 is X, set 2 is Y. If set=12 (default), fac is not available and pch,cex, col, lwd can be defined differently for each set. X space only. Package 'mixOmics' version >= 5.0.4

- PLS-GLR: plsRglm. Set 1 is X, set 2 is Y. If set=12 (default), fac is not available and pch,cex, col, lwd can be defined differently for each set. Correlations are computed with Y on the link scale.

- PCR: mvr. Set 1 is X, set 2 is Y. If set=12 (default), fac is not available and pch,cex, col, lwd can be defined differently for each set.

- CDA: discrimin, discrimin.coa.

- NSCOA: dudi.nsc. For NSCOA there is no real correlation, but the classical representation of columns is arrows. This is why MVA.corplot was made able to deal with this analysis.

- CCA: cca, cca. Constraints (only quantitative constraints are extracted) in constrained space only.

- Mix analysis: dudi.mix, dudi.hillsmith. Only quantitative variables are displayed.

- RDA (or PCAIV): pcaiv, pcaivortho, rda. With rda, space 1 is constrained space, space 2 is unconstrained space. Only constrained space is available with pcaiv, the opposite for pcaivortho. Set 1 is constraints (only quantitative constraints are extracted), set 2 is dependent variables (only

set 2 is available for pcaivortho). If set=12 (default), fac is not available and pch,cex, col, lwd can be defined differently for each set.

- CCorA: CCorA, rcc. Space 1 is X, space 2 is Y. With rcc a third space is available, in which coordinates are means of X and Y coordinates. In this third space, set 1 is X, set 2 is Y. If set=12 (default), fac is not available and pch,cex, col, lwd can be defined differently for each set.

- rCCorA: rcc. Space 1 is X, space 2 is Y, space 3 is a "common" space in which coordinates are means of X and Y coordinates. In space 3, set 1 is X and set 2 is Y. If set=12 (default), fac is not available and pch,cex, col, lwd can be defined differently for each set.

- CIA: coinertia. Space 1 is X, space 2 is Y, space 3 is a "common" space where X and Y scores are normed. In space 3, set 1 is X and set 2 is Y. If set=12 in space 3 (default), fac is not available and pch,cex, col, lws can be defined differently for each set.

- 2B-PLS: pls. Space 1 is X, space 2 is Y, space 3 is a "common" space in which coordinates are means of X and Y coordinates. In space 3, set 1 is X and set 2 is Y. If set=12 (default), fac is not available and pch,cex, col, lwd can be defined differently for each set.

- 2B-sPLS: pls. Space 1 is X, space 2 is Y, space 3 is a "common" space in which coordinates are means of X and Y coordinates. In space 3, set 1 is X and set 2 is Y. If set=12 (default), fac is not available and pch,cex, col, lwd can be defined differently for each set.

- Multilevel 2B-(s)PLS: pls. Space 1 is X, space 2 is Y, space 3 is a "common" space in which coordinates are means of X and Y coordinates. In space 3, set 1 is X and set 2 is Y. If set=12 (default), fac is not available and pch,cex, col, lwd can be defined differently for each set. Package 'mixOmics' version >= 5.0.4.

- rGCCA: wrapper.rgcca. Space can be 1 to n, the number of blocks (i.e. datasets).

- sGCCA: wrapper.sgcca. Space can be 1 to n, the number of blocks (i.e. datasets).

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
require(ade4)
data(olympic)
PCA <- dudi.pca(olympic$tab,scannf=FALSE)
MVA.plot(PCA,"corr")
```

MVA.cv *Cross validation*

Description

Performs cross validation with different PLS and/or discriminant analyses.

Usage

```
MVA.cv(X, Y, repet = 10, k = 7, ncomp = 5, model = c("PLSR","CPPLS",
  "PLS-DA", "PPLS-DA", "LDA", "QDA", "PLS-DA/LDA", "PLS-DA/QDA", "PPLS-DA/LDA",
  "PPLS-DA/QDA"), lower = 0.5, upper = 0.5, Y.add = NULL, weights = rep(1,
  nrow(X)), set.prior = FALSE, crit.DA = c("plug-in", "predictive",
  "debiased"), ...)
```

Arguments

X	a data frame of independent variables.
Y	the dependent variable(s): numeric vector, data frame of quantitative variables or factor.
repet	an integer giving the number of times the whole 2CV procedure has to be repeated.
k	an integer giving the number of folds.
ncomp	an integer giving the number of components to be used for all models except LDA and QDA (can be re-set depending on the size of the train sets).
model	the model to be fitted (see Details).
lower	a vector of lower limits for power optimisation in CPPLS or PPLS-DA (see cppls.fit).
upper	a vector of upper limits for power optimisation in CPPLS or PPLS-DA (see cppls.fit).
Y.add	a vector or matrix of additional responses containing relevant information about the observations, in CPPLS or PPLS-DA (see cppls.fit).
weights	a vector of individual weights for the observations, in CPPLS or PPLS-DA (see cppls.fit).
set.prior	only used when a LDA or QDA is performed (coupled or not with a PLS model). If TRUE, the prior probabilities of class membership are defined according to the mean weight of individuals belonging to each class. If FALSE, prior probabilities are obtained from the data sets on which LDA/QDA models are built.
crit.DA	criterion used to predict class membership when a LDA or QDA is used. See predict.lda.
...	other arguments to pass to plsr (PLSR, PLS-DA) or cppls (CPPLS, PPLS-DA).

Details

When a discriminant analysis is used ("PLS-DA", "PPLS-DA", "LDA", "QDA", "PLS-DA/LDA", "PLS-DA/QDA", "PPLS-DA/LDA" or "PPLS-DA/QDA"), the training sets are generated in respect to the relative proportions of the levels of Y in the original data set (see splitf).

"PLS-DA" is considered as PLS2 on a dummy-coded response. For a PLS-DA based on the CPPLS algorithm, use "PPLS-DA" with lower and upper limits of the power parameters set to 0.5.

Value

model	model used.
type	type of model used.
repet	number of times the whole 2CV procedure was repeated.
k	number of folds.
k	number of components used.
crit.DA	criterion used to classify individuals of the test sets.
models.list	list of of models generated (repet*k models), for PLSR, CPPLS, PLS-DA, PPLS-DA, LDA and QDA.
models1.list	list of of (P)PLS-DA models generated (repet*k models), for PLS-DA/LDA, PLS-DA/QDA, PPLS-DA/LDA and PPLS-DA/QDA.
models2.list	list of of LDA/QDA models generated (repet*k models), for PLS-DA/LDA, PLS-DA/QDA, PPLS-DA/LDA and PPLS-DA/QDA.
RMSEP	RMSEP vales (repet values).
Q2	Q2 values (repet values).
NMC	NMC values (repet values).

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

predict.MVA.cmv, mvr, lda, qda

Examples

```
require(pls)
require(MASS)

# PLSR
data(yarn)
## Not run: MVA.cv(yarn$NIR,yarn$density,model="PLSR")

# PPLS-DA coupled to LDA
data(mayonnaise)
## Not run: MVA.cv(mayonnaise$NIR,factor(mayonnaise$oil.type),model="PPLS-DA/LDA",crit.inn="NMC")
```

MVA.loadplot *Loading plot of multivariate analyses*

Description

Displays a loading plot of a multivariate analysis.

Usage

```
MVA.loadplot(x, xax = 1, yax = 2, fac = NULL, set = c(12, 1, 2), space = 1, map = TRUE,
  xlab = NULL, ylab = NULL, main = NULL, points = TRUE, ident = TRUE, links = TRUE,
  line = TRUE, labels = NULL, main.pos = c("bottomleft", "topleft","bottomright",
  "topright"), main.cex = 1.3, legend = FALSE, legend.pos = c("topleft", "topright",
  "bottomleft", "bottomright"), legend.title = NULL, legend.lab = NULL, pch = 16,
  cex = 1, col = 1, lwd = 1, lty = 1, drawextaxes = TRUE, drawintaxes = TRUE, xlim = NULL,
   ylim = NULL)
```

Arguments

x	a multivariate analysis (see Details).
xax	the horizontal axis.
yax	the vertical axis. This can be set to NULL for a one-dimensional graph.
fac	only used for one-dimensional graphs. An optional factor defining groups of variables.
set	variables to be displayed, when several sets are available (see Details). 12 (default) for both sets, 1 for X, 2 for Y.
space	variables to be displayed, when several spaces are available (see Details). space is the number of the space to be plotted.
map	logical indicating if a two-dimensional (TRUE, default) or a one-dimensional graph should be drawn. A one-dimensional graph can show loadings for one or two dimensions, both horizontally.
xlab	only used for two-dimensional graphs. Legend of the horizontal axis. If NULL (default), automatic labels are used depending on the multivariate analysis.
ylab	legend of the vertical axis. If NULL (default), automatic labels are used depending on the multivariate analysis.
main	optional title of the graph.
points	only used for two-dimensional graphs. If FALSE, lines or points (see links) are replaced with their corresponding label (defined by labels).
ident	logical indicating if variable names should be displayed. Only used when points=TRUE for two-dimensional graphs.
links	only used for two-dimensional graphs when points=TRUE. Logical indicating if variables should be linked to the origin of the graph. If FALSE, points are displayed at the extremity of the segments.

line	only used for one-dimensional graphs when yax=NULL. Logical indicating if loadings should be linked (default) as displayed as sticks.
labels	only used if ident=TRUE. Names of the variables. If NULL (default), labels correspond to variable names found in the data used in the multivariate analysis.
main.pos	only used for one-dimensional graphs. Position of the title, if main is not NULL. Default to "bottomleft".
main.cex	size of the title, if main is not NULL.
legend	logical indicating if a legend should be added to the graph.
legend.pos	position of the legend, if legend is TRUE. Default to "topleft".
legend.title	optional title of the legend, if legend is TRUE.
legend.lab	legend labels, if legend is TRUE. If NULL for a one-dimensional graph, dimension names are used. If NULL for a two-dimensional graph, levels of the factor defined by fac are used.
pch	only used for two-dimensional graphs. Symbol(s) used for points, when points are displayed (see links). If fac is not NULL, can be a vector of length one or a vector giving one value per group. Otherwise a vector of any length can be defined, which is recycled if necessary.
cex	size of the points and/or of the variable names. For two-dimensional graphs: if fac is not NULL, can be a vector of length one or a vector giving one value per group; otherwise a vector of any length can be defined, which is recycled if necessary.
col	color(s) used for points, variable names and/or lines/sticks. For one-dimensional graphs, can be a vector of length one or a vector giving one value per line. For two-dimensional graphs: if fac is not NULL, can be a vector of length one or a vector giving one value per group. Otherwise a vector of any length can be defined, which is recycled if necessary (not available for density histograms, see dhist).
lwd	width of lines. For one-dimensional graphs, can be a vector of length one or a vector giving one value per line. For two-dimensional graphs: if fac is not NULL, can be a vector of length one or a vector giving one value per group. Otherwise a vector of any length can be defined, which is recycled if necessary.
lty	only used for one-dimensional graphs. Can be a vector of length one or a vector giving one value per line.
drawextaxes	logical indicating if external axes should be drawn.
drawintaxes	only used for two-dimensional graphs. Logical indicating if internal axes should be drawn.
xlim	only used in two-dimensional graphs. Limits of the horizontal axis. If NULL, limits are computed automatically.
ylim	limits of the vertical axis. If NULL, limits are computed automatically.

Details

This function should not be use directly. Prefer the general MVA.plot, to which all arguments can be passed.

Many multivariate analyses are supported, from various packages:

- PCA: prcomp, princomp, dudi.pca, rda, pca, pca.

- sPCA: spca.

- IPCA: ipca.

- sIPCA: sipca.

- LDA: lda, discrimin.

- PLS-DA (PLS2 on a dummy-coded factor): plsda. X space only.

- sPLS-DA (sPLS2 on a dummy-coded factor): splsda. X space only.

- Multilevel (s)PLS-DA ((s)PLS-DA on a dummy-coded factor): multilevel. X space only. Package 'mixOmics' version >= 5.0.4.

- CPPLS: mvr. X space only.

- PLSR: mvr, pls, plsR. X space only.

- sPLSR: pls. X space only.

- Multilevel (s)PLSR: multilevel. X space only. Package 'mixOmics' version >= 5.0.4

- PLS-GLR: plsRglm.

- PCR: mvr.

- CDA: discrimin, discrimin.coa.

- NSCOA: dudi.nsc.

- MCA: dudi.acm.

- Mix analysis: dudi.mix, dudi.hillsmith.

- PCIA: procuste. Set 1 is X, set 2 is Y.

- RDA (or PCAIV): pcaiv, pcaivortho, rda. With rda, space 1 is constrained space, space 2 is unconstrained space. Only constrained space is available with pcaiv, the opposite for pcaivortho.

- CCorA: rcc. Space 1 is X, space 2 is Y.

- rCCorA: rcc. Space 1 is X, space 2 is Y.

- CIA: coinertia. Space 1 is X, space 2 is Y.

- 2B-PLS: pls. Space 1 is X, space 2 is Y.

- 2B-sPLS: pls. Space 1 is X, space 2 is Y.

- Multilevel 2B-(s)PLS: pls. Space 1 is X, space 2 is Y. Package 'mixOmics' version >= 5.0.4.

- rGCCA: wrapper.rgcca. Space can be 1 to n, the number of blocks (i.e. datasets).

- sGCCA: wrapper.sgcca. Space can be 1 to n, the number of blocks (i.e. datasets).

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
require(ade4)
data(olympic)
PCA <- dudi.pca(olympic$tab,scannf=FALSE)
MVA.plot(PCA,"load")
```

`MVA.pairplot` *Paired plot of multivariate analyses*

Description

Displays a paired plot (*i.e.* a score plot of paired points) of a multivariate analysis.

Usage

```
MVA.pairplot(x, xax = 1, yax = 2, pairs = NULL, scaling = 2, space = 1, fac = NULL,
  xlab = NULL, ylab = NULL, main = NULL, ident = TRUE, labels = NULL, cex = 0.7, col = 1,
  lwd = 1, main.pos = c("bottomleft", "topleft", "bottomright", "topright"),
  main.cex = 1.3, legend = FALSE, legend.pos = c("topleft", "topright", "bottomleft",
  "bottomright"), legend.title = NULL, legend.lab = NULL, drawextaxes = TRUE,
  drawintaxes = TRUE, xlim = NULL, ylim = NULL)
```

Arguments

`x`	a multivariate analysis (see Details).
`xax`	the horizontal axis.
`yax`	the vertical axis. Cannot be `NULL`, only two-dimensional graphs can be drawn.
`pairs`	two-level factor identifying paired individuals (in the same order in both sets of points). Can be omitted with multivariate analyses where two sets of points are available in the same space (see `MVA.scoreplot`). In this case these sets are automatically detected.
`scaling`	type of scaling. Only available with some analyses performed with the `vegan` package. See Details of `MVA.scoreplot`.
`space`	scores to be displayed, when several spaces are available (see Details of `MVA.scoreplot`). `space` is the number of the space to be plotted.
`fac`	an optional factor defining groups pairs.
`xlab`	legend of the horizontal axis. If `NULL` (default), automatic labels are used depending on the multivariate analysis.
`ylab`	legend of the vertical axis. If `NULL` (default), automatic labels are used depending on the multivariate analysis.
`main`	optional title of the graph.
`ident`	logical indicating if variable names should be displayed.
`labels`	names of the individuals. If `NULL` (default), labels correspond to row names of the data used in the multivariate analysis.
`cex`	size of the labels. If `fac` is not `NULL`, can be a vector of length one or a vector giving one value per group. Otherwise a vector of any length can be defined, which is recycled if necessary.
`col`	color(s) used for arrows and labels. If `fac` is not `NULL`, can be a vector of length one or a vector giving one value per group. Otherwise a vector of any length can be defined, which is recycled if necessary.

lwd	width of arrows. If fac is not NULL, can be a vector of length one or a vector giving one value per group. Otherwise a vector of any length can be defined, which is recycled if necessary.
main.pos	position of the title, if main is not NULL. Default to "bottomleft".
main.cex	size of the title, if main is not NULL.
legend	logical indicating if a legend should be added to the graph.
legend.pos	position of the legend, if legend is TRUE. Default to "topleft".
legend.title	optional title of the legend, if legend is TRUE.
legend.lab	legend labels, if legend is TRUE. If NULL and fac is defined, levels of fac are used.
drawextaxes	logical indicating if external axes should be drawn..
drawintaxes	logical indicating if internal axes should be drawn.
xlim	limits of the horizontal axis. If NULL, limits are computed automatically.
ylim	limits of the vertical axis. If NULL, limits are computed automatically.

Details

This function should not be use directly. Prefer the general MVA.plot, to which all arguments can be passed.

All multivariate analyses supported by MVA.scoreplot can be used for a paired plot.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
require(ade4)
data(macaca)
PCIA <- procuste(macaca$xy1,macaca$xy2)
MVA.plot(PCIA,"pairs")
```

MVA.plot *Plotting of multivariate analyses*

Description

Displays several kinds of plots for multivariate analyses.

Usage

```
MVA.plot(x, type = c("scores", "loadings", "correlations", "biplot", "pairs",
  "trajectories"), ...)
```

Arguments

x	a multivariate analysis (see Details).
type	the type of plot to be displayed: score plot (default), loading plot, correlation circle, biplot, score plot showing paired samples or score plot showing trajectories, respectively.
...	arguments to be passed to subfunctions. See Details.

Details

Different subfunctions are used depending on the type of plot to be displayed: `MVA.scoreplot`, `MVA.loadplot`, `MVA.corplot`, `MVA.biplot`, `MVA.pairplot` or `MVA.trajplot`. These functions should not be used directly (everything can be done with the general `MVA.plot`) but for convenience, arguments and analyses supported are detailed in separate help pages.

Warning: the use of `attach` before running a multivariate analysis can prevent `MVA.plot` to get the values it needs, and make it fail.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

MVA.scoreplot *Score plot of multivariate analyses*

Description

Displays a score plot of a multivariate analysis.

Usage

```
MVA.scoreplot(x, xax = 1, yax = 2, scaling = 2, set = c(12, 1, 2), space = 1,
  byfac = TRUE, fac = NULL, barycenters = TRUE, stars = TRUE, contours = FALSE,
  dhist = TRUE, weights = 1, xlab = NULL, ylab = NULL, main = NULL, pch = 16,
  cex = 1, col = 1, points = TRUE, labels = NULL, main.pos = c("bottomleft",
  "topleft", "bottomright", "topright"), main.cex = 1.3, fac.lab = NULL,
  fac.cex = 1, legend = FALSE, legend.pos = c("topleft", "topright", "bottomleft",
  "bottomright"), legend.title = NULL, legend.lab = NULL, legend.cex = 1,
  drawextaxes = TRUE, drawintaxes = TRUE, xlim = NULL, ylim = NULL,
  keepmar = FALSE)
```

Arguments

x	a multivariate analysis (see Details).
xax	the horizontal axis.
yax	the vertical axis. This can be set to `NULL` for a one-dimensional graph. The type of graph to be drawn in this case depends on the value of `dhist`.

scaling	type of scaling. Only available with some analyses performed with the vegan package. See Details.
set	scores to be displayed, when several sets are available (see Details). 12 (default) for both sets, 1 for rows or X, 2 for columns or Y.
space	scores to be displayed, when several spaces are available (see Details). space is the number of the space to be plotted.
byfac	only used with MCA and mix analyses (see Details). If TRUE, a separate score plot is displayed for each factor included in the analysis. In this case fac cannot be used and if main=NULL, the factor names are displayed as titles on the graphs.
fac	an optional factor defining groups of individuals.
barycenters	only used if fac is not NULL. If TRUE (default), the name of each group (defined by fac.lab) is diplayed at the position of the barycenter of this group. Available for two-dimensional graphs and for dotcharts in the one-dimensional case (see dhist).
stars	only used if fac is not NULL. If TRUE (default), the individual of each group are linked to their corresponding barycenter.
contours	only used if fac is not NULL. If TRUE, a polygon of contour is displayed for each group.
dhist	only used in the one-dimensional case. If TRUE (default), a density histogram is displayed. If FALSE, a dotchart is displayed.
weights	individual weights, used to calculate barycenter positions (see barycenters).
xlab	legend of the horizontal axis. If NULL (default), automatic labels are used depending on the multivariate analysis.
ylab	legend of the vertical axis. If NULL (default), automatic labels are used depending on the multivariate analysis. Available for two-dimensional graphs and for density histograms in the one-dimensional case (see dhist).
main	optional title of the graph. Can be a vector of several values for MCA and mix analyses when byfac=TRUE (see byfac).
pch	symbol(s) used for points, when points are displayed (see points). If fac is not NULL, can be a vector of length one or a vector giving one value per group. Otherwise a vector of any length can be defined, which is recycled if necessary. Available for two-dimensional graphs and for dotcharts in the one-dimensional case (see dhist). Re-used for all graphs for MCA and mix analyses when byfac=TRUE (see byfac).
cex	size of the points or of the labels (see points). Available for two-dimensional graphs and for dotcharts in the one-dimensional case (see dhist). For two-dimensional graphs: if fac is not NULL, can be a vector of length one or a vector giving one value per group; otherwise a vector of any length can be defined, which is recycled if necessary. For dotcharts, gives the size used for points and all labels (see dotchart). Re-used for all graphs for MCA and mix analyses when byfac=TRUE (see byfac).
col	color(s) used for points or labels (see points). If fac is not NULL, can be a vector of length one or a vector giving one value per group. Otherwise a vector of any length can be defined, which is recycled if necessary (not available for density

histograms, see dhist). Re-used for all graphs for MCA and mix analyses when byfac=TRUE (see byfac).

points — only used for two-dimensional graphs. If FALSE, points are replaced with their corresponding label (defined by labels). Re-used for all graphs for MCA and mix analyses when byfac=TRUE (see byfac).

labels — used in two-dimensional graphs when points=FALSE and in dotcharts (see dhist). Names of the individuals. If NULL (default), labels correspond to row names of the data used in the multivariate analysis. Re-used for all graphs for MCA and mix analyses when byfac=TRUE (see byfac).

main.pos — position of the title, if main is not NULL. Default to "bottomleft". Re-used for all graphs for MCA and mix analyses when byfac=TRUE (see byfac).

main.cex — size of the title, if main is not NULL. Re-used for all graphs for MCA and mix analyses when byfac=TRUE (see byfac).

fac.lab — only used if fac is not NULL. Labels used to display barycenters in two-dimensional graphs or on the vertical axis of a dotchart in the one-dimensional case (see dhist). If NULL, levels of the factor defined by fac are used. In case of a MCA or a mix analysis with byfac=TRUE (see byfac), labels cannot be changed and correspond to the levels of the factor displayed on each graph.

fac.cex — only used if fac is not NULL and in two-dimensional graphs. Labels used to display barycenters. Can be a vector of length one or a vector giving one value per group. Re-used for all graphs for MCA and mix analyses when byfac=TRUE (see byfac).

legend — logical indicating if a legend should be added to the graph. Available for two-dimensional graphs and for density histograms in the one-dimensional case (see dhist).

legend.pos — position of the legend, if legend is TRUE. Default to "topleft".

legend.title — optional title of the legend, if legend is TRUE. Not available for MCA and mix analyses when byfac=TRUE (see byfac).

legend.lab — legend labels, if legend is TRUE. If NULL, labels defined by fac.labels are used (see fac.labels).

legend.cex — size of legend labels, if legend is TRUE.

drawextaxes — logical indicating if external axes should be drawn. Available for two-dimensional graphs and for density histograms in the one-dimensional case (see dhist).

drawintaxes — logical indicating if internal axes should be drawn.

xlim — limits of the horizontal axis. If NULL, limits are computed automatically. Re-used for all graphs for MCA and mix analyses when byfac=TRUE (see byfac).

ylim — only used in two-dimensional graphs. Limits of the vertical axis. If NULL, limits are computed automatically. Re-used for all graphs for MCA and mix analyses when byfac=TRUE (see byfac).

keepmar — only used in two-dimensional graphs. Logical indicating if margins defined by MVA.scoreplot should be kept after plotting (necessary for biplots).

Details

This function should not be use directly. Prefer the general MVA.plot, to which all arguments can be passed.

Many multivariate analyses are supported, from various packages:

- PCA: prcomp, princomp (if scores=TRUE), dudi.pca, rda, pca, pca. scaling can be defined for rda (see scores.rda).

- sPCA: spca.

- IPCA: ipca.

- sIPCA: sipca.

- PCoA: cmdscale (with at least on non-default argument), dudi.pco, wcmdscale (with at least one non-default argument), capscale, pco, pcoa.

- nMDS: monoMDS, metaMDS, nmds, isoMDS.

- LDA: lda, discrimin.

- PLS-DA (PLS2 on a dummy-coded factor): plsda. X space only.

- sPLS-DA (sPLS2 on a dummy-coded factor): splsda. X space only.

- Multilevel (s)PLS-DA ((s)PLS-DA on a dummy-coded factor): multilevel. X space only. Package 'mixOmics' version >= 5.0.4.

- CPPLS: mvr. X space only.

- PLSR: mvr, pls, plsR. X space only.

- sPLSR: pls. X space only.

- Multilevel (s)PLSR: multilevel. X space only. Package 'mixOmics' version >= 5.0.4

- PLS-GLR: plsRglm.

- PCR: mvr.

- CDA: discrimin, discrimin.coa.

- NSCOA: dudi.nsc.

- MCA: dudi.acm.

- Mix analysis: dudi.mix, dudi.hillsmith.

- COA: dudi.coa, cca. Set 1 is rows, set 2 is columns. If set=12 (default), fac is not available and pch,cex, col can be defined differently for each set. scaling can be defined for rda (see scores.cca).

- DCOA: dudi.dec. Set 1 is rows, set 2 is columns. If set=12 (default), fac is not available and pch,cex, col can be defined differently for each set.

- PCIA: procuste. Set 1 is X, set 2 is Y. If set=12 (default), fac is not available and pch,cex, col can be defined differently for each set.

- DPCoA: dpcoa. Set 1 is categories, set 2 is collections. If set=12 (default), fac is not available and pch,cex, col can be defined differently for each set.

- RDA (or PCAIV): pcaiv, pcaivortho, rda. With rda, space 1 is constrained space, space 2 is unconstrained space. Only constrained space is available with pcaiv, the opposite for pcaivortho. scaling can be defined for rda (see scores.rda).

- db-RDA (or CAP): capscale. Space 1 is constrained space, space 2 is unconstrained space.

- CCA: cca, cca. With rda, space 1 is constrained space, space 2 is unconstrained space. Only constrained space is available with cca. Set 1 is rows, set 2 is columns. scaling can be defined for cca (see scores.cca).

- CCorA: CCorA, rcc. Space 1 is X, space 2 is Y. With rcc a third space is available, in which coordinates are means of X and Y coordinates.

- rCCorA: rcc. Space 1 is X, space 2 is Y, space 3 is a "common" space in which coordinates are means of X and Y coordinates.

- CIA: coinertia. Space 1 is X, space 2 is Y, space 3 is a "common" space where X and Y scores are normed. In space 3, set 1 is X and set 2 is Y. If set=12 in space 3 (default), fac is not available and pch,cex, col can be defined differently for each set.

- 2B-PLS: pls. Space 1 is X, space 2 is Y, space 3 is a "common" space in which coordinates are means of X and Y coordinates.

- 2B-sPLS: pls. Space 1 is X, space 2 is Y, space 3 is a "common" space in which coordinates are means of X and Y coordinates.

- Multilevel 2B-(s)PLS: pls. Space 1 is X, space 2 is Y, space 3 is a "common" space in which coordinates are means of X and Y coordinates. Package 'mixOmics' version >= 5.0.4.

- rGCCA: rgcca, wrapper.rgcca. Space can be 1 to n, the number of blocks (i.e. datasets).

- sGCCA: sgcca, wrapper.sgcca. Space can be 1 to n, the number of blocks (i.e. datasets).

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
data(iris)
PCA <- prcomp(iris[,1:4])
MVA.plot(PCA,"scores")
MVA.plot(PCA,"scores",fac=iris$Species,col=1:3,pch=15:17)
```

MVA.synt *Synthesis quality of multivariate analyses*

Description

Gives a simple estimator of the quality of the (descriptive) synthesis performed by a wide range of multivariate analyses.

Usage

```
MVA.synt(x, rows = 5)
```

Arguments

x	a multivariate analysis (see Details).
rows	maximum number of axes to print in the output.

Details

Many multivariate analyses are supported, from various packages. The list will progressively get longer (and additional criteria will be given).

- PCA: prcomp, princomp, dudi.pca, rda, pca, pca: % of total variance explained by each axis.

- sPCA: spca: % of total variance explained by each axis.

- IPCA: ipca: kurtosis of each axis.

- sIPCA: sipca: kurtosis of each axis.

- PCoA: cmdscale (with eig=TRUE), dudi.pco, wcmdscale (with eig=TRUE), capscale, pco, pcoa: % of total variance explained by each axis.

- nMDS: monoMDS, metaMDS, nmds, isoMDS: stress.

- LDA: lda, discrimin: % of intergroup variance explained by each axis.

- PLS-DA (PLS2 on a dummy-coded factor): plsda: % of intergroup variance explained by each axis.

- CPPLS: mvr: % of X and Y variances explained by each axis.

- PLSR: mvr, plsR: % of X and Y variances explained by each axis (only Y for the moment with plsR).

- PCR: mvr: % of X and Y variances explained by each axis.

- CDA: discrimin, discrimin.coa: % of intergroup variance explained by each axis.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
data(iris)
PCA <- prcomp(iris[,1:4])
MVA.synt(PCA,"scores")
```

MVA.test *Significance test based on cross (model) validation*

Description

Performs a permutational significance test based on cross (model) validation with different PLS and/or discriminant analyses. See MVA.cv and MVA.cmv for more details about how cross (model) validation is performed.

Usage

```
MVA.test(X, Y, cmv = FALSE, ncomp = 5, kout = 7, kinn = 8, model = c("PLSR",
  "CPPLS", "PLS-DA", "PPLS-DA", "LDA", "QDA", "PLS-DA/LDA", "PLS-DA/QDA",
  "PPLS-DA/LDA","PPLS-DA/QDA"), Q2diff = 0.05, lower = 0.5, upper = 0.5,
  Y.add = NULL, weights = rep(1, nrow(X)), set.prior = FALSE,
  crit.DA = c("plug-in", "predictive", "debiased"), p.method = "fdr",
  nperm = 999,...)
```

Arguments

X	a data frame of independent variables.
Y	the dependent variable(s): numeric vector, data frame of quantitative variables or factor.
cmv	a logical indicating if the values (Q2 or NMC) should be generated through cross-validation (classical K-fold process) or cross model validation (inner + outer loops).
ncomp	an integer giving the number of components to be used to generate all submodels (cross-validation) or the maximal number of components to be tested in the inner loop (cross model validation). Does not concern LDA and QDA.
kout	an integer giving the number of folds (cross-validation) or the number of folds in the outer loop (cross-model validation).
kinn	an integer giving the number of folds in the inner loop (cross model validation only).
model	the model to be fitted.
Q2diff	the threshold to be used if the number of components is chosen according to Q2 (cross model validation only).
lower	a vector of lower limits for power optimisation in CPPLS or PPLS-DA (see cppls.fit).
upper	a vector of upper limits for power optimisation in CPPLS or PPLS-DA (see cppls.fit).
Y.add	a vector or matrix of additional responses containing relevant information about the observations, in CPPLS or PPLS-DA (see cppls.fit).
weights	a vector of individual weights for the observations, in CPPLS or PPLS-DA (see cppls.fit).
set.prior	only used when a LDA or QDA is performed (coupled or not with a PLS model). If TRUE, the prior probabilities of class membership are defined according to the mean weight of individuals belonging to each class. If FALSE, prior probabilities are obtained from the data sets on which LDA/QDA models are built.
crit.DA	criterion used to predict class membership when a LDA or QDA is used. See predict.lda.
p.method	method for p-values correction. See help of p.adjust.
nperm	number of permutations.
...	other arguments to pass to plsr (PLSR, PLS-DA) or cppls (CPPLS, PPLS-DA).

Details

When Y consists in quantitative response(s), the null hypothesis is that each response is not predicted better than what would happen by chance. In this case, Q2 is used as the test statistic. When Y contains several responses, a p-value is computed for each response and p-values are corrected for multiple testing.

When Y is a factor, the null hypothesis is that the factor has no discriminant ability. In this case, the proportion of misclassified individuals (NMC) is used as the test statistic.

Whatever the response, the reference value of the test statistic is obtained by averaging 20 values coming from independently performed cross (model) validation on the original data.

The function deals with the limitted floating point precision, which can bias calculation of p-values based on a discrete test statistic distribution.

Value

method	a character string indicating the name of the test.
data.name	a character string giving the name(s) of the data, plus additional information.
statistic	the value of the test statistics.
permutations	the number of permutations.
p.value	the p-value of the test.
p.adjust.method	a character string giving the method for p-values correction.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

References

Westerhuis J, Hoefsloot HCJ, Smit S, Vis DJ, Smilde AK, van Velzen EJJ, van Duijnhoven JPM and van Dorsten FA (2012) Assessment of PLSDA cross validation. Metabolomics (2008) 4:81-89.

See Also

MVA.cv, MVA.cmv

Examples

```
require(pls)
require(MASS)

# PLSR
data(yarn)
## Not run: MVA.test(yarn$NIR,yarn$density,cmv=TRUE,model="PLSR")

# PPLS-DA coupled to LDA
data(mayonnaise)
## Not run: MVA.test(mayonnaise$NIR,factor(mayonnaise$oil.type),model="PPLS-DA/LDA",crit.inn="NMC")
```

MVA.trajplot *Trajectory plot of multivariate analyses*

Description

Displays a trajectory plot (*i.e.* a score plot with trajectories linking defined points) of a multivariate analysis.

Usage

```
MVA.trajplot(x, xax = 1, yax = 2, trajects, trajlab = NULL, scaling = 2,
  set = c(12, 1, 2), space = 1, xlab = NULL, ylab = NULL, main = NULL,
  pch = 16, cex = 1, trajlab.cex = 1, col = 1, lwd = 1, lty = 1,
  points = TRUE, allpoints = TRUE, arrows = TRUE, labels = NULL,
  main.pos = c("bottomleft", "topleft", "bottomright", "topright"),
  main.cex = 1.3, legend = FALSE, legend.pos = c("topleft", "topright",
  "bottomleft", "bottomright"), legend.title = NULL, legend.lab = NULL,
  legend.cex = 1, drawextaxes = TRUE, drawintaxes = TRUE, xlim = NULL,
  ylim = NULL)
```

Arguments

x	a multivariate analysis (see Details).
xax	the horizontal axis.
yax	the vertical axis. Cannot be NULL, only two-dimensional graphs can be drawn.
trajects	vector or list of vectors identifying trajectories. Each vector should give the number of the individuals to be linked, ordered from the first to the last one.
trajlab	optional traject labels.
scaling	type of scaling. Only available with some analyses performed with the vegan package. See Details of MVA.scoreplot.
set	scores to be displayed, when several sets are available (see Details of MVA.scoreplot). 12 (default) for both sets, 1 for rows or X, 2 for columns or Y.
space	scores to be displayed, when several spaces are available (see Details of MVA.scoreplot). space is the number of the space to be plotted.
xlab	legend of the horizontal axis. If NULL (default), automatic labels are used depending on the multivariate analysis.
ylab	legend of the vertical axis. If NULL (default), automatic labels are used depending on the multivariate analysis.
main	optional title of the graph.
pch	symbols used for points. Can be a vector giving one value per trajectory (and a last one for non-linked points if allpoints=TRUE).
cex	size of the labels. Can be a vector giving one value per trajectory (and a last one for non-linked points if allpoints=TRUE).

trajlab.cex	size of trajectory labels. Can be a vector giving one value per trajectory.
col	color(s) used for arrows and labels. If fac is not NULL, can be a vector of length one or a vector giving one value per group. Otherwise a vector of any length can be defined, which is recycled if necessary.
lwd	width of trajectory segments. Can be a vector giving one value per trajectory.
lty	type of trajectory segments. Can be a vector giving one value per trajectory.
points	logical indicating if points should be displayed. If FALSE, points are replaced with their corresponding label (defined by labels).
allpoints	logical indicating if points which do not belong to any trajectory should be drawn.
arrows	logical indicating if trajectories should be oriented with arrows.
labels	names of the individuals. If NULL (default), labels correspond to row names of the data used in the multivariate analysis.
main.pos	position of the title, if main is not NULL. Default to "bottomleft".
main.cex	size of the title, if main is not NULL.
legend	logical indicating if a legend should be added to the graph.
legend.pos	position of the legend, if legend is TRUE. Default to "topleft".
legend.title	optional title of the legend, if legend is TRUE.
legend.lab	legend labels, if legend is TRUE. If NULL and trajlab is defined, values of trajlab are used.
legend.cex	size of legend labels, if legend is TRUE.
drawextaxes	logical indicating if external axes should be drawn..
drawintaxes	logical indicating if internal axes should be drawn.
xlim	limits of the horizontal axis. If NULL, limits are computed automatically.
ylim	limits of the vertical axis. If NULL, limits are computed automatically.

Details

This function should not be use directly. Prefer the general MVA.plot, to which all arguments can be passed.

All multivariate analyses supported by MVA.scoreplot can be used for a paired plot.\

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
require(ade4)
data(olympic)
PCA <- dudi.pca(olympic$tab,scannf=FALSE)
MVA.plot(PCA,"traject",trajects=list(1:10,25:30),col=c(2,3,1),trajlab=c("T1","T2"))
```

overdisp.glmer *Estimation of overdispersion with glmer models*

Description

Estimates 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

Performs 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.

Details

Since a G-test is an approximate test, an exact test is preferable when the number of individuals is small (200 is a reasonable minimum). See fisher.multcomp in that case.

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, fisher.multcomp

Examples

```
x <- matrix(c(44,56,36,64,64,40),ncol=2,dimnames=list(c("Control","Treatment1","Treatment2"),
  c("Alive","Dead")),byrow=TRUE)
G.test(x)
pairwise.G.test(x)
```

---

pairwise.mood.medtest *Pairwise Mood's median tests*

Description

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

Usage

```
pairwise.mood.medtest(resp, fact, exact = NULL, p.method = "fdr")
```

Arguments

resp	response vector.
fact	grouping factor.
exact	a logical indicating whether exact p-values should be computed.
p.method	method for p-values correction. See help of p.adjust.

Details

If `exact=NULL`, Fisher's exact tests are used if the number of data values is < 200; otherwise chi-square tests are used (with Yates continuity correction).

Value

`method`	a character string indicating 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

`mood.medtest`

Examples

```
set.seed(0904)
response <- c(rnorm(10),rnorm(10,0.8),rnorm(10,2))
fact <- gl(3,10,labels=LETTERS[1:3])
mood.medtest(response~fact)
pairwise.mood.medtest(response,fact)
```

---

pairwise.perm.manova *Pairwise permutational MANOVAs*

Description

Performs pairwise comparisons between group levels with corrections for multiple testing. These pairwise comparisons are relevant after a permutational MANOVA, such as performed by `adonis`.

Usage

```
pairwise.perm.manova(resp, fact, test = c("Pillai", "Wilks",
  "Hotelling-Lawley", "Roy", "Spherical"), nperm = 999, p.method = "fdr")
```

Arguments

resp	response. Either a matrix (one column per variable; objects of class "data.frame" are accepted and internally converted into matrices) or a distance matrix.
fact	grouping factor.
test	choice of test statistic when resp is a matrix (see anova.mlm).
nperm	number of permutations.
p.method	method for p-values correction. See help of p.adjust.

Details

If resp is a matrix, a classical MANOVA is performed and the distribution of the (pseudo-)F is computed through permutations. The function deals with the limitted floating point precision, which can bias calculation of p-values based on a discrete test statistic distribution.

If resp is a distance matrix, adonis is used to perform each comparison.

Value

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

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

anova.mlm, adonis

Examples

```
require(vegan)
data(iris)

# Permutational MANOVA
adonis(iris[,1:4]~Species,data=iris,method="euclidian")

# Pairwise comparisons
# (not enough permutations here but faster to run)
pairwise.perm.manova(iris[,1:4],iris$Species,nperm=49)

# or
pairwise.perm.manova(dist(iris[,1:4],"euclidian"),iris$Species,nperm=49)
```

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

Description

Performs 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.

Details

The function deals with the limitted floating point precision, which can bias calculation of p-values based on a discrete test statistic distribution.

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.to.groups *Letter summary of similarities and differences*

Description

Converts a pairwise-comparison matrix into a character-based display in which common characters identify levels or groups that are not significantly different. The function is based on `multcompLetters`.

Usage

```
pairwise.to.groups(pairwise.test, component = "p.value", alpha = 0.05)
```

Arguments

`pairwise.test`	a list, typically of class `"pairwise.htest"`.
`component`	name of the component of `pairwise.test` containing the pairwise-comparison matrix. The default `"p.value"` corresponds to all objects of class `"pairwise.htest"`.
`alpha`	significance threshold.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`multcompLetters` for confidence intervals (and tests).

Examples

```
attach(airquality)
Month <- factor(Month,labels=month.abb[5:9])
(test <- pairwise.t.test(Ozone, Month))
pairwise.to.groups(test)
detach()
```

pcor *(Semi-)Partial correlation*

Description

Computes the (semi-)partial correlation of x and y, controlling for z.

Usage

```
pcor(x, y, z, semi = FALSE, use = "complete.obs", method = c("pearson",
  "kendall", "spearman"))
```

Arguments

x	a numeric vector.
y	a numeric vector.
z	a numeric vector, matrix, data frame or list giving the controlling variables. For matrices, variables must be placed in columns.
semi	logical. If TRUE the semi-partial correlation coefficient is computed. In that case only y is controlled for z.
use	same as use of cor.
method	same as method of cor.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

pcor.test for confidence intervals (and tests).

Examples

```
set.seed(1444)
x <- 1:30
y <- 1:30+rnorm(30,0,2)
z1 <- runif(30,0,4)
z2 <- 30:1+rnorm(30,03)
pcor(x,y,z1)
pcor(x,y,list(z1,z2))
```

`pcor.test` *Tests for (semi-)partial association/correlation between paired samples*

Description

Tests for (semi-)partial association between paired samples while controlling for other variables, using one of Pearson's product moment correlation coefficient or Spearman's *rho*.

Usage

```
pcor.test(x, y, z, semi = FALSE, conf.level = 0.95, nrep = 1000,
  method = c("pearson", "spearman"))
```

Arguments

`x`	a numeric vector.
`y`	a numeric vector.
`z`	a numeric vector, matrix, data frame or list giving the controlling variables. For matrices, variables must be placed in columns.
`semi`	logical. If `TRUE` the semi-partial correlation coefficient is computed and tested. In that case only `y` is controlled for `z`.
`conf.level`	confidence level for confidence interval..
`nrep`	number of replicates for computation of the confidence interval of a Spearman's rank correlation coefficient (by bootstraping).
`method`	a character string indicating which correlation coefficient is to be used for the test. One of "pearson" or "spearman".

Details

If `method` is `"pearson"` and if there are at least 4+k complete series of observation (where k is the number of controlling variables), an asymptotic confidence interval of the correlation coefficient is given based on Fisher's Z transform.

If `method` is `"spearman"`, the p-value is computed through the AS89 algorithm if the number of complete series of observation is less than 10, otherwise via the asymptotic t approximation (in both cases the `pspearman` function is used). A confidence interval of the correlation coefficient, computed by bootstraping, is given.

Value

`data.name`	a character string giving the name(s) of the data.
`alternative`	a character string describing the alternative hypothesis, always two-sided.
`method`	a character string indicating how the association was measured.
`conf.int`	a condidence interval for the measure of association.

statistic	the value of the test statistic.
parameter	the degrees of freedom of the test (only for a Pearson's correlation coefficient).
p.value	the p-value of the test.
estimate	the estimated measure of association, with name "cor" or "rho" corresponding to the method employed.
null.value	he value of the association measure under the null hypothesis, always 0.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

pcor

Examples

```
set.seed(1444)
x <- 1:30
y <- 1:30+rnorm(30,0,2)
z1 <- runif(30,0,4)
z2 <- 30:1+rnorm(30,03)
pcor.test(x,y,z1)
pcor.test(x,y,list(z1,z2))
```

---

perm.anova *Permutational Analysis of Variance*

Description

Performs 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.

The function deals with the limitted floating point precision, which can bias calculation of p-values based on a discrete test statistic distribution.

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

Performs 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.

Details

The function deals with the limitted floating point precision, which can bias calculation of p-values based on a discrete test statistic distribution.

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

Performs 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.

Details

The function deals with the limitted floating point precision, which can bias calculation of p-values based on a discrete test statistic distribution.

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

Performs a permutational Student's t-test.

Usage

```
perm.t.test(x, ...)

## Default S3 method:
perm.t.test(x, y, paired = FALSE, ...)

## S3 method for class 'formula'
perm.t.test(formula, data, alternative = c("two.sided", "less", "greater"),
  paired = FALSE, nperm = 999, ...)
```

Arguments

`x`	a numeric vector of data values.
`y`	a numeric vector of data values.
`paired`	a logical indicating whether you want a paired t-test.
`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"`.
`nperm`	number of permutations.
`...`	further arguments to be passed to or from other methods.

Details

The function deals with the limitted floating point precision, which can bias calculation of p-values based on a discrete test statistic distribution.

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])

# Not enough permutations here but faster to run

# Unpaired test
perm.t.test(response~fact,nperm=49)

# Paired test
perm.t.test(response~fact,paired=TRUE,nperm=49)
```

---

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

Description

Performs a permutational F test to compare two variances.

Usage

```
perm.var.test(x, ...)

## Default S3 method:
perm.var.test(x, y, ...)

## S3 method for class 'formula'
perm.var.test(formula, data, alternative = c("two.sided", "less",
  "greater"), nperm = 999, ...)
```

Arguments

x	a numeric vector of data values.
y	a numeric vector of data values.
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".
nperm	number of permutations.
...	further arguments to be passed to or from other methods.

Details

The function deals with the limitted floating point precision, which can bias calculation of p-values based on a discrete test statistic distribution.

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, always 1.

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)
```

plotresid *Simple analysis of model residuals*

Description

Plots 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, mlm and manova), lmer, glmer, glmmadmb, lme, gls, nls, nlsList, survreg and least.rect models.

Usage

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

Arguments

model	an object of class "lm", "merMod", "glmmadmb", "lme", "gls", "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, gls, 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 mlm or 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, gls, nls, nlsList, survreg, least.rect, qresiduals, qqPlot, shapiro.test, mqqnorm, mshapiro.test

plotsurvivors *Survivor curve*

Description

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

Description

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

Usage

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

Arguments

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

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)
```

---

predict.coadisc *Predict method for CDA*

Description

Predicts class of the grouping factor based on a Correspondence Discriminant Analysis (performed using discrimin.coa).

Usage

```
## S3 method for class 'coadisc'
predict(object, newdata, dim=object$nf, method = c("mahalanobis", "euclidian"), ...)
```

Arguments

object	object of class inheriting from "coadisc".
newdata	contingency table (either a "matrix", "table" or "data.frame" object) giving new individuals (one row per individual).
dim	number of dimensions to be used for prediction.

method	distance metric to be used for prediction. In all cases the predicted class corresponds to the minimum distance between the new individual and the centroid of each class. Default is Mahalanobis distance.
...	further arguments to be passed to or from other methods.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

discrimin.coa

Examples

```
require(ade4)
data(perthi02)

CDA <- discrimin.coa(perthi02$tab,perthi02$cla,scan=FALSE)
new <- matrix(c(17,45,32,17,17,52,28,29,6,10,7,7,7,5,10,4,37,34,23,9),ncol=20)
predict(CDA,new)
```

---

predict.MVA.cv *Predict method for cross-validated submodels*

Description

Predicts response based on submodels generated by cross (model) validation. For regression models (PLSR and CPPLS), the predicted value is given with its confidence interval. For discriminant analyses, the predicted class is given with its probability (computed from the values predicted by all submodels).

Usage

```
## S3 method for class 'MVA.cv'
predict(object, newdata, conf.level = 0.95, crit.DA = c("plug-in", "predictive",
  "debiased"), ...)
## S3 method for class 'MVA.cmv'
predict(object, newdata, conf.level = 0.95, crit.DA = c("plug-in", "predictive",
  "debiased"), ...)
```

Arguments

object	object of class inheriting from "MVA.cv" or "MVA.cmv".
newdata	vector, matrix or data frame giving new individuals (one row per individual).
conf.level	confidence level for prediction of a quantitative response.
crit.DA	criterion used to predict class membership when a LDA or QDA is used. See predict.lda.
...	further arguments to be passed to or from other methods.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

MVA.cv, MVA.cmv

---

prop.bin.multcomp *Pairwise comparisons after a test for given probabilities*

Description

Performs pairwise comparisons after a global test for given response probabilities (i.e. when the response variable is a binary variable), by using exact binomial tests. The function is in fact a wrapper to pairwise comparisons of proportions to given values on a contingency table.

Usage

```
prop.bin.multcomp(formula, data, p, 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).
p	theoretical probabilities.
p.method	method for p-values correction. See help of p.adjust.

Details

If the response is a 0/1 variable, the probability of the '1' group is tested. In any other cases, the response is transformed into a factor and the probability of the second level is tested.

Value

method	name of the test.
data.name	a character string giving the name(s) of the data.
observed	observed probabilities.
expected	expected probabilities.
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.multcomp, chisq.theo.bintest

Examples

```
response <- c(rep(0:1,c(40,60)),rep(0:1,c(55,45)),rep(0:1,c(65,35)))
fact <- gl(3,100,labels=LETTERS[1:3])
p.theo <- c(0.5,0.45,0.2)
chisq.theo.bintest(response~fact,p=p.theo)
prop.bin.multcomp(response~fact,p=p.theo)
```

---

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

Description

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

Usage

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

Arguments

x	contingency 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),200,replace=TRUE)
populations <- sample(LETTERS[1:3],200,replace=TRUE)
tab.cont <- table(populations,proportions)
p.theo <- c(0.4,0.5,0.7)
prop.test(tab.cont,p=p.theo)
prop.multcomp(tab.cont,p=p.theo)
```

---

prop.multinom *Proportions and standard errors*

Description

Computes proportions (and their standard errors) when the number of classes is >= 2, based on predicted values of a model. The function is intended to be used parallel to a multinomial log-linear model.

Usage

```
prop.multinom(x)
```

Arguments

`x`	either a factor or a matrix with K columns giving the counts for each of the K classes.

Details

The proportions can be computed through the `predict` function applied on a multinomial log-linear model (see `multinom`). However, standard errors (or confidence intervals) cannot be obtained by this way. The present function uses differents GLMs (in each case considering one category vs. the sum of all others) to obtain proportions and standard errors. Overdispersion is taken into account by default, using a quasibinomial law in all GLMs built.

Value

`probs`	the calculated proportions.
`se`	the calculated standard errors.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`multinom`, `glm`

Examples

```
response <- data.frame(A=c(2,2,4,0,2,14,6,16,0,0),
     B=c(2,0,0,0,6,2,10,6,0,0),
     C=c(12,6,0,6,2,0,0,0,0,0),
     D=c(0,0,0,14,0,0,0,0,2,0),
     E=c(0,0,0,0,0,0,0,0,16,15))
prop.multinom(response)
```

`prop.multinom.test` *Wald test for comparison of proportions*

Description

Performs pairwise comparisons of proportions when the number of classes is >= 2 with corrections for multiple testing.

Usage

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

Arguments

`x`	either a factor or a matrix with K columns giving the counts for each of the K classes.
`p.method`	method for p-values correction. See help of `p.adjust`.

Details

The function builds multinomial log-linear models (using multinom) and applies Wald tests to compare the intercepts to 0. All necessary models (each time using a different reference level/class) are built to get p-values of all possible comparisons among levels/classes.

Value

method	a character string indicating the 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.
z.tab	table of z values.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

multinom, binom.test

Examples

```
response <- factor(rep(LETTERS[1:4],c(20,40,42,13)))
table(response)/length(response)
prop.multinom.test(response)
```

---

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

Description

Extracts 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.
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).

Details

A factor named `cut` must have been called in `lsmeans`, to compute LSMeans per cut point (i.e. rating). Additionally, the argument `mode` of `lsmeans` must have been set to "linear.predictor". Finally, the call to `lsmeans` is typically like `lsmeans(model,~factor|cut,mode="linear.predictor")` where `factor` is the factor (or interaction) giving levels for which LSMeans have to be computed.

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,mode="linear.predictor")

# 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

Computes 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

Computes 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")
```

---

scat.cr *"Correlation" of variables to axes in MCA or mix analyses*

Description

Represents the "correlation" of variables to axes in a MCA (from dudi.acm) or a mix analysis (from dudi.hillsmith or dudi.mix).

Usage

```
scat.cr(dudi.obj, axis = 1)
```

Arguments

dudi.obj	object obtained from dudi.acm, dudi.hillsmith or dudi.mix.
axis	axis to be represented (the first by default).

Details

For quantitative variables, the squared correlation coefficient is displayed. For ordered factors, the squared multiple correlation coefficient is displayed. For unordered factors, the correlation ratio is displayed.

Author(s)

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

See Also

dudi.acm, dudi.hillsmith, dudi.mix

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.cr(amix)
```

---

se *Standard error of the mean*

Description

Computes 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

Generates 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

Computes 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 of the interval.

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)
```

---

splitf *Divide into groups respecting relative proportions*

Description

Divides a data frame randomly, but respecting the relative proportions of levels of a factor in the original data frame. Each subset has roughly the same number of individuals, and the same relative proportions in respect to levels of the given factor.

Usage

```
splitf(set, fac, k)
```

Arguments

set	a data frame containing values to be divided into groups.
fac	a reference factor giving the relative proportions to be respected in each subset of set.
k	an integer giving the number of subsets to be generated.

Value

A list of subsets of set.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

split

Examples

```
data(iris)
iris2 <- iris[c(1:50,51:80,101:120),]

# Proportions to be respected
table(iris2$Species)/nrow(iris2)

# Splitting
result <- splitf(iris2,iris2$Species,3)

# All subsets have the same size
lapply(result,nrow)

# And respect the initial proportions
lapply(result,function(x) table(x$Species)/nrow(x))
```

---

user.cont *User defined contrasts for LSMeans*

Description

Returns 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)
```

---

wald.ptheo.test *Wald test for comparison of a proportion to a theoretical value*

Description

Performs a Wald test for comparison of a proportion to a theoretical value.

Usage

```
wald.ptheo.test(y, blocks = NULL, p = 0.5)
```

Arguments

`y`	either a binary response (numeric vector or factor, with only two possible values except NA) or a two-column matrix with the columns giving the numbers of successes (left) and failures (right).
`blocks`	optional blocking (random) factor.
`p`	hypothesized probability of success.

Details

The function builds a logistic (mixed) regression and applies a Wald test to compare the estimated value of the intercept to its theoretical value under H0. Eventual overdispersion is taken into account, by using a quasi-binomial law in case of no blocks or by introducing an individual-level random factor if blocks are present.

If the response is a 0/1 vector, the probability of the '1' group is tested. With other vectors, the response is transformed into a factor and the probability of the second level is tested.

If the response is a two-column matrix, the probability of the left column is tested.

If the reponse is a vector and no blocking factor is present, the exact binomial test performed by `binom.test` should be preferred since it is an exact test, whereas the Wald test is an approximate test.

Value

`method`	name of the test.
`data.name`	a character string giving the name(s) of the data.
`statistic`	test statistics of the test.
`p.value`	p-value of the test.
`estimate`	the estimated proportion (calculated without taking into account the blocking factor, if present).
`alternative`	a character string describing the alternative hypothesis, always `"two.sided"`.
`null.value`	the value of the proportion under the null hypothesis.
`parameter`	the degrees of freedom for the t-statistic, only whith overdispersion and no blocks.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

See Also

`binom.test`, `glm`, `glmer`

Examples

```
set.seed(2006)
response <- sample(0:1,60,replace=TRUE)

# Comparison to p=0.5
wald.ptheo.test(response)

# Comparison to p=0.8
wald.ptheo.test(response,p=0.8)

# With a blocking factor

require(lme4)
```

```
blocks <- gl(3,20)
wald.ptheo.test(response,blocks)
```

`wilcox.paired.multcomp` *Non parametric pairwise comparisons for paired data*

Description

Performs 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.signtest` *Wilcoxon sign test*

Description

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

Usage

```
wilcox.signtest(x, ...)

## Default S3 method:
wilcox.signtest(x, y = NULL, mu = 0, conf.level = 0.95, ...)

## S3 method for class 'formula'
wilcox.signtest(formula, data, subset, ...)
```

Arguments

`x`	a numeric vector of data values.
`y`	an optional numeric vector of data values (for paired two-sample test).
`mu`	theoretical median (one-sample test) or theoretical median of `x-y` differences.
`conf.level`	confidence level of the interval.
`formula`	a formula of the form `a ~ b`, where `a` and `b` give the data values and corresponding groups.
`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.
`...`	further arguments to be passed to or from other methods.

Details

If zeroes (i.e. null differences with `mu`) are present, the median of the data different from `mu` is tested in the one-sample situation; the median of the `x-y` differences different from `mu` in the two-sample situation.

Value

`method`	a character string indicating the name of the test.
`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	the p-value of the test.
alternative	a character string giving the alternative hypothesis, always "two.sided"
estimate	the estimated median or median of x-y differences, depending on the test performed.
conf.int	a confidence interval for the median tested.

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)
```

---

wmean *Weighted arithmetic mean*

Description

Computes the weighted arithmetic mean of a vector.

Usage

```
wmean(x, w = rep(1, length(x)), na.rm = TRUE)
```

Arguments

x	numeric vector.
w	numeric vector of weights.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.

Author(s)

Maxime Hervé <mx.herve@gmail.com>

Examples

```
mean(1:10)
wmean(1:10,w=10:1)
```

Index