Package 'bmisc'

October 4, 2011

Type Package

Title Miscellaneous functions

Version 0.2-13

Author Benoit Bruneau

Maintainer Benoit Bruneau <benoit.bruneau1@gmail.com>

URL http://r-forge.r-project.org/projects/bmisc/

Description

This package has different functions that I have accumulated with time. This is the Alpha version.

Depends car, lattice, zoo, robustbase, methods, tcltk

License LGPL >= 3.0

Repository R-Forge

Repository/R-Forge/Project bmisc

Repository/R-Forge/Revision 114

LazyLoad yes

R topics documented:

```
att.strp                    Attibute stripper
```

Description

Strips an object of its attributes

Usage

```
att.strp(obj)
```

Arguments

```
obj             the name of an object (vector, matrix, data.frame, array or list)
```

Details

This function strips an object of its attributes. In the case of a `vector`, all attributes are removed.
For a `matrix` or an `array`, only `c('dim','dimnames')` are kept. When `att.strp` is used
on a `data.frame`, all attributes of the variables are striped and only `c('names','row.names','na.action',`
`'class')` are kept for the `data.frame` object.

Value

returns an object of the same `class` as `obj`.

Author(s)

Benoit Bruneau

Examples

```
######################################
#   Creating different objects        #
#     with added attributes (label)   #
######################################

### numerical vector ###
x <- 1:10
attr(x,"label") <- "test1"
attributes(x)

### data frame ###
z=data.frame(x,x)
attr(z,"labels") <- "test2"
attributes(z)
attributes(z[,1])
attributes(z[,2])

### array ###
y=array(x,c(2,2,2))
attr(y,"labels") <- "test3"
attributes(y)
```

```
attributes(y[,,1])
attributes(y[,,2])

### list containing the vector, ###
### data frame and array        ###
u=list(x,z,y)
attr(u,"labels") <- "test4"
attributes(u)
attributes(u[[1]])
attributes(u[[2]])
attributes(u[[3]])

######################################
#        attribute stripping        #
######################################
x2=att.strp(x)
z2=att.strp(z)
y2=att.strp(y)
u2=att.strp(u)

######################################
#   verification of the attributes   #
#     for all stripped objects       #
######################################

### numerical vector ###
attributes(x2)

### data frame ###
attributes(z2)
attributes(z2[,1])
attributes(z2[,2])

### array ###
attributes(y2)
attributes(y2[,,1])
attributes(y2[,,2])

### list containing the vector, ###
### data frame and array        ###
attributes(u2)
attributes(u2[[1]])         # vector in the list

attributes(u2[[2]])         # data frame in the list
attributes(u2[[2]][,1])     # data frame in the list
attributes(u2[[2]][,2])     # data frame in the list

attributes(u2[[3]])         # array in the list
attributes(u2[[3]][,,1])    # array in the list
attributes(u2[[3]][,,2])    # array in the list
```

`bmisc`	*Miscellaneous functions*

Description

This package has different functions that I have accumulated with time. This is the Alpha version.

Format

Package:	bmisc
Type:	Package
Version:	0.2-13
Date:	2011-10-03
License:	LGPL >= 3.0
LazyLoad:	yes

Details

For pdf version of the help, write `vignette("bmisc")`.

Author(s)

Benoit Bruneau

Maintainer: Benoit Bruneau <benoit.bruneau1@gmail.com>

---

ceiling.lg   *ceiling largest*

Description

Ceiling to largest digit

Usage

```
ceiling.lg(x)
```

Arguments

x   Numeric vector

Details

Gives the ceiling to largest digit (i.e., 54 -> 60).

Examples

```
ceiling.lg(250)
ceiling.lg(25000000)
```

---

clean *Clean a Data Frame*

Description

Cleans a `data.frame` from a starting point with a defined threshold

Usage

```
clean(data= x,  col.start =1,  min.val=NULL)
```

Arguments

`data`	then name of the `data.frame`
`col.start`	indicate the columns from which to start reading
`min.val`	`numeric`. Read details

Details

`min.val` is the minimum value accepted in a column. Colomns with this value or higher will be kept in the `data.frame`.

More will be added to this function.

Value

returns the data.frame with the clean columns

Author(s)

Benoit Bruneau

Examples

```
x=rnorm(50 , 20,  12)
y=runif(50 )
z=rpois(50,  3)
v=x*y/z
t=z*v
pp=data.frame(aa=x,  bb=y, cc=v,  dd=z,  ee=t)
summary(pp)

pp1 = clean(pp, min.val=0.06)
```

---

corr.perm   *Pearson Correlation by Permutation*

Description

Tests the Pearson correlation estimate (r) by use of permutation

Usage

```
corr.perm(x,y,nperm=999)
```

Arguments

`x,y`	Two vectors of same length used for correlation analysis
`nperm`	Number of permutations (default = 999)

Value

`Correlation`	Pearson r
`t.stat`	Calculated test statistic (t)
`No.perm`	number of permutations
`P.perm`	pvalue estimated by permutations
`P.para`	parametric pvalue estimated
`inf`	inferior limit of the confidence interval
`sup`	superior limit of the confidence interval
`df`	degree of freedom

Examples

```
x <- rnorm(50,0,1)
y <- runif(50,0,1)*x
toto = corr.perm(x, y)
```

---

cv *Coefficient of Variation (CV)*

Usage

```
cv(x, na.rm=T)
```

Arguments

x   an R object (vector, matrix,...)

na.rm   a logical value indicating whether NA values should be stripped before the com-
                  putation proceeds

Details

The coefficient of variation (CV) is the ratio of the standard deviation to the mean. The CV is
defined for the absolute value of the mean to ensure it is always positive.

Examples

```
x=rnorm(50)
cv(x)
```

day *day*

Description

Day of year as decimal number (001-366).

Usage

```
day(x)
```

Arguments

x

Examples

```
# will soon be available
```

Errbar   *Error Bars*

Description

Adds error bars on a plot

Usage

```
Errbar(x, y, xinf=NULL, xsup=NULL, yinf=NULL, ysup=NULL, xint=NULL,
       xint=NULL, cap=0.05,...)
```

Arguments

x	numeric vector
y	numeric vector
xinf, xsup	numeric vectors containing the upper (xsup) and/or lower (xinf) limits of the confidence interval for x-axis values.
yinf, ysup	numeric vectors containing the upper (ysup) and/or lower (yinf) limit of the confidence interval for y-axis values.
xint	numeric vectors containing the confidence intervals for x-axis values.
yint	numeric vectors containing the confidence intervals for y-axis values.
...	additional graphical arguments ([par](#)) such as col, lty, lwd and/or arguments for [arrows](#) .

Details

If xint and/or yint are defined, individually defined limits (ie. xinf, xsup, yinf, ysup) are not used.

See Also

[arrows](#), [par](#)

Examples

```
x <- 1:10
y <- x + rnorm(10)

yci <- runif(10)
xci <- runif(10)

plot(x,y, ylim=c(min(y-yci),max(y+yci)))
Errbar( x, y, yint=yci)

plot(x,y, xlim=c(min(x-xci),max(x+xci)))
Errbar( x, y, xint=xci )

plot(x,y, ylim=c(min(y-yci),max(y+yci)), xlim=c(min(x-xci),max(x+xci)))
Errbar( x, y, yint=yci, xint=xci )
```

```
# Gives an Error message
#plot(x,y, ylim=c(min(y-yci),max(y+yci)))  ## adds the yint and gives
#Errbar( x, y, ysup=1, yint=yci)           ## an error message for the ysup
```

---

fct *Print bmisc functions*

Description

Print all functions of bmisc package

Usage

```
fct()
```

---

find.beta *Logistic curve parameter estimates*

Description

Finds the parameters of a logistic curve for given inflection points.

Usage

```
find.beta(beta=0.5, minv, maxv, prop=0.01)
```

Arguments

beta	stating value of beta. Default is 0.5.
minv	the minimum value on the abscissa is the first inflection point.
maxv	the maximum value on the abscissa is the second inflection point.
prop	the proportion of the instantaneous slope at 50% probability that should be used to define the position of the inflection points of the curve. Default is 0.01.

Details

A logistic curve is defined by:

$$y = 1/(1 + e^{-(\alpha + \beta x)}) <==> 1/(1 + e^{-\beta(x - x_{50})})$$

Depending on the sign of β, the curve will be negative or positive.

Value

find.beta() returns a data.frame with the following columns:

beta	the estimated β for the given inflection points.
alpha	the estimated α for the given inflection points.
x50	the value of x when y is 0.5 (x_{50}).
angle.x50	the angle of the instantaneous slope at x_{50}.
min	the value of the first inflection point.
max	the value of the second inflection point.
angle.infl	the angle of the instantaneous slope at the inflection points.

Author(s)

Benoit Bruneau

See Also

deriv

Examples

```
res1=find.beta(beta=0.1,minv=1000,maxv=1700, prop=0.01)
res2=find.beta(beta=0.1,minv=500,maxv=1700, prop=0.01)
par(mfrow=c(2,1))

xlim=c(0,res1$max+((res1$max-0)*0.2))

curve(1/(1+exp(-res1$beta*(x-res1$x50))), xlim=xlim, ylab="Probability",lwd=2)
abline(v=c(res1$max,res1$min,res1$x50), col=c("red","red","blue"))
lines(x=c(-500,res1$x50),y=c(0.5,0.5), lty=2, col=gray(0.4))
text(x=0,y=0.5,labels="x50", pos=3, col=gray(0.4))

curve(1/(1+exp(-res2$beta*(x-res2$x50))), xlim=xlim,ylab="Probability",lwd=2)
abline(v=c(res2$max,res2$min,res2$x50),, col=c("red","red","blue"))
lines(x=c(-500,res2$x50),y=c(0.5,0.5), lty=2, col=gray(0.4))
text(x=0,y=0.5,labels="x50", pos=3, col=gray(0.4))
```

---

format.hms   *Format seconds into hours*

Description

Transforms time format

Usage

```
format.hms(sec)
```

Arguments

sec   time expressed in seconds

Value

returns hrs:min:sec

Examples

```
format.hms(20000)
```

---

gam.Check   *Some diagnostics for a fitted gam model*

Description

Takes a fitted gam object produced by gam() and produces some diagnostic information about the fitting procedure and results. The default is to produce 4 residual plots, and some information about the convergence of the smoothness selection optimization.

Usage

```
gam.Check(b,...)
## Default S3 method:
gam.Check(b,
         main=c("Normal Q-Q Plot","Resids vs. Linear Pred.",
         "Histogram of Residuals","Response vs. Fitted Values"),

         xlab=c("Theorical Quantiles", "Linear Predictor",
         "Residuals","Fitted Values"),

         ylab= c("Sample Quantiles","Residuals","Frequency",
         "Response"),

         text=NULL, args.histplot=NULL, ...))
```

Arguments

b	a fitted gam object as produced by gam().
main	a character vector containing the four titles to be used.
xlab	a character vector containing the four x labels to be used.
ylab	a character vector containing the four y labels to be used.
text	a character or expression vector specifying the text to be written.
args.histplot	
	list of additional arguments to pass to histplot()
...	additional text and graphical parameters (see par, mtext)

Details

This function plots 4 standard diagnostic plots, and some other convergence diagnostics. Usually the 4 plots are various residual plots. The printed information relates to the optimization used to select smoothing parameters. For the default optimization methods the information is summarized in a readable way, but for other optimization methods, whatever is returned by way of convergence diagnostics is simply printed.

This is a modified version of gam.check from mgcv-package so that main titles, x labels and y labels can be customized.

References

Wood S.N. (2006) Generalized Additive Models: An Introduction with R. Chapman and Hall/CRC Press.

Examples

```
library(mgcv)
set.seed(0)
dat <- gamSim(1,n=200)
b<-gam(y~s(x0)+s(x1)+s(x2)+s(x3),data=dat)
plot(b,pages=1)


gam.check(b)

gam.check(b, main=c("A","B","C","D"))
```

get.partial.etas   *get partial etas*

Usage

```
get.partial.etas(model)
```

Arguments

```
model
```

Examples

```
# will soon be available
```

---

histplot	*histplot*

Usage

```
histplot(dat, breaks="Sturges", barc="steelblue", borc="white",
         fit.norm=TRUE, lcol="brown", stat=NULL,
         stat.lab=c("Mean","Median"), box=TRUE, rug=TRUE,
         main,...)
```

Arguments

dat   one of:

- a numeric vector
- an object of class c('norm','lm','aov','glm','gam') resulting from a calls to c(norm.test,lm,aov,glm,gam)

breaks   one of:

- a vector giving the breakpoints between histogram cells,
- a single number giving the number of cells for the histogram,
- a character string naming an algorithm to compute the number of cells (see 'Details'),
- a function to compute the number of cells.

In the last three cases the number is a suggestion only.

barc   a color to be used to fill the bars.

borc   a color to be used for the borders the bars.

fit.norm   a logical variable indicating whether to fit a normal density curve (TRUE) or not (FALSE).

lcol   color of the normal density curve

stat   the statistic to add on the graph. One of (c("all","mean","median")). Default is NULL.

stat.lab   a character vector with the labels for the estimated mean and/or median. Default is c("Mean","Median").

rug   a logical variable indicating whether to superpose a rug (TRUE) or not (FALSE).

main   the main title of the graph

...   additional arguments to be passed to plot (see par)

Details

The default for breaks is "Sturges": see nclass.Sturges. Other names for which algorithms are supplied are "Scott" and "FD" / "Freedman-Diaconis" (with corresponding functions nclass.scott and nclass.FD). Alternatively, a function can be supplied which will compute the intended number of breaks as a function of x.

See Also

hist

Examples

```
x=rnorm(50)
histplot(x)

norm.x=norm.test(x)
histplot(norm.x)
```

---

inv.pred   *Inverse Predictions with SE*

Usage

```
inv.pred( object, cf=1:2, y )
```

Arguments

object   an object of class `c('lm','glm')` resulting from a calls to `c(lm,glm)`

cf   the linear coefficients (`'intercept'`,`'slope'`) to be used.

y   the `y` value for which `x` will be estimated with it's standard error.

Details

More to come.

Author(s)

Benoit Bruneau

is.even *is even*

Description

Identifies if a value is even or not

Usage

```
is.even(x)
```

Arguments

x   numeric vector

Details

Will returns TRUE if roundup(x) is an even number.

Value

logical

See Also

is.odd

Examples

```
is.even(5)
is.even(6)
```

`is.odd`   *is odd*

Description

Identifies if a value is odd or not

Usage

```
is.odd(x)
```

Arguments

x   numeric vector

Details

Will returns TRUE if [roundup](x) is an odd number.

Value

logical

See Also

is.even

Examples

```
is.odd(5)
is.odd(6)
```

last *last*

Usage

```
last(x)
```

Arguments

x

Examples

```
# will soon be available
```

---

lev *Levene type tests*

Description

Testing for homoscedasticity after an ANOVA.

Usage

```
lev(y, ...)
## S3 method for class 'lm'
lev(y, trim.alpha=0.1, type=c("abs","sq"))
## S3 method for class 'formula'
lev(y, data, trim.alpha=0.1, type=c("abs","sq"))
## Default S3 method:
lev(y, group, data=NULL, trim.alpha=0.1, type=c("abs","sq"))
```

Arguments

y	response variable for the default method, lm class object for the lm method or formula class object for the formula methode. If y is a linear-model object or a formula, the variables on the right-hand-side of the model must all be factors and must be completely crossed.
group	for the default method, factor (concatenated factor when multiple factors).
data	data.frame containing the dependant variable and the factor(s).
trim.alpha	Alpha level (percentiles) trimming the data on which the mean will be evaluated.
type	Type of transformation made on the residuals. Either "abs" for absolute values or "sq" for squared values

Details

When using the lm method, data doesn't need to be defined. When using the formula or default methods, data can be defined if the data used is in a data.frame.

When group is manually defined in the default method, use paste(x,y,z) or interaction(x,y,z) form where "x", "y" and "z" are factors.

Value

lev returns an object of class(S4) "lev".

An object of class "ypr" has the the following slots:

statistics	statistics for each test.
p.value	p-value for each test.
residuals	residuals for each test (r-scores for O'Briens test.

References

Brown, Morton B. and Alan B. Forsythe (1974). Robust Tests for Equality of Variances, J. Am. Statistical Assoc., 69, 364-367.

Fligner, Michael A. and Timothy J. Killeen (1976). Distribution-free two sample tests for scale. J. Am. Statistical Assoc., 71: 210-213.

Levene, Howard (1960). "Robust tests for equality of variances". In Ingram Olkin, Harold Hotelling, et alia. Stanford University Press. pp. 278-292.

O'Brien, R.G. (1981). A simple test for variance effects in experimental designs. Psychological Bulletin, 89, 570-574.

Conover, William J., Johnson, Mark E., and Myrle M. Johnson (1981). A comparative study of tests for homogeneity of variances, with applications to the outer continental shelf bidding data. Technometrics 23, 351-361.

See Also

fligner.test

Examples

```
z=data.frame( yy=c(rep("c",50),rep("d",50)),
              x=c(rnorm(50),rnorm(50,10)),
              s=rep(c(rep("a",25),rep("b",25)),2),
              qq=rep(c(rep("w",10),rep("t",10)),5))

mod=lm(x~yy*qq*s, data=z)
formula= x~yy*qq*s

lev(y=x, group= paste(yy,qq,s), data=z, type="abs")
lev(y=x, group= paste(yy,qq,s), data=z, type="sq")

lev(y=x, group= interaction(yy,qq,s), data=z)

lev(y=formula, data=z)
lev(mod)
```

`lib.code`	*Retreives the code for* `lib()`.

Description

Will print in the R windows the code for `lib()` **(READ DETAILS)**.

Usage

```
lib.code()
lib(pack, install=TRUE, load=TRUE, quietly=TRUE,
    warn.conflicts=FALSE)
```

Arguments

`pack`   Character vector specifying which package(s) to load/install.

Details

USE `lib.code()` TO GET THE CODE FOR THE FUNCTION `lib()`.

`lib.code()` prints in R the code for `lib()`. Copy and paste the code for `lib()` in the file
`"C:/Program Files/R/R-2.12.1/etc/Rprofile.site"` (Windows) or `"~/.Rprofile"`
(Mac).

`lib()` will load packages named in a charcater vector. If install is `TRUE`, packages not yet installed
will be installed.

Author(s)

Benoit Bruneau

Examples

```
lib.code()
```

---

lsmean   *Least Squares Means*

Description

THIS FUNCTION IS FROM PACKAGE `pda` THAT IS STILL UNDER CONSTRUCTION ON R-Forge. IT HAS BEEN INCLUDED IN `bmisc` FOR PRACTICAL REASONS.

Caution: This routine is not fully tested for models with nested factors or mixed models. Please check results against another package (e.g. SAS proc mixed). It appears to correctly handle `lme` objects, but does not work well for `aov` objects that include `Error()` type nesting in the formula. Further, it does not properly handle polynomial terms–only the linear term is included. For now, create dummies like x2 = x*x manually and include x2 in your model.

Usage

```
lsmean(object, ...)
## Default S3 method:
lsmean(object, ..., factors, effects = FALSE, se.fit = TRUE,
    adjust.covar = TRUE)
## S3 method for class 'lm'
lsmean(object, data, factors, expr, contrast, effects = FALSE,
    se.fit = TRUE, adjust.covar = TRUE, pdiff = FALSE,
    reorder = FALSE, lsd, level = .05, rdf, coef, cov, ...)
## S3 method for class 'lme'
lsmean(object, data, factors, ..., rdf, coef, cov)
## S3 method for class 'lmer'
lsmean(object, data, factors, expr, ..., rdf, coef, cov)
## S3 method for class 'listof'
lsmean(object, data, factors, stratum, expr, contrast, ...)
```

Arguments

`object`	response vector (default) or model object (lm).
`...`	factors and covariates (must be same length as y).
`data`	data frame in which to interpret variables(found from object if missing).
`factors`	character vector containing names of x.factor and trace.factoras first two entries. Must be in `names(data)` and `labels(object)`.Default is all factor names.
`effects`	drop intercept if `TRUE` (only works properly with sum-to-zero contrasts).
`se.fit`	compute pointwise standard errors if T.
`adjust.covar`	adjust means to average covariate values if T; otherwise use covariate mean for each combination of factors.
`pdiff`	Include letters to signify significant differences.
`reorder`	Reorder means from largest to smallest.
`lsd`	Include average LSD if `TRUE` (also need `pdiff=TRUE`).
`level`	Significance level for `pdiff` calculations.
`rdf`	Residual degrees of freedom.

coef	Coefficients for fixed effects in object.
cov	Covariance matrix for fixed effects.
expr	Call expression (formula)
contrast	Type of contrasts (default is attribute `contrasts` of `object`)
stratum	Name of stratum for lsmean calculation as character string.

Value

Data frame containing unique factor levels of factors, predicted response (pred) and standard errors
(se). WARNING: lsmean may not function properly if there are empty cells. Standard errors for
mixed models using methods `lmer` and `listof` are not fully debugged.

Author(s)

Brian S. Yandell

See Also

predict.

Examples

```
## Not run:
lsmean(y,x1,x2)
# the following does the same thing
fit <- lm(y~x1+x2)
data <- data.frame(y,x1,x2)
lsmean(fit,data,factors=c("x1","x2"))

## End(Not run)
```

make.z   *make z*

Usage

```
make.z(x, index = NULL)
```

Arguments

```
x
index
```

Examples

```
# will soon be available
```

---

mc.long   *Pairwise t tests in long format*

Description

Calculate pairwise T tests between group levels with corrections for multiple testing presented in
long format

Usage

```
mc.long(y, ...)
## S3 method for class 'formula'
mc.long( y, data=NULL, subset, ...)
## S3 method for class 'lm'
mc.long( y, ...)
## Default S3 method:
mc.long(y, group, data=NULL, p.adjust.method="holm",
        column=NULL, digits=NULL, subset, silent=FALSE, ...)
```

Arguments

y	response variable for the default method, or lm or formula object. If y is a linear-model object or a formula, the variables on the right-hand-side of the model must all be factors and must be completely crossed.
group	for the default method, factor (concatenated factor when multiple factors). See details.
data	data.frame where the dependant variable and the factor(s) are
p.adjust.method	
	method for adjusting p values. Default is Holm's method. (see P.adjust)
column	new names for the factor(s); this is optional
digits	controls the number of digits for the presented results presented
subset	logical expression indicating elements to keep: missing values are taken as false.
silent	a logical variable indicating whether to indicate the general warning (FALSE) or not (TRUE).
...	additional arguments to pass to p.adjust, pairwise.t.test and/or t.test.

Details

When making multiple t tests for all combinations, the n option of P.adjust can be used to
identify the number of comparisons that are actually used. This is only to simplify the uses p values
corrections on the full output matrix when only some of the comparisons are meaningfull or chosen
for hypothesis testing.

When group is manually defined, use paste(x,y,z) or interaction(x,y,z) form; "x",
"y" and "z" are the factors. There is no restrictions on the number of factors.

Value

Object of class "data.frame" containing the results.

See Also

P.adjust, pairwise.t.test, pair.diff, DTK.test, TukeyHSD and glht

Examples

```
z=data.frame( yy=c(rep("c",50),rep("d",50)),
              x=c(rnorm(50),rnorm(50,10)),
              s=rep(c(rep("a",25),rep("b",25)),2),
              qq=rep(c(rep("w",10),rep("t",10)),5))

mod=lm(x~yy*qq*s, data=z)
formula= x~yy*qq*s

mc.long(y=x, group= paste(yy,qq,s), data=z)
mc.long(y=x, group= paste(yy,qq,s), data=z, p.adjust.method="sidak")
mc.long(y=x, group= paste(yy,qq,s), data=z, p.adjust.method="sidak", n=15)
mc.long(y=x, group= interaction(yy,qq,s), data=z)

mc.long(y=formula, data=z)

mc.long(mod)

res <- mc.long(mod)    #### results are put in "res" object.
```

---

mse   *Mean square error*

Description

Estimates the mean square error (mse)

Usage

```
mse(model)
```

Arguments

model   an object containing the results of a model.

Details

The mean square error is also known as the unexplained variance or the variance of the residuals.

Examples

```
z=data.frame( yy=c(rep("c",50),rep("d",50)),
              x=c(rnorm(50),rnorm(50,10)),
              s=rep(c(rep("a",25),rep("b",25)),2),
              qq=rep(c(rep("w",10),rep("t",10)),5))

mod=lm(x~yy*qq*s, data=z)

mse(mod)
```

---

n *Sample size*

Description

Gives sample size (n) without NA's

Usage

```
n(x)
```

Arguments

x   Vector (numeric or character)

Examples

```
x= rep(c(rnorm(30,20,5),NA),3)

length(x)    ### 93
n(x)         ### 90
```

---

norm.test   *Normality tests*

Description

Lilliefors (Kolmogorov-Smirnov), Shapiro-Francia, Shapiro-Wilk, D'Agostino Skewness, Anscombe-Glynn Kurtosis and D'Agostino-Pearson normality tests.

Usage

```
## Default S3 method:
 plot(norm.test(x, title=NULL, type=c("G1","b1","mc")))
```

Arguments

x	one of:
	- a numeric vector - an object of class `c('lm','aov','glm','gam')` resulting from a calls to `c(lm,aov,glm,gam)`
title	the title at the top of the results. Default is "Normality Tests".
sk	type of skewness used in D'Agostino skewness test. Can be "G1","b1" or "mc". Read details.
type	type of residuals which should be used. See details.

Details

This function can be used on objects belonging to `c('lm', 'aov', 'glm', 'gam')` classes. For example, `class(aov.model)` gives `"aov" "lm"` and `class(glm.model)` gives `"glm" "lm"`. The `type` of residuals can be defined. It generally includes `c("working", "response", "deviance", "pearson", "partial")`.

D'Agostino-Pearson's test is more appropriate for analysing a vector with duplicate values in it. The more there are duplicate values, the more Shapiro-Wilk will be far from correctly testing the $H0$ hypothesis.

Given samples from a population, the equation for the sample skewness g_1 is a biased estimator of the population skewness. The use of G_1 or b_1 is advisable. For large samples, the various skewness estimates yield similar results. For small normal distributed samples, b_1 is less biased than G_1. However, for small non-normal distributed samples, G_1 is less biased than b_1. These two skewness estimate can be sensitive to outliers in the data (contaminated data). Therefore, the medcouple `mc` is also an option in `type`. It has a good performance on uncontaminated data and is robust on contaminated data. For more information on medcouple, please read references in `mc{`**robustbase**`}`.

- Typical definition used in many older textbooks:

$$g_1 = \frac{m_3}{m_2^{3/2}}$$

  where m_3 is the sample third central moment, and m_2 is the sample variance.

- Definition used in SAS and SPSS:

$$G_1 = g_1 \frac{k_3}{k_2^{3/2}} = g_1 \frac{\sqrt{n(n-1)}}{n-2}$$

where k_3 is the unique symmetric unbiased estimator of the third cumulant and k_2 is the symmetric unbiased estimator of the second cumulant.

- Definition used in MINITAB and BMDP:

$$b_1 = \frac{m_3}{s^3} = g_1 \left(\frac{n-1}{n} \right)^{3/2}$$

More will be added to this section especially for Anscombe-Glynn Kurtosis test.

Value

An S4 object of class 'norm' containig the following components:

`statistics`	the statistics for each analysis
`p.value`	estimated p-values based on the statistics
`data`	original data (`data.frame`)
`data.name`	names of the object called
`title`	title for the result

References

D. N. Joanes and C. A. Gill (1998), Comparing measures of sample skewness and kurtosis. *The Statistician*, **47**, 183–189.

G. Brys, M. Hubert and A. Struyf (2003), A Comparison of Some NewMeasures of Skewness. in *Developments in Robust Statistics* **ICORS 2001**, eds. R. Dutter, P. Filzmoser, U. Gather, and P.J. Rousseeuw, Heidelberg: Springer-Verlag, 98–113

G. Brys, M. Hubert and A. Struyf (2004), A Robust Measure of Skewness; *JCGS* **13** (4), 996–1017.

Examples

```
x <- rnorm(300, 50, 10)
y   <- 5*(x +10*(rnorm(300,1,2)))

norm.test(x)              ## mc skewness
norm.test(x, type="G1") ## G1 skewness
norm.test(x, type="b1") ## b1 skewness

mod <- lm(y~x)
norm.test(mod)
```

P.adjust   *Adjust P-values for Multiple Comparisons*

Description

Given a set of p-values, returns p-values adjusted using one of several methods. This is a modified version of `p.ajust` from `stats`. It now includes `"sidak"` correction.

Usage

```
P.adjust(p, method = P.adjust.methods, n = length(p))

P.adjust.methods
 c("holm", "hochberg", "hommel", "sidak", "bonferroni", "BH",
   "BY", "fdr", "none")
```

Arguments

p   vector of p-values (possibly with NAs).

method   correction method

n   number of pvalues considered for correction; only set this (to non-default) when
                  you know what you are doing! See details

Details

The adjustment methods include the Bonferroni correction (`"bonferroni"`) in which the p-values are multiplied by the number of comparisons. Less conservative corrections are also included by Holm (1979) (`"holm"`), Hochberg (1988) (`"hochberg"`), Hommel (1988) (`"hommel"`), Benjamini & Hochberg (1995) (`"BH"`), and Benjamini & Yekutieli (2001) (`"BY"`), respectively. A pass-through option (`"none"`) is also included. The `P.adjust.methods` vector contains the set of correction methods for the benefit of methods that need to have the method as an option and pass it on to `P.adjust`.

The first five methods are designed to give strong control of the family wise error rate. There seems no reason to use the unmodified Bonferroni correction because it is dominated by Holm's method, which is also valid under arbitrary assumptions.

Hochberg's and Hommel's methods are valid when the hypothesis tests are independent or when they are non-negatively associated (Sarkar, 1998; Sarkar and Chang, 1997). Hommel's method is more powerful than Hochberg's, but the difference is usually small and the Hochberg p-values are faster to compute.

The `"BH"` and `"BY"` method of Benjamini, Hochberg, and Yekutieli control the false discovery rate, the expected proportion of false discoveries amongst the rejected hypotheses. The false discovery rate is a less stringent condition than the family wise error rate, so these methods are more powerful than the others.

When making multiple comparisons, n can be used to identify the number of comparisons that are actually used. Correction is then done on the full output matrix when only some of the comparisons are meaningfull or chosen for hypothesis testing. This can be done with the `"bonferroni"` and `"sidak"` correction. If other methods are used, exclude the unwanted p.values before applying correction. Unless you know what you are doing, **DO NOT** modify n if all comparisons are used. Most of the time n should be equal to length(p).

Note that you can set n larger than `length(p)` which means the unobserved p-values are assumed to be greater than all the observed p for `"bonferroni"` and `"holm"` methods and equal to 1 for the other methods.

Value

A vector of corrected p-values (same length as `p`).

References

Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series* B, **57**, 289–300.

Benjamini, Y., and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics* **29**, 1165–1188.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, **6**, 65–70.

Hommel, G. (1988). A stagewise rejective multiple test procedure based on a modified Bonferroni test. *Biometrika*, **75**, 383–386.

Hochberg, Y. (1988). A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, **75**, 800–803.

Shaffer, J. P. (1995). Multiple hypothesis testing. *Annual Review of Psychology*, **46**, 561–576. (An excellent review of the area.)

Sarkar, S. (1998). Some probability inequalities for ordered MTP2 random variables: a proof of Simes conjecture. *Annals of Statistics*, **26**, 494–504.

Sarkar, S., and Chang, C. K. (1997). Simes' method for multiple hypothesis testing with positively dependent test statistics. *Journal of the American Statistical Association*, **92**, 1601–1608.

Wright, S. P. (1992). Adjusted P-values for simultaneous inference. *Biometrics*, **48**, 1005–1013. (Explains the adjusted P-value approach.)

See Also

`pairwise.t.test`, `mc.long`, `DTK.test`, `TukeyHSD` and `glht`

Examples

```
require(graphics)

set.seed(123)
x <- rnorm(50, mean=c(rep(0,25),rep(3,25)))
p <- 2*pnorm( sort(-abs(x)))

round(p, 3)
round(P.adjust(p), 3)
round(P.adjust(p,"BH"), 3)

## or all of them at once (dropping the "fdr" alias):
P.adjust.M <- P.adjust.methods[P.adjust.methods != "fdr"]
p.adj <- sapply(P.adjust.M, function(meth) P.adjust(p, meth))
round(p.adj, 3)
## or a bit nicer:
noquote(apply(p.adj, 2, format.pval, digits = 3))
```

```
## and a graphic:
matplot(p, p.adj, ylab="P.adjust(p, meth)", type = "l", asp=1, lty=1:6,
        main = "P-value adjustments")
legend(.7,.6, P.adjust.M, col=1:6, lty=1:6)

## Can work with NA's:
pN <- p; iN <- c(46,47); pN[iN] <- NA
pN.a <- sapply(P.adjust.M, function(meth) P.adjust(pN, meth))
## The smallest 20 P-values all affected by the NA's :
round((pN.a / p.adj)[1:20, ] , 4)
```

`pack.list`   *List of installed packages*

Description

Creates a character vector of currently installed packages and prints it to either a text file or directly into Rprofile file.

Usage

```
pack.list(rprofile=FALSE)
```

Arguments

rprofile   logical. Should vector of currently installed packages be added to the Rprofile
                (`TRUE`) or saved in a text file (`FALSE`)

Details

rprofile = `FALSE`
A `SAVE` dialog box opens.

rprofile = `TRUE`
For Mac and Linux users, `.Rprofile` is either modified or created. On Windows systems, users are prompted for `Rprofile.site` location. This file is usually in `"C:/Program Files/R/R-x.xx.x/etc"`. Once it's selected, the file is modified to include the list of packages currently installed in R.

Author(s)

Benoit Bruneau

See Also

[Startup](#) explains what happens when R starts.

Examples

```
pack.list()
pack.list(rprofile=T)
```

---

pair.diff   *Mean differences matrix and their associated standard Errors*

Description

Creates two lower triangle matrix: The mean differences and their standard error.

Usage

```
pair.diff(y, ...)
## S3 method for class 'formula'
pair.diff(y, data=NULL ...)
## S3 method for class 'lm'
pair.diff( y, ...)
## Default S3 method:
pair.diff( y, group, data=NULL, ...)
```

Arguments

y   response variable for the default method, or lm or formula object. If y is
                a linear-model object or a formula, the variables on the right-hand-side of the
                model must all be factors and must be completely crossed.

group   for the default method, factor (concatenated factor when multiple factors). See
                details.

data   data.frame where the dependant variable and the factor(s) are.

...   additional arguments to pass to mean and/or sd.

Details

When group is manually defined, use paste(x,y,z) or interaction(x,y,z) form where "x",
"y" and "z" are the factors. There is no restrictions on the number of factors.

This function can be usefull with pairwise.t.test since the matrix created are of the same
format.

Value

Object of class "list" containing two matrices:

diff.m   Mean differences half matrix

diff.se   Standard error associated with the mean differences half matrix

See Also

Is included in mc.long for the long format of the results.

Examples

```
z=data.frame( yy=c(rep("c",50),rep("d",50)),
              x=c(rnorm(50),rnorm(50,10)),
              s=rep(c(rep("a",25),rep("b",25)),2),
              qq=rep(c(rep("w",10),rep("t",10)),5))

mod=lm(x~yy*qq*s, data=z)
y= x~yy*qq*s

pair.diff(y=x, group= paste(yy,qq,s), data=z)
pair.diff(y=x, group= interaction(yy,qq,s), data=z)
pair.diff(y=y, data=z)
pair.diff(mod)
```

performance   *performance*

Usage

```
performance(expr, samples = 1, gcFirst = TRUE)
```

Arguments

expr

samples

gcFirst

Examples

```
# will soon be available
```

plot.lev	*Plot Diagnostics for an* lev *Object*

Description

Four boxplots (selectable by `which`) are available.

Usage

```
plot(obj, which=1L:4L, ...)
```

Arguments

obj   an object of class 'lev' resulting from a calls to lev.

which   if a subset of the plots is required, specify a subset of the numbers 1:4:

- `1` : residuals of mean (Levene)
- `2` : residuals of trimmed mean (Robust Levene)
- `3` : residuals of median (Brown-Forsythe)
- `3` : r scores (O'Brien)

...   arguments to be passed to boxplot.

Examples

```
z=data.frame( yy=c(rep("c",50),rep("d",50)),
        x=c(rnorm(50),rnorm(50,10)),
        s=rep(c(rep("a",25),rep("b",25)),2),
        qq=rep(c(rep("w",10),rep("t",10)),5))

mod=lm(x~yy*qq*s, data=z)
res=lev(mod)

plot(res)
```

plot.logit	*Standard plot for maturity ogive*

Usage

```
plot.logit(object, se.pred=TRUE, leg=TRUE, ref=TRUE, range.x=c("data","full"),
          warn.val=TRUE, ylab="ProbabilitC)",xlab="Longueur C  la fourche (mm)",
          main=NULL, enc.utf8=FALSE)
```

Arguments

object	an object of class 'glm' resulting from a call to glm.
se.pred	logical; if TRUE, SE is plotted.
leg	logical; if TRUE, a legend containing logistic equation and estimated values for the variables is plotted.
ref	logical; if TRUE, reference lines for L90, L50 and L10 are plotted.
range.x	the range used to define xlim in the plot. Read 'details'.
warn.val	logical; if TRUE plots $x01$ and $x99$ when the fit is suspicious.
main	an overall title for the plot. If NULL ...
ylab	a title for the y axis.
xlab	a title for the x axis.
enc.utf8	logical; if TRUE, iconv(x,"utf-8") is used on text in the graph. Read 'details'.

Details

When using RConsole inside of Eclipse, encoding is wrong. The use of enc.utf8=TRUE is a temporary fix for correctly plotting characters with accents.

Author(s)

Benoit Bruneau

QQplot   *QQplot*

Usage

```
QQplot(dat, quant=TRUE,cex.q=2,norm=T, ...)
```

Arguments

dat
: one of:
  - a numeric vector
  - an object of class `c('norm','lm','aov','glm','gam')` resulting from a calls to `c(norm.test,lm,aov,glm,gam)`

quant
: logical; T for adding quantiles 75, 50 (median) and 25.

cex.q
: numeric vector giving the amount by which plotting symbols should be magnified relative to the default

norm
: logical; T adds a line to a normal quantile-quantile plot.

...
: additional arguments to be passed (see `par`, `qqnorm`)

Examples

```
x=rnorm(50)
QQplot(x)

norm.x=norm.test(x)
QQplot(norm.x)
```

---

r.colors   *Pie charts of all R character colors*

Description

Creates a pdf file with pie charts of all the 657 basic character colors of R

Usage

```
r.colors(file)
```

Arguments

file   the directory in which the pdf file will be created

Details

Define the directory in which the file should saved by writing `file="C:/temp"` for example. If file is not defined, it will be saved in `"C:/"` on windows and in `"home"` on Mac.

Value

None

Examples

```
r.colors()
```

reject.z *reject z*

Usage

```
reject.z(x, index = NULL, threshold = 2)
```

Arguments

```
x
index
threshold
```

Examples

```
# will soon be available
```

---

replace.z   *replace z*

Usage

```
replace.z(x, index = NULL, threshold = 2)
```

Arguments

```
x
index
threshold
```

Examples

```
# will soon be available
```

resid.ortho *Orthogonal residuals*

Usage

```
xxx( data ,  ,  ,  )
```

Arguments

```
data
```

Author(s)

Benoit Bruneau

---

rivard   *Rivard Weights Calculation*

Description

This function applies Rivard equations to mid-year weight at age data to adjust values to Jan-1 basis.

Usage

```
rivard(pds, pred=FALSE, K=2, plus.gr=FALSE)
```

Arguments

```
data
```

Details

More to come. Will be adding interpolation for spawning season.

Examples

```
x=rnorm(30,800,10)
rivard(data.frame("2000"=x,"2001"=x*1.2, "2002"=x*0.8,"2003"=x*0.5))
```

rollmin	*rollmin*

Usage

```
rollmin(x, k, na.pad = FALSE, align = c("center", "left", "right"),
        ...)
```

Arguments

x

k

na.pad

align

...

Examples

```
# will soon be available
```

---

roundup *roundup*

Description

The "conventional" rounding of 5 to the higher value

Usage

```
roundup(x, numdigits = 0)
```

Arguments

x   numeric vector.

digits   integer indicating the number of decimal places to be used.

Details

Rounds a 5 to the next value. Therefore roundup(2.5) is 3. This can be usefull when the rounded values are to be presented in a document (eg. table, graph,...).

When rounded values are used in other calculations, round should be used since it follows the IEC 60559 standard.

Value

numeric vector.

See Also

round

Examples

```
round(2.5)
roundup(2.5)
```

runmax	*runmax*

Usage

```
runmax(x, window)
```

Arguments

x

window

Examples

```
# will soon be available
```

runmean   *runmean*

Usage

```
runmean(x, window)
```

Arguments

```
x
window
```

Examples

```
# will soon be available
```

---

runmin   *runmin*

Usage

```
runmin(x, window)
```

Arguments

x

window

Examples

```
# will soon be available
```

`s.an`	*Simulations for YPR model*

Description

Not ready yet. Use `for` loops for now.

Usage

```
xxx( data , , , )
```

Arguments

```
data
```

Author(s)

Benoit Bruneau

---

se   *Standard Error*

Usage

```
se(x, na.rm=T)
```

Arguments

x   an R object (vector, matrix,...)

na.rm   a logical value indicating whether NA values should be stripped before the com-
                  putation proceeds

Details

The standard error of the mean is defined as:

$$SE = \frac{sd}{\sqrt{n}}$$

where sd is the standard deviation of the sample and n is the sample size.

Examples

```
x=rnorm(50)
se(x)
```

---

show.North   *North arrow for a map*

Description

Draws North arrow on a map

Usage

```
show.North(pos, arrow.col="black", arrow.fill="black", arrow.lwd=1,
           N.cex=1, N.family="HersheyGothicEnglish")
```

Arguments

pos   Position of the arrow. Default is 'topright'. See details.

arrow.col   Arrow color.

arrow.fill   Color inside the head of the arrow. NA for no color.

arrow.lwd   Line width of the arrow.

N.cex   Character size for 'N'.

N.family   Font family of 'N'.

Details

The position of the north arrow is defined by pos and can either be numeric or character.

If pos is a numeric vector, it is a vector of the form c(x,y) where x and y are fractions of the plotting region. If x and y are not in $[0, 1]$, then the north arrow is drawn outside the bounds of the plotting region and a warning message is given.

If pos is a character vector, it should be a single keyword from:

- c('topright','topleft','bottomright','bottomleft')

Examples

```
plot(1)
show.North()
show.North(c(0.8,0.9))
show.North(c(1.01,0.9))  ### gives a warning
```

sort.vdf	*Sort Data Frames and Vectors*

Description

Single function enabling `data.frame` and `vector` sorting

Usage

```
sort.vdf(x, by, increasing=TRUE)
```

Arguments

x	`data.frame` or `vector`
by	A one-sided formula using + for ascending and - for descending. Sorting is left to right in the formula. This is for data.frame only.
increasing	logical. Should the sort be increasing (`TRUE`) or decreasing (`FALSE`)? This is for sorting vectors only.

Details

See example.

Author(s)

Kevin Wright and modified by Benoit Bruneau

Examples

```
x=rnorm(10)
y=runif(30)
z=data.frame(x,y)

sort.vdf(x)              ### Sort a vector in increasing order
sort.vdf(z)              ### Gives an error message
sort.vdf(z,by= ~ +x)     ### Sort (z) by a column (+x)
sort.vdf(z,by= ~ +x +y)  ### Sort (z) by two column (+x and then +y)
sort.vdf(z,by= ~ +x -y)  ### Sort (z) by two column (+x and then -y)
```

---

ttest.perm   *Student's t-tests by Permutation*

Description

Performs two sample t-tests or paired t-test by use of permutation

Usage

```
ttest.perm(vec1, vec2, nperm=999, alternative = "two.sided",
           var.equal = T, silent=FALSE, type="i", exact=FALSE)
```

Arguments

vec1, vec2   two numeric vectors used for Student's t-test analysis

nperm   number of permutations (default = 999)

alternative   one of the following: "two.sided", "less" or "greater".

var.equal   a logical variable indicating whether to treat the two variances as being equal
                (TRUE) or not (FALSE).

silent   a logical variable indicating whether calculation results are printed (FALSE) to
                the R console or not (TRUE).

type   one of the following: "i" for independant samples or "p" for paired samples.

exact   a logical variable indicating whether to perform the exact test (TRUE) or not
                (FALSE).

Details

The permutational t-test does not require normality of the distributions of each variable. It is also
quite robust to heteroscedasticity.

Use exact=TRUE to perform two sample t-test on all the possible combination. This option can
only be used when the sum of the sample sizes ($n_1 + n_2$) is smaller than 20. It is recommended to
use this option when sample sizes are small. It is not implemented yet in the paired t-test.

nperm can not be higher than the maximum number of combination possible (n_{comb}).

$n_{comb} = N!/(n_1!n_2!)$
       where n_comb is the number of possible combinations, $N!$ is factorial$(n_1 + n_2)$, $n_1!$ is
       factorial(n(vec1)) and $n_2!$ is factorial(n(vec2)).

There is more to come in this section.

Value

t.ref   reference value of the t-statistic

p.param   parametric p-value

p.perm   permutational p-value

nperm   number of permutations

perm.t   list of the t statistics (only for independant sample ttest), starting with the refer-
                ence value, followed by all values obtained under permutations.

Examples

```
x <- rnorm(50,0,1)
y <- runif(50,0,1)*x
toto = ttest.perm(x, y)  ##independant samples ttest
```

---

unload *Unload packages*

Description

Unloads one or multiple packages.

Usage

```
unload(pack)
```

Arguments

pack   Character vector specifying which packages to unload.

Author(s)

Benoit Bruneau

Examples

```
library(mgcv)
search()
unload(mgcv)
search()
```

week.1 *week.1*

Description

Week of the year starting on the first of January (01-53)

Usage

```
week.1(x)
```

Arguments

x

Author(s)

Denis Chabot

Examples

```
# will soon be available
```

---

week.num   *week.num*

Description

Week of the year as decimal number (00-53) using Sunday or Monday as the first day 1 of the week (and typically with the first Sunday of the year as day 1 of week 1).

Usage

```
week.num(x, day=c("sunday", "monday"))
```

Arguments

x   A vector of dates.

day   Either "sunday" or "monday". Default is "sunday".

Details

Argument `day` indicates if the week starts on `"sunday"` or `"monday"`.

Examples

```
dated <-as.Date(c("2006-05-18","2006-05-07","2006-04-23",
                  "2006-04-24","2006-05-07","2007-05-17",
                  "2007-05-06","2007-04-22","2007-04-29"))

week.num(dated, "monday")
week.num(dated)
```

Index