Package ‘Deducer’

May 6, 2026

Version 0.9-2

Date 2026-05-05

Title A Data Analysis GUI for R

Maintainer Ian Fellows <`ian@fellstat.com`>

Description An intuitive, cross-platform graphical data analysis system. It uses menus and dialogs to guide the user efficiently through the data manipulation and analysis process, and has an excel like spreadsheet for easy data frame visualization and editing. Deducer works best when used with the Java based R GUI JGR, but the dialogs can be called from the command line. Dialogs have also been integrated into the Windows Rgui.

Depends R (>= 2.15.0), ggplot2 (>= 2.0.0), JGR(>= 1.7-10), car, MASS

Imports rJava, e1071, scales, plyr, foreign, multcomp, effects

Suggests Hmisc, brunnermunzel, cli

SystemRequirements Java, JRI

License GPL-2

URL `https://www.deducer.org/` `https://www.fellstat.com`

NeedsCompilation no

Author Ian Fellows [aut, cre]

Repository CRAN

Date/Publication 2026-05-06 21:10:02 UTC

Contents

`add.cross.strata.test` *Apply a Stratified test to a Contingency Table*

Description

Applies and adds a hypothesis test to a `contingency.tables` object.

Usage

```
add.cross.strata.test(tables,name,htests,types=c("asymptotic","monte.carlo","exact"))
```

Arguments

`tables`	An object of class `contingency.tables`
`name`	The name of the hypothesis test
`htests`	A function or list of functions which take a three dimensional array as it's argument and returns an object of class `htest`
`types`	A character vector with the same number of items as htests, indicating what type of test was done

Value

A `contingency.tables` object identical to `tables`, but with the test applied to each table.

See Also

`add.mantel.haenszel` `add.test`

Examples

```
dat<-data.frame(a=rnorm(100)>.5,b=rnorm(100)>0,c=rnorm(100)>(-.5))
tables<-contingency.tables(
 row.vars=a,
 col.vars=b,
 stratum.var=c,data=dat)
add.cross.strata.test(tables,"Mantel-Haenszel",list(function(x) mantelhaen.test(x,correct=FALSE)),
 "asymptotic")
tables
```

`add.mantel.haenszel` *Apply the Mantel-Haenszel test to a Contingency Table*

Description

Applies and adds the Cochran-Mantel-Haenzsel test to a `contingency.tables` object. The Cochran-Mantel-Haenzsel tests the independence of two nominal variables, stratified by a third nominal variable, assuming no three way interaction.

Usage

```
add.mantel.haenszel(tables,conservative=FALSE)
```

Arguments

`tables`	An object of class `contingency.tables`
`conservative`	Should a continuity 'correction' be applied

Details

This is a convenience function wrapping `mantelhaen.test` in a `add.cross.strata.test` call. See `mantelhaen.test` for further details.

Value

A `contingency.tables` object identical to `tables`, but with the test applied to each table.

See Also

`add.cross.strata.test` `add.test` `mantelhaen.test`

Examples

```
dat<-data.frame(a=rnorm(100)>.5,b=rnorm(100)>0,c=rnorm(100)>(-.5))
tables1<-contingency.tables(
 row.vars=a,
 col.vars=b,
 stratum.var=c,data=dat)
tables1<-add.mantel.haenszel(tables1)
print(tables1,prop.r=TRUE,prop.c=TRUE,prop.t=FALSE)
```

`add.test` *Apply a test to a Contingency Tables object*

Description

Applies and adds a test to a `contingency.tables` object.

Usage

```
add.test(tables,name,htests,types=c("asymptotic","monte.carlo","exact"))

add.chi.squared(tables, simulate.p.value = FALSE, B = 10000)

add.likelihood.ratio(tables, conservative = FALSE, simulate.p.value = FALSE, B = 10000)

add.fishers.exact(tables, simulate.p.value = FALSE, B = 10000)

add.correlation(tables,method=c("spearman","kendall"))

add.kruskal(tables,nominal=c("both","rows","cols"))
```

Arguments

`tables`	An object of class `contingency.tables`
`name`	Name of the test
`htests`	A function or list of functions which take a matrix as it's argument and returns an object of class `htest`
`types`	A character vector with the same number of items as `htests`, indicating what type of test was done
`conservative`	Should a conservative p-value be computed. i.e. One with a continuity correction for asymptotic tests and not using the mid p-value for exact and approximate tests
`simulate.p.value`	If `TRUE` p-values will be computed via monte carlo simulation
`B`	the number of samples for the monte carlo simulation
`method`	the type of correlation
`nominal`	Should the rows or columns be considered nominal.

Details

add.test applies a supplied list of tests to all of the tables in `tables`.

add.chi.squared is a wrapper function applying the `chisq.test` function to each table. add.likelihood.ratio is a wrapper function applying the `likelihood.test` function to each table. add.fishers.exact is a wrapper function applying the `fisher.test` function to each table. add.correlation is a wrapper function applying the `cor.test` function to each table. add.kruskal is a wrapper function applying the `kruskal.test` function to each table.

Value

A `contingency.tables` object identical to `tables`, but with the test applied to each table.

See Also

`add.cross.strata.test` `likelihood.test` `cor.test` `kruskal.test`

Examples

```
dat<-data.frame(a=rnorm(100)>.5,b=rnorm(100)>0,c=rnorm(100)>(-.5))
tables<-contingency.tables(
row.vars=a,
col.vars=b,
stratum.var=c,data=dat)
tables<-add.chi.squared(tables,simulate.p.value=TRUE,B=10000)
tables<-add.likelihood.ratio(tables)
tables<-add.fishers.exact(tables)
tables<-add.correlation(tables,method='kendall')
tables<-add.kruskal(tables)
tables<-add.mantel.haenszel(tables)
print(tables)
remove(tables)
```

`as.matrix.cor.matrix` *as.matrix method*

Description

as matrix

Usage

```
## S3 method for class 'cor.matrix'
as.matrix(x,...)
```

Arguments

`x`	Object of class "`cor.matrix`".
`...`	further arguments. unsued

Value

a matrix

`chi.noncentral.conf` *Non-central Chi-Squared Confidence Interval*

Description

Confidence interval for the Non-centrality parameter of Non-central chi-squared distribution

Usage

```
chi.noncentral.conf(chival,df,conf,prec=.00001)
```

Arguments

`chival`	The observed Chi-Squared value
`conf`	The confidence level (e.g. .95)
`df`	Degrees of freedom
`prec`	Precision of estimate

Value

A 2X2 matrix whose rows represent the upper and lower bounds, and whose columns represent the parameter value and upper tail percentiles.

References

Smithson, M.J. (2003). Confidence Intervals, Quantitative Applications in the Social Sciences Series, No. 140. Thousand Oaks, CA: Sage.

See Also

`Chisquare`

Examples

```
chi.noncentral.conf(6,1,.95)
#           Result:

#      Non-Central          %
#Lower   0.2089385 0.97500899
#Upper  19.4443359 0.02499302
```

`contin.tests.to.table` *contin.tests.to.table*

Description

Makes a nice table out of a `contin.tests` object

Usage

```
contin.tests.to.table(tests,test.digits=3,...)
```

Arguments

`tests`	a `contin.tests` object
`test.digits`	The number of digits to round to
`...`	other paramaters

Value

A nice table

`contingency.tables` *Contingency Tables*

Description

Creates a contingency.tables object

Usage

```
contingency.tables(row.vars, col.vars, stratum.var, data=NULL, missing.include=FALSE )
```

Arguments

`row.vars`	A variable or data frame evaluated in data
`col.vars`	A variable or data frame evaluated in data
`stratum.var`	A variable evaluated in data
`data`	A `data.frame`
`missing.include`	A logical indicating whether a missing category should be included in the table

Value

A list with class "contingency.tables." Each element of the list is a single contingency table of class "contin.table" corresponding to each combination of elements of `row.vars` and `col.vars` stratified by `stratum.var`

See Also

`extract.counts`

Examples

```
temp.data<-data.frame(a=rnorm(100)>0,b=rnorm(100)>0,gender=rep(c("male","female"),50))
#a vs. b stratified by gender
tab<-contingency.tables(a,b,gender,data=temp.data)
tab

##add in chi-squared tests
tab<-add.chi.squared(tab)
tab
```

`cor.matrix` *cor.matrix*

Description

Creates a correlation matrix

Usage

```
cor.matrix(variables,with.variables,data=NULL,test=cor.test,...)
```

Arguments

`variables`	variables
`with.variables`	An optional set of variables to correlate with `variables`. If nothing is specified, all variables in `variables` are correlated with themselves.
`data`	A data.frame from which the variables and factor will be selected.
`test`	A function whose first two arguments are the variables upon which the correlation will be calculated, and whose result is an object of class `htest`.
`...`	further arguments for `test`.

Value

A `multi.test` object, representing a table of the results of `func` applied to each of the variables.

See Also

`cor.test` `as.matrix.cor.matrix`

Examples

```
dat<-data.frame(aa=rnorm(100),bb=rnorm(100),cc=rnorm(100),dd=rnorm(100))
dat$aa<-dat$aa+dat$dd
dat$cc<-dat$cc+dat$aa
cor.matrix(dat,test=cor.test)
cor.matrix(d(aa,cc),data=dat,test=cor.test,method="kendall")
cor.matrix(d(aa,cc),d(dd,bb),data=dat,test=cor.test,method="spearman")
```

`d` *wrapper for data.frame*

Description

This function creates data frames, tightly coupled collections of variables which share many of the properties of matrices and of lists, used as the fundamental data structure by most of R's modeling software. It is a keystroke saving wrapper for the data.frame function. The only difference is that check.names and stringsAsFactors are FALSE by default.

Usage

```
d(..., row.names = NULL, check.rows = FALSE,
                check.names = FALSE,
                stringsAsFactors = FALSE)
```

Arguments

`...`	items
`row.names`	NULL or a single integer or character string specifying a column to be used as row names, or a character or integer vector giving the row names for the data frame.
`check.rows`	if TRUE then the rows are checked for consistency of length and names.
`check.names`	logical. If TRUE then the names of the variables in the data frame are checked to ensure that they are syntactically valid variable names and are not duplicated. If necessary they are adjusted (by make.names) so that they are.
`stringsAsFactors`	logical: should character vectors be converted to factors?

See Also

`data.frame`

Examples

```
x <- d(rnorm(10),1:10)
```

`deducer` *GUI Access functions*

Description

splits a variable into two groups

Usage

```
deducer(cmd=NULL)

data.viewer()
```

Arguments

`cmd`	The command to be executed

`deducer.addMenu` *Controls Deducer's command line menus*

Description

Controls Deducer's command line menus

Usage

```
deducer.addMenu(name, pos=length(menus)+1)
deducer.setMenus(newMenus)
deducer.getMenus()
deducer.addMenuItem(name, pos=NULL, command, menuName, silent=TRUE)
menuFunctions()
```

Arguments

`name`	name of item or menu to add
`pos`	position at which to add the item or menu
`menuName`	the name of the menu to add the item to
`command`	A character vector representing the R command to be run
`silent`	Should the command be executed silently
`newMenus`	new menus

Examples

```
#add a menu with two items
deducer.addMenu("TestMenu")
deducer.addMenuItem("test1",,"cat('test1 selected')","TestMenu")
deducer.addMenuItem("test2",,"print(summary(lm(rnorm(100)~rnorm(100))))","TestMenu")

#Add menu to gui if applicable
if(.windowsGUI){
#winMenuAdd("TestMenu")
#winMenuAddItem("TestMenu", "test1", "cat('test1 selected')")
#winMenuAddItem("TestMenu", "test2", "print(summary(lm(rnorm(100)~rnorm(100))))")
}else if(.jgr){
jgr.addMenu("TestMenu")
jgr.addMenuItem("TestMenu", "test1", "cat('test1 selected')")
jgr.addMenuItem("TestMenu", "test2", "print(summary(lm(rnorm(100)~rnorm(100))))")
}
```

`descriptive.table` *Table of Descriptives*

Description

Table of descriptive statistics, possibly stratified

Usage

```
descriptive.table(vars,
strata,
data,
func.names = c("Mean","St. Deviation","Median",
"25th Percentile","75th Percentile",
"Minimum","Maximum","Skew","Kurtosis","Valid N"),
func.additional)
```

Arguments

`vars`	A variable or data.frame containing variables on which to run descriptive statistics.
`data`	The data frame in which vars is evaluated
`strata`	A variable or data.frame containing variables on which to stratify
`func.names`	A character vector of built-in statistics
`func.additional`	A named list of functions. Each function should take a numeric vector as its argument, and return a single value

Value

Returns a list of `matrix` objects containing descriptive information on all variables in `dat`. One for each level or combination of levels in `strata`.

See Also

`frequencies` `mean` `by`

Examples

```
data(mtcars)
##means and standard deviations
descriptive.table(vars = d(mpg,hp),data= mtcars,
func.names =c("Mean","St. Deviation","Valid N"))
##stratifying by cyl
descriptive.table(vars = d(mpg,hp) ,
strata = d(cyl),data= mtcars,
func.names =c("Mean","St. Deviation","Valid N"))


func.list=list(mean.deviance=function(x) mean(abs(x-mean(x))))

##Adding deviance as a statistic
descriptive.table(vars = d(mpg,hp) ,
strata = d(cyl),data= mtcars,
func.names =c("Mean","St. Deviation","Valid N"),func.additional=func.list)
```

`devel` *Deducer's plug-in development tools*

Description

functions pertaining to GUI development

Usage

```
addComponent(container, component, top, right, bottom,
left, topType = "REL", rightType = "REL", bottomType = "REL",
leftType = "REL")
getSize(component)
setSize(component,width,height)
execute(cmd)

ButtonGroupWidget
CheckBoxesWidget
DeducerMain
JLabel
RDialog
```

```
SimpleRDialog
SimpleRSubDialog
SingleVariableWidget
SliderWidget
TextAreaWidget
VariableListWidget
VariableSelectorWidget
ComboBoxWidget
RDialogMonitor
ListWidget
AddRemoveButtons
TextFieldWidget
ObjectChooserWidget
```

Arguments

`container`	A Java Swing container with Anchor layout
`component`	a Java Swing component
`top`	location of top of component 0 - 1000
`right`	location of right of component 0 - 1000
`bottom`	location of bottom of component 0 - 1000
`left`	location of left of component 0 - 1000
`topType`	Type of constraint on top of component. Can be "REL", "ABS", or "NONE"
`rightType`	Type of constraint on right of component. Can be "REL", "ABS", or "NONE"
`bottomType`	Type of constraint on bottom of component. Can be "REL", "ABS", or "NONE"
`leftType`	Type of constraint on left of component. Can be "REL", "ABS", or "NONE"
`height`	new height of component or window in pixels
`width`	new width of component or window in pixels
`cmd`	the command to be executed

Details

addComponent adds a Java object of class Component to a container (usually an RDialog or SimpleRDialog). the location of the component is determined by the top, right, bottom, and left arguments, which are numbers between 1 and 1000 indicating the distance from either the top (or left) of the container, with 1000 indicating the opposite side of the container. Each side can be constrained in three different ways. If the Type is "REL", the side will scale proportional to the container when the container is resized. If it is "ABS", it is not rescaled. If it is "NONE", the location of that side is determined by the componet's preferred size, which can be set with the "setPreferedSize" method.

getSize gets the height and width

setSize sets the height and width

execute executes a character representing a command, as if it were entered into the console

The rest of the items are references to the Java classes of commonly used GUI components. see www.deducer.org for more details and usage.

`dich` *dich*

Description

splits a variable into two groups

Usage

```
dich(variables,data=NULL,cut=NULL,group1=NULL,group2=NULL)
```

Arguments

`variables`	variables to be dichotomized
`data`	A data.frame
`cut`	An optional cut point dividing `factor` into two groups.
`group1`	An optional vector of levels of `factor` defining group 1.
`group2`	An optional vector of levels of `factor` defining group 2.

Value

a data.frame containing the variables, recoded into two groups.

`extract.counts` *Extract Contingency Table Arrays*

Description

Extracts the counts of a contingency.tables object

Usage

```
extract.counts(tables)
```

Arguments

`tables`	A `contingency.table` object

Value

A named list of three dimensional arrays. One for each `contin.table` in `tables`

See Also

`contingency.tables`

Examples

```
temp.data<-data.frame(a=rnorm(100)>0,b=rnorm(100)>0,gender=rep(c("male","female"),50))
#a vs. b stratified by gender
tab<-contingency.tables(a,b,gender,data=temp.data)
tab

##extract counts
extract.counts(tab)

##Yields something like the following:
#$`a by b`
#, , female
#
#      FALSE TRUE
#FALSE    11    9
#TRUE     15   15
#
#, , male
#
#      FALSE TRUE
#FALSE    10   10
#TRUE     22    8
```

`frequencies` *Frequency Tables*

Description

Creates a set of frequency tables.

Usage

```
frequencies(data,r.digits=1)
```

Arguments

`data`	A data.frame containing the variables on which to run frequencies
`r.digits`	how many digits should the percentages be rounded to

Value

Returns a list of `freq.table` objects. One for each variable in `data`.

See Also

`table` `xtabs` `descriptive.table` `prop.table`

Examples

```
dat<-data.frame(rnorm(100)>0,trunc(runif(100,0,5)))
##rounding to 1
frequencies(dat)
##rounding to 4
frequencies(dat,4)
```

`get.objects` *get objects*

Description

Enumerates all objects of a certain class

Usage

```
get.objects(cn,env = globalenv(),includeInherited=TRUE)
```

Arguments

`cn`	The name of the class
`env`	environment to look in
`includeInherited`	Should objects inheriting cn be included

Value

a character vector

`ggcorplot` *Correlation matrix*

Description

Plots a correlation matrix

Usage

```
ggcorplot(cor.mat,data=NULL,lines=TRUE,line.method=c("lm","loess"),type="points",
alpha=.25,main="auto",var_text_size=5,
cor_text_limits=c(5,25),level=.05)
```

Arguments

`cor.mat`	a `cor.matrix` object to plot
`data`	the data.frame used to compute the correlation matrix
`lines`	Logical. Should regression lines be drawn.
`type`	type of plot. "points" or "bins"
`line.method`	Character. Type of regression line.
`alpha`	numeric. level of alpha transparency for the points.
`main`	Title of the plot. defaults to the method of cor.mat.
`var_text_size`	size of the diagonal variable names.
`cor_text_limits`	lower and upper bounds for the size of the correlation text.
`level`	the size of the test differentiated by text color.

Author(s)

Mike Lawrence and Ian Fellows

See Also

`cor.matrix` `qscatter_array`

Examples

```
data(mtcars)
corr.mat1<-cor.matrix(variables=d(mpg,carb,carb+rnorm(length(carb))),,
 data=mtcars,
 test=cor.test,
 method='spearman',
alternative=”two.sided”,exact=FALSE)

p<-ggcorplot(corr.mat1,data = mtcars)
print(p)
## Not run:

has.hex<-require(”hexbin”)
if(has.hex){
data(diamonds)
corr.mat<-cor.matrix(variables=d(price,carat,color),,
 data=diamonds,
 test=cor.test,
 method='spearman',
alternative=”two.sided”)

p1 <- ggcorplot(cor.mat=corr.mat,data=diamonds,type=”bins”,
cor_text_limits=c(5,15),
lines=FALSE)
print(p1)
rm('corr.mat')
```

```
}

## End(Not run)
```

`Helpers` *Helpers*

Description

GUI helpers

Usage

```
.assign.classnames()
```

`k.sample.test` *K Sample Test*

Description

Performs a K independent sample test.

Usage

```
k.sample.test(formula,data,test=oneway.test,...)
```

Arguments

`formula`	A formula, the left hand side of which indicated the outcomes, and the right hand side of which contains the factor
`data`	A data.frame
`test`	A function whose first argument is a formula with the outcome on the lhs and the factor on the rhs. The second argument should be the data to be used for the formula. The result of the function should be an object of class `htest`.
`...`	further arguments for func

Value

A `multi.test` object, representing a table of the results of `func` applied to each of the variables.

See Also

`oneway.test` `kruskal.test` `wilcox.test`

Examples

```
dat<-data.frame(a=rnorm(100),b=rnorm(100),c=rnorm(100),d=cut(rnorm(100),4))
k.sample.test(d(a,b)~d,dat)
k.sample.test(dat[,-4]~dat$d,var.equal=TRUE)
k.sample.test(d(a,c)~d,dat,kruskal.test)
```

`likelihood.test` *Likelihood Ratio (G test) for contingency tables*

Description

Performs a likelihood ratio test of independence

Usage

```
likelihood.test(x,y=NULL,conservative=FALSE)
```

Arguments

`x`	A vector or a matrix
`y`	A vector that is ignored if x is a matrix and required if x is a vector
`conservative`	If `TRUE`, the Williams' continuity correction is used

Value

A list with class "htest" containing the following components:

`statistic`	the value the chi-squared test statistic.
`parameter`	the degrees of freedom of the approximate chi-squared distribution of the test statistic.
`p.value`	the p-value for the test.
`method`	a character string indicating the type of test performed, and whether the continuity correction was used.
`data.name`	a character string giving the name(s) of the data.

Author(s)

Pete Hurd and Ian Fellows

See Also

`chisq.test`

Examples

```
data(InsectSprays)
likelihood.test(InsectSprays$count>7,InsectSprays$spray)
```

`multi.test` *multi.test*

Description

Creates a table from a list of htests

Usage

```
multi.test(tests)
```

Arguments

`tests`	A named list of htest objects representing the same test applied to a number of different conditions or variables.

Value

A `multi.test` object, representing a table of the `htest` objects.

`one.sample.test` *One Sample Test*

Description

Performs a one sample test.

Usage

```
one.sample.test(variables,data=NULL,test=t.test,...)
```

Arguments

`variables`	A variable or dataframe of variables
`data`	The data frame in which variables is evaluated
`test`	A function whose first argument is the sample to be tested, and whose result is an object of class `htest`.
`...`	further arguments for func

Value

A `multi.test` object, representing a table of the results of `test` applied to each of the variables.

See Also

`t.test` `shapiro.test`

Examples

```
data(anorexia)

#are subjects' weights at baseline and endpoint significantly different from normal
one.sample.test(variables=d(Prewt,Postwt),
data=anorexia,
test=shapiro.test)

#does CBT work at increasing mean wt
anorexia.sub<-subset(anorexia,Treat=="CBT")
one.sample.test(variables=Postwt-Prewt,
data=anorexia.sub,
test=t.test)
```

`onesample.plot` *onesample.plot*

Description

plots for one sample tests

Usage

```
onesample.plot(variables,data=NULL,test.value,scale=FALSE,type="hist",alpha=.2)
```

Arguments

`variables`	An expression denoting a set of variable.
`data`	A data.frame from which the variables will be selected.
`test.value`	null hypothesis test value
`scale`	scale variables
`type`	type of plot. 'hist' or 'box' are allowed
`alpha`	transparency of points for box plot

Examples

```
data(mtcars)
onesample.plot(variables=d(mpg,cyl,disp,hp,drat,wt,qsec,vs,am,
gear,carb),data=mtcars,type='hist')

onesample.plot(variables=d(mpg,cyl,disp,hp,drat,wt,qsec,vs,am,
gear,carb),data=mtcars,type='box',alpha=1)
```

`oneway.plot` *One Way PLot*

Description

plots a categorical variable against a series of continuous variables

Usage

```
oneway.plot(formula,data=NULL,alpha=.2,
box=TRUE,points=TRUE,scale=FALSE)
```

Arguments

`formula`	A formula, the left hand side of which indicated the outcomes, and the right hand side of which contains the factor
`data`	A data.frame
`alpha`	alpha transparency level for the points.
`box`	prints boxplot
`points`	prints jitter plot
`scale`	standardize the variables prior to plotting

Value

a ggplot object

Examples

```
oneway.plot(d(DriversKilled, drivers, front, rear,  kms, PetrolPrice)~law,as.data.frame(Seatbelts))
```

`perm` *Vector Permutations*

Description

Enumerates all permutations of a vector

Usage

```
perm(vec,duplicates=FALSE)
```

Arguments

`vec`	The vector to permute
`duplicates`	Should duplicate permutations be listed

Value

Returns a matrix where each row is a permutation of vec. All possible permutations are listed, and if `duplicates=TRUE` non-unique permutations are also listed.

See Also

`sample`

Examples

```
perm(1:4)
perm(LETTERS[4:8])
```

`perm.t.test` *Permutation t-test*

Description

Two Sample t-test via monte-carlo permutation

Usage

```
perm.t.test(x,y,statistic=c("t","mean"),
alternative=c("two.sided", "less", "greater"), midp=TRUE, B=10000)
```

Arguments

`x`	a numeric vector containing the first sample
`y`	a numeric vector containing the second sample
`statistic`	The statistic to be permuted. See details
`alternative`	The alternative hypothesis
`midp`	should the mid p-value be used
`B`	The number of monte-carlo samples to be generated

Details

This function performs a two sample permutation test. If the mean is permuted, then the test assumes exchangability between the two samples. if the t-statistic is used, the test assumes either exchangability or a sufficiently large sample size. Because there is little lost in the way of power, and the assumptions are weaker, the t-statistic is used by default.

Value

A list with class "htest" containing the following components:

`statistic`	The observed value of the statistic.
`p.value`	the p-value for the test.
`method`	a character string indicating the type of test performed.
`data.name`	a character string giving the name(s) of the data.
`B`	The number of samples generated
`alternative`	the direction of the test

See Also

`t.test`

Examples

```
perm.t.test(rnorm(100),runif(100,-.5,.5))
```

`plot.cor.matrix` *Plot method*

Description

Produces a circle plot for an object of class "`plot.cor.matrix`"

Usage

```
## S3 method for class 'cor.matrix'
plot(x,y=NULL,size=10,...)
```

Arguments

`x`	Object of class "`cor.matrix`".
`y`	unused
`size`	maximum radius size
`...`	further arguments. unsued

Value

a ggplot object

`print.contin.table` *Print method*

Description

Print object of class "`contin.table`" in nice layout.

Usage

```
## S3 method for class 'contin.table'
print(
x,digits=3,prop.r=TRUE,prop.c=TRUE,prop.t=TRUE,
expected.n=FALSE,residuals=FALSE,std.residuals=FALSE,
adj.residuals=FALSE,no.tables=FALSE,...)
```

Arguments

`x`	Object of class "`contin.table`".
`digits`	Number of digits to round to.
`prop.r`	Logical. print row proportions.
`prop.c`	Logical. print column proportions.
`prop.t`	Logical. print proportions.
`expected.n`	Logical print expected cell counts.
`residuals`	Logical. print residuals.
`std.residuals`	Logical. print standardized residuals.
`adj.residuals`	Logical. Print Adjusted residuals
`no.tables`	Logical. Suppress tables
`...`	further arguments

Value

none

Author(s)

Ian Fellows based on the CrossTable function from the gmodels package maintained by Gregory R. Warnes

---

`print.contin.tests` *Print method*

Description

Print object of class "`contin.tests`" in nice layout.

Usage

```
## S3 method for class 'contin.tests'
print(x,test.digits, ...)
```

Arguments

`x`	Object of class "`contin.tests`".
`test.digits`	Number of digits to be printed
`...`	further arguments to be passed to or from methods.

Value

none

`print.contingency.tables` *Print method*

Description

Print object of class "`contingency.tables`" in nice layout.

Usage

```
## S3 method for class 'contingency.tables'
print(x,digits=3,prop.r=TRUE,prop.c=TRUE,prop.t=TRUE,
expected.n=FALSE,no.tables=FALSE,...)
```

Arguments

`x`	Object of class "`contin.table`".
`digits`	Number of digits to round to.
`prop.r`	Logical. print row proportions.
`prop.c`	Logical. print column proportions.
`prop.t`	Logical. print proportions.
`expected.n`	Logical print expected cell counts.
`no.tables`	Logical. Suppress tables
`...`	further arguments

Value

none

`print.cor.matrix` *Print method*

Description

Print object of class "`cor.matrix`" in nice layout.

Usage

```
## S3 method for class 'cor.matrix'
print(x,digits=4,N=TRUE,CI=TRUE,stat=TRUE,p.value=TRUE,...)
```

Arguments

`x`	Object of class "`cor.matrix`".
`digits`	Number of digits to round to.
`N`	Logical. print a row for sample size.
`CI`	Logical. print a row for confidence intervals if they exist.
`stat`	Logical. print a row for test statistics.
`p.value`	Logical. print a row for p-values.
`...`	further arguments

Value

none

`print.freq.table` *Print method*

Description

Print object of class "`freq.table`" in nice layout.

Usage

```
## S3 method for class 'freq.table'
print(x,...)
```

Arguments

`x`	Object of class "`freq.table`".
`...`	further arguments

Value

none

`print.multi.test` *Print method*

Description

Print object of class "`multi.test`" in nice layout.

Usage

```
## S3 method for class 'multi.test'
print(x,...)
```

Arguments

`x`	Object of class "`multi.test`".
`...`	further arguments

Value

none

`qscatter_array` *qscatter_array*

Description

Creates an array of scatterplots

Usage

```
qscatter_array(variables,with.variables,data,x.lab="",y.lab="",
main="Correlation Array",common.scales=TRUE,alpha=.25)
```

Arguments

`variables`	variables
`with.variables`	An optional set of variables to correlate with `variables`. If nothing is specified, all variables in `variables` are correlated with themselves.
`data`	A data.frame from which the variables will be selected.
`x.lab`	A label for the x axis
`y.lab`	A label for the y axis
`main`	A label for the plot
`common.scales`	should common x and y scales be used.
`alpha`	alpha transparency

Examples

```
data(mtcars)
qscatter_array(d(cyl,disp,hp,drat),
data=mtcars) + geom_smooth(method="lm")
qscatter_array(d(cyl,disp,hp,drat),d(wt,carb),data=mtcars,common.scales=FALSE)
```

`recode.variables` *Recode*

Description

Recodes a set of variables according to a set of rules

Usage

```
recode.variables(data,recodes)
```

Arguments

`data`	A `data.frame` to be recoded
`recodes`	Definition of the recoding rules. See details

Details

`recodes` contains a set of recoding rules separated by ";". There are three different types of recoding rules:

1. The simplest codes one value to another. If we wish to recode 1 into 2, we could use the rule `"1->2;"`.

2. A range of values can be coded to a single value using `"1:3->4;"`. This rule would code all values between 1 and 3 inclusive into 4. For factors, a value is between two levels if it is between them in the factor ordering. One sided ranges can be specified using the Lo and Hi key words (e.g.`"Lo:3->0; 4:Hi->1"`)

3. Default conditions can be coded using "else." For example, if we wish to recode all values >=0 to 1 and all values <0 to missing, we could use (`"0:Hi->1; else->NA"`)

Value

returns a recoded `data.frame`

Author(s)

Ian Fellows adapted from code by John Fox

Examples

```
data<-data.frame(a=rnorm(100),b=rnorm(100),male=rnorm(100)>0)
recode.variables(data[c(”a”,”b”)] , ”Lo:0 -> 0;0:Hi -> 1;”)
data[c(”male”)] <- recode.variables(data[c(”male”)] , ”1 -> 'Male';0 -> 'Female';else -> NA;”)
```

`rocplot` *ROC Plot for a logistic regression model*

Description

Plots the ROC Curve

Usage

```
rocplot(logistic.model,diag=TRUE,pred.prob.labels=FALSE,prob.label.digits=3,AUC=TRUE)
```

Arguments

`logistic.model`	a glm object with binomial link function.
`diag`	a logical value indicating whether a diagonal reference line should be displayed.
`pred.prob.labels`	a logical value indicating whether the predictive probabilities should be displayed
`prob.label.digits`	The number of digits of the predictive probabilities to be displayed.
`AUC`	a logical value indicating whether the estimated area under the curve should be displayed

Value

a ggplot object

Author(s)

Ian Fellows adapted from the lroc function by Virasakdi Chongsuvivatwong

Examples

```
model.glm <- glm(formula=income>5930.5 ~ education + women + type,
family=binomial(),data=Prestige,na.action=na.omit)
rocplot(model.glm)
```

`sort.data.frame` *Sort Data*

Description

Sorts a data frame

Usage

```
## S3 method for class 'data.frame'
sort(x, decreasing, by, ...)
```

Arguments

`x`	A `data.frame` to be sorted
`decreasing`	unused
`by`	A character, a one sided formula, or an expression indicating the sorting order
`...`	further arguments

Details

If `by` is a formula, or a character vector coerce-able into a formula, `x` is sorted by each element of the formula, with ties broken by subsequent elements. Elements preceded by a '-' indicate descending order, otherwise ascending order is used. Parentheses or any formula operator other than + and - are ignored, so sorting by `a*b` will sort based on the product of a and b.

If `by` is not a formula, a ~ is appended to the left hand side of the call, and coerced into a formula.

The decreasing argument is included for generic method consistency, and is not used.

Value

returns `x`, sorted.

Author(s)

Ian Fellows adapted from code by Ari Friedman and Kevin Wright

See Also

`sort` `order`

Examples

```
data(mtcars)

#sort by the number of cylenders
sort(mtcars, by= ~cyl)
sort(mtcars, by= cyl) #identical: no need for ~

#sort in descending order
sort(mtcars, by= -cyl)

#break ties with horse power
sort(mtcars,by= cyl +hp )
sort(mtcars,by= cyl -hp )

#randomly permute the data
sort(mtcars,by= rnorm(nrow(mtcars)) )

#reverse order
sort(mtcars,by= nrow(mtcars):1 )

#sort by squared deviation from mean hp
sort(mtcars,by= -(hp-mean(hp))^2 )
sort(mtcars,by= "-(hp-mean(hp))^2" ) #identical
```

`summarylm` *Summary table for a linear model*

Description

Computes the coefficients, std. errors, t values, and p-values for a linear model in the presence of possible heteroskedasticity.

Usage

```
summarylm(object,correlation=FALSE,symbolic.cor = FALSE,white.adjust=FALSE,...)
```

Arguments

`object`	an object of class lm.
`correlation`	a logical value indicating whether parameter correlations should be printed.
`symbolic.cor`	logical. If TRUE, print the correlations in a symbolic form (see symnum) rather than as numbers. Effective only if white.adjust is FALSE.
`white.adjust`	value passed to `hccm` indicating the type of robust adjustment to be used. If TRUE, type is assumed to be 'hc3'
`...`	additional parameters passed to stats::summary.lm

Details

If white.adjust is false, the function returns a value identical to stats::summary.lm. Otherwise, robust summaries are computed

Value

A summary table

Examples

```
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2,10,20, labels=c("Ctl","Trt"))
weight <- c((ctl-mean(ctl))*10+mean(ctl), trt)
lm.D9 <- lm(weight ~ group)
summarylm(lm.D9,white.adjust=TRUE)
```

`table.to.data` *Table –> data.frame*

Description

Creates a data.frame from a table

Usage

```
table.to.data(x)
```

Arguments

`x`	A matrix or table representing the cross tabulation of two variables

Value

A two column data.frame where each row is an observation and each column is a variable.

See Also

`xtabs`

Examples

```
tab<-matrix(c(4,5,6,9,7,3),ncol=3)
tab
table.to.data(tab)
```

`two.sample.test` *Two Sample Test*

Description

Performs a two independent sample test.

Usage

```
two.sample.test(formula,data=NULL,test=t.test,...)
```

Arguments

`formula`	A formula, the left hand side of which indicated the outcomes, and the right hand side of which contains the factor
`data`	A data.frame
`test`	A function whose first two arguments are the two-samples to be tested, and whose result is an object of class `htest`.
`...`	further arguments for test

Value

A `multi.test` object, representing a table of the results of `test` applied to each of the variables.

See Also

`t.test` `ks.test` `wilcox.test`

Examples

```
dat<-data.frame(a=rnorm(100),b=rnorm(100),c=rnorm(100),d=rnorm(100)>(-.5))
two.sample.test(d(a,b) ~ d,dat,ks.test)
two.sample.test(a ~ dich(b,cut=0) ,dat,t.test)
two.sample.test(d(a^2,abs(b),c)~d,dat,wilcox.test)
```

Index