Package 'DescTools'

May 4, 2014

Type Package

Title Tools for descriptive statistics

Version 0.99.7

Date 2014-05-03

Author Andri Signorell. Includes R source code and/or documentation previously published by (in alphabetical order): Nanina Anderegg, Tomas Aragon, Antti Arppe, Ben Bolker, Stephane Champely, Daniel Chessel, Leanne Chhay, Michael Dewey, Harold C. Doran, Stephane Dray, Charles Dupont, Jeff Enos, Claus Ekstrom, Martin Elff, John Fox, Tal Galili, Matthias Gamer, Juergen Gross, Gabor Grothendieck, Frank E. Harrell Jr, Michael Hoehle, Christian W. Hoffmann, Markus Huerzeler, Rob J. Hyndman, Pablo J. Villacorta Iglesias, David Kahle, Matthias Kohl, Mikko Korpela, Jim Lemon, Martin Maechler, Arni Magnusson, Daniel Malter, Alina Matei, David Meyer, Yongyi Min, Markus Naepflin, Derek Ogle, Sandrine Pavoine, Roland Rapold, William Revelle, Venkatraman E. Seshan, Greg Snow, Michael Smithson, Werner A. Stahel, Yves Tille, Adrian Trapletti, Kevin Ushey, Jeremy VanDerWal, Bill Venables, John Verzani, Gregory R. Warnes, Stefan Wellek, Peter Wolf, Achim Zeileis

Maintainer Andri Signorell <andri@signorell.net>

Description DescTools contains a bunch of basic statistic functions and convenience wrappers for efficiently describing data, creating specific plots, doing reports using MS Word, Excel or PowerPoint. The package's intention is to offer a toolbox, which facilitates the (notoriously time consuming) first descriptive tasks in data analysis, consisting of calculating descriptive statistics, drawing graphical summaries and reporting the results. Many of the included functions can be found scattered in other packages and other sources written partly by Titans of R. The reason for collecting them here, was primarily to have them consolidated in ONE instead of dozens of packages (which themselves might depend on other packages which are not needed at all), and to provide a common and consistent interface as far as function and arguments naming, NA handling, recycling rules etc. are concerned. Google style guides were used as naming rules (in absence of convincing alternatives). The 'camel style' was consequently applied to functions borrowed from contributed R packages as well.

Depends tcltk

Suggests RDCOMClient

Imports boot

License GPL (>= 2)

LazyLoad yes

LazyData yes

ByteCompile yes

NeedsCompilation yes

R topics documented:

`DescTools-package` *Tools for Efficient Descriptive Statistics*

Description

DescTools contains a bunch of basic statistic functions and convenience wrappers for efficiently describing data, creating specific plots and using MS-Office (Word, Excel or PowerPoint) for doing reports.
The package's intention is to assemble a toolbox with basic functions to facilitate the (notoriously time consuming) first descriptive tasks in data analysis, consisting of calculating descriptive statistics, drawing graphical summaries and reporting the results. Many of the included functions can be found scattered in other packages and other sources written partly by Titans of R. The reason for collecting them here, was primarily to have them consolidated in ONE instead of dozens of packages (which themselves might depend on other packages which are not needed at all), and to provide a common and consistent interface as far as function and arguments naming, NA handling, recycling rules etc. are concerned. In this sense this is my "Best-of"-collection of R-functions, I've happened to need in recent years.
Google styleguides were used as naming rules (in absence of convincing alternatives). The 'CamelStyle' was consequently applied to functions borrowed from contributed R packages as well.

Feedback, feature requests, bugreports and other suggestions are welcome!

Details

Package:	DescTools
Type:	Package
Version:	0.99.7
Date:	2014-05-03
Depends:	tcltk
Suggests:	RDCOMClient
Imports:	boot
License:	GPL (>= 2)
LazyLoad:	yes
LazyData:	yes
ByteCompile:	yes
NeedsCompilation:	yes

A grouped list of the functions:

Operators, calculus, transformations:

%()%	Between operators, %()%, %(]%, %[)%, %[]%
%nin%	"not in" operator
%overlaps%	Determines whether two collections have common elements
%like%	Simple operator to search for a specified pattern
AUC	Calculate area under the curve
Primes	Find all primes less than n
Factorize	Prime factorization of integers

GCD	Calculate the greatest common divisor
LCM	Calculate the least common multiple
Permn	Determine all possible permutations of a set
Fibonacci	Generates single Fibonacci numbers or a Fibonacci sequence
Frac	Return the fractional part of a numeric value
Ndec	Count decimal places of a number
BoxCox, BoxCoxInv	Box Cox transformation and inverse transformation
LogGen, LogLin	Log linear hybrid, generalized log
LogSt, LogStInv	Calculate started logarithmic transformation and it's inverse
Logit, LogitInv	Generalized logit and inverse logit function
Winsorize	Data cleaning by winsorization
Recode	Recode a factor with altered levels
Rename	Change name(s) of a named object
Sort	Sort extension for matrices and data.frames
Large, Small	Returns the kth largest, resp. smallest values
Rev	Reverses the order of rows and columns of a matrix
Untable	Recreates original list based on a n-dimensional frequency table
Dummy	Generate dummy codes for a factor
FisherZ, FisherZInv	Fisher's z-transformation and its inverse

Information and manipulation functions:

GetAllSubsets	Generates all possible subsets out of a list of elements
GetPairs	Generates all pairs out of one or two sets of elements
WhichFlags	Returns the names of dichotomous columns of a data.frame
WhichFactors	Returns the names of factors in a given data.frame
WhichNumerics	Returns the names of numerical columns of a data.frame
IsWhole	Is x a whole number?
FctArgs	Retrieve the arguments of a functions
ParseFormula	Parse a formula and return the splitted parts of if
Label	Get or set the label attribute of an object
Mbind	Bind matrices to 3-dimensional arrays
VecRot	Shift the elements of a vector in a circular mode to the right or to the left by n characters.
LOCF	Imputation of data following the "last observation carried forward" rule

String functions:

StrTrim	Delete white spaces from a string
StrTrunc	Truncate string on a given length and add ellipses if it really was truncated
StrAbbr	Abbreviates a string
StrCap	Capitalize the first letter of a string
StrDist	Compute distances between strings
StrRev	Reverse a string
StrCountW	Count the words in a string
StrChop	Split a string by a fixed number of characters.
StrVal	Extract numeric values from a string

Conversion functions:

AscToChar, CharToAsc	Converts ASCII codes to characters and vice versa
DecToBin, BinToDec	Converts numbers from binmode to decimal and vice versa
DecToHex, HexToDec	Converts numbers from hexmode to decimal and vice versa
DecToOct, OctToDec	Converts numbers from octmode to decimal and vice versa
DegToRad, RadToDeg	Convert Degrees to Radians and vice versa
CartToPol, PolToCart	Transform Cartesian to Polar Coordinates and vice versa

Colors:

SetAlpha	Add an alpha channel to a color.
ChooseColorDlg	Display the system's color dialog to choose a color
PlotRCol	Display R colors in a dialog
ColorLegend	Add a color legend to a plot
ColToGray, ColToGrey	Convert colors to gcrey/grayscale
ColToHex, HexToCol	Convert a color into hex string
ColToHsv	R color to HSV conversion
ColToRgb, RgbToCol	Color to RGB conversion and back
FindColor	Get color on a defined color range
TextContrastColor	Choose textcolor depending on background color
PalRedToBlack, PalTibco	Defined color palettes

Plots:

Canvas	Canvas for geometric plotting
AddLoess	Add a loess smoother to an existing plot
AddErrBars	Add horizontal or vertical error bars to an existing plot
DrawArc, DrawRegPolygon	Draw elliptic, circular arc(s) or regular polygon(s)
DrawCircle, DrawEllipse	Draw a circle, ellipse
DrawBezier	Draw a Bezier curve
DrawAnnulus, DrawAnnulusSector	Draw one or several annuli, resp. sector of an annulus
DrawBand	Draw confidence band
BoxedText	Add text surrounded by a box to a plot
Rotate	Rotate a geometric structure
SpreadOut	Spread out a vector of numbers so that there is a minimum interval between any two elements. This can be used to place textlabels in a plot so that they do not overlap.
AreaIdent	Helps identifying all the points in a specific area.
identify.formula	Formula interface for `identify`.
PtInPoly	Identify all the points within a polygon.
PlotACF, PlotMonth	Create a combined plot of a time series and its autocorrelation and partial autocorrelation
PlotArea	Create an area plot
PlotBag	Create a two-dimensional boxplot
PlotBubble	Draw a bubble plot
PlotCandlestick	Plot candlestick chart
PlotCirc	Create a circular plot
PlotCorr	Plot a correlation matrix
PlotDesc	Create a descriptive plot of a vector x dependent on its class
PlotDotCI	Plot a dotchart with confidence intervals
PlotDotCIp	Plot a dotchart with binomial confidence intervals
PlotFaces	Produce a plot of Chernoff faces
PlotFdist	Frequency distribution plot, combination of histogram,

	boxplot and ecdf.plot
PlotMarDens	Scatterplot with marginal densities
PlotMultiDens	Plot multiple density curves
PlotPolar	Plot values on a circular grid
PolarGrid	Plot a grid in polar coordinates
PlotPyramid	Pyramid plot (back-back histogram)
PlotTreemap	Plot of a treemap.
PlotVenn	Plot a Venn diagram
PlotViolin	Plot violins instead of boxplots
PlotQQ	QQ-plot for an optional distribution
PlotWeb	Create a web plot

Statistics:

Freq	Frequency table
PercTable	Two dimensional percentage table
Mode	Mode, the most frequent value
Gmean, Gsd	Geometric mean and geometric standard deviation
Hmean	Harmonic Mean
HuberM, TukeyBiweight	Huber M-estimator of location and Tukey's biweight robust mean
MeanAD	Mean absolute deviation
MeanSE	Standard error of mean
Skew, Kurt	Skewness and kurtosis
YuleQ, YuleY	Yule's Q and Yule's Y
TschuprowT	Tschuprow's T
Phi, ContCoef, CramerV	Phi, Pearson's Contingency Coefficient and Cramer's V
CohenKappa, Kappam	Cohen's Kappa and weighted Kappa and Kappa for more than 2 raters
CronbachAlpha	Cronbach's alpha
ICC	Intraclass correlations
GoodmanKruskalTauA	Goodman Kruskal's tau-a
GoodmanKruskalGamma	Goodman Kruskal's gamma
KendallTauB	Kendall's tau-b
StuartTauC	Stuart's tau-c
SomersDelta	Somers' delta
Lambda	Goodman Kruskal's lambda
UncertCoef	Uncertainty coefficient
TheilU	Theil's U1 and U2 coefficient
Entropy, MutInf	Shannon's entropy, mutual information
OddsRatio, RelRisk	Odds ratio and relative risk
Lc	Calculate and plot Lorenz curve
Gini, Atkinson	Gini- and Atkinson coefficient
Herfindahl, Rosenbluth	Herfindahl- and Rosenbluth coefficient
BinomCI, MultinomCI	Confidence intervals for binomial and multinomial proportions
BinomDiffCI	Calculate confidence interval for a risk difference
MeanCI, MedianCI	Confidence interval for the mean and median
MeanDiffCI	Confidence interval for the difference of two means
VarCI	Confidence interval for the variance
CorCI	Confidence interval for Pearson's correlation coefficient
RobScale	Robust data standardization
RobRange	Robust range

Tests:	
SignTest	Signtest
ZTest	Z–test for known population variance
JonckheereTerpstraTest	Jonckheere-Terpstra test
PageTest	Page test for ordered alternatives
CochranQTest	Cochran's Q-test
SiegelTukeyTest	Siegel-Tukey test for equality in variability
LeveneTest	Levene's test for homogeneity of variance
MosesTest	Moses Test of extreme reactions
RunsTest	Runs test for randomness
JarqueBeraTest	Jarque–Bera Test
AndersonDarlingTest	Anderson-Darling test for normality
CramerVonMisesTest	Cramer-von Mises test for normality
LillieTest	Lilliefors (Kolmogorov-Smirnov) test for normality
PearsonTest	Pearson chi-square test for normality
ShapiroFranciaTest	Shapiro-Francia test for normality
MHChisqTest	Mantel-Haenszel Chisquare test
CochranArmitageTest	Cochran-Armitage test for trend in binomial proportions
BreslowDayTest, WoolfTest	Test for homogeneity on 2x2xk tables over strata

Date functions:	
AddMonths	Add a number of months to a given date
IsDate	Check whether x is a date object
IsWeekend	Check whether x is falls on a weekend
Date	Create a date from numeric representation of year, month, day
Day, Month, Year	Extract part of a date
Week, Weekday	Returns ISO week and weekday of a date
Quarter	Quarter of a date
Yearday, Yearmonth	The day in the year of a date
Zodiac	The zodiac sign of a date :-)

GUI-Helpers:	
ChooseColorDlg	Display color dialog to choose a color
ImportDlg	Get path of a data file to be opened
SelectVarDlg	Select elements of a set by click
PasswordDlg	Display a dialog containing an edit field, showing only ***.
PlotPar	Display the R plot parameters in a dialog

Reporting:	
CatTable	Print a table with the option to have controlled linebreaks
FormatFix	Format a figure with a fixed number of digits
Desc	Produce a rich description of an object
DescWrd	Produce the same description as above but send the results to a Word document and add an adequate graphic representation
GetCurrWrd	Get a handle to a running Word instance
GetNewWrd	Create a new Word Instance
WrdCaption	Insert a title in Word
WrdPlot	Insert the active plot to Word

WrdR	Insert an R command and its output in a Word document
WrdSetFont	Set the font in Word
WrdTable	Insert a table in Word
WrdText	Insert normal text to Word
WrdInsertBookmark	Insert a new bookmark in a Word document
WrdGoto	Place cursor to a specific bookmark, or another text position.
WrdUpdateBookmark	Update the text of a bookmark's range
GetNewXL	Create a new Excel instance
XLGetRange	Get the values of one or several cell range(s) in Excel
XLGetWorkbook	Get the values of all sheets of an Excel workbook
XLView	Use Excel as viewer for a data.frame
GetNewPP	Create a new PowerPoint instance
PpPlot	Insert active plot to PowerPoint

All Wrd*, XL* and Pp* functions require the package RDCOMClient to be installed. Hence the use of these functions is restricted to Windows systems. RDCOMClient is available at: http://www.stats.ox.ac.uk/pub/RWin/bin/windows/contrib/3.0/.

There are a few options for the graphical output that can be set. For the moment these are restricted to footnote signs named "footnote1", "footnote2". For setting another footnote signe write e.g. options("footnote1"="*").

Author(s)

Andri Signorell
Helsana Versicherungen AG, Health Sciences, Zurich
HWZ University of Applied Sciences in Business Administration Zurich.

Includes R source code and/or documentation previously published by (in alphabetical order):
Nanina Anderegg, Tomas Aragon, Antti Arppe, Ben Bolker, Stephane Champely, Daniel Chessel, Leanne Chhay, Michael Dewey, Harold C. Doran, Stephane Dray, Charles Dupont, Jeff Enos, Claus Ekstrom, Martin Elff, John Fox, Tal Galili, Matthias Gamer, Juergen Gross, Gabor Grothendieck, Frank E. Harrell Jr, Michael Hoehle, Christian W. Hoffmann, Markus Huerzeler, Rob J. Hyndman, Pablo J. Villacorta Iglesias, David Kahle, Matthias Kohl, Mikko Korpela, Jim Lemon, Martin Maechler, Arni Magnusson, Daniel Malter, Alina Matei, David Meyer, Yongyi Min, Markus Naepflin, Derek Ogle, Sandrine Pavoine, Roland Rapold, William Revelle, Venkatraman E. Seshan, Greg Snow, Michael Smithson, Werner A. Stahel, Yves Tille, Adrian Trapletti, Kevin Ushey, Jeremy VanDerWal, Bill Venables, John Verzani, Gregory R. Warnes, Stefan Wellek, Peter Wolf, Achim Zeileis

Thank you all!

Maintainer: Andri Signorell <andri@signorell.net>

Examples

```
# distribution plot (combination of histogram, densitycurve, boxplot and ecdf.plot)
old.par <- par(no.readonly=TRUE)
PlotFdist(x=d.pizza$delivery_min, na.rm=TRUE)

# plot multiple density curves
par(old.par)
```

```
PlotMultiDens( split(d.pizza$delivery_min, d.pizza$driver), na.rm=TRUE
              , main="delivery time ~ driver", xlab="delivery time [min]", ylab="density"
               , panel.first=grid())


# areaplot with stapled areas
tab <- table( d.pizza$date, d.pizza$driver )
PlotArea(x=as.Date(rownames(tab)), y=tab, xaxt="n", xlab="Date", ylab="Pizzas delivered" )
# add x-axis and some text labels
xrng <- pretty(range(as.Date(rownames(tab))))
axis(side=1, at=xrng, labels=xrng)
text( x=min(d.pizza$date + .5, na.rm=TRUE), y=cumsum(tab[2,])-2.5,
     label=levels(d.pizza$driver), adj=c(0,0.5), col=TextContrastColor( gray.colors(7)))


# dotchart with confidence intervals
x <- do.call("rbind", tapply( d.pizza$temperature, d.pizza$driver, MeanCI, na.rm=TRUE))
rownames(x) <- levels(d.pizza$driver)
PlotDotCI(x)


# Plot pyramid
xy.pop <- c(3.2,3.5,3.6,3.6,3.5,3.5,3.9,3.7,3.9,3.5,3.2,2.8,2.2,1.8,1.5,1.3,0.7,0.4)
xx.pop <- c(3.2,3.4,3.5,3.5,3.5,3.7,4,3.8,3.9,3.6,3.2,2.5,2,1.7,1.5,1.3,1,0.8)
agelabels <- c("0-4","5-9","10-14","15-19","20-24","25-29","30-34","35-39","40-44",
               "45-49","50-54","55-59","60-64","65-69","70-74","75-79","80-44","85+")

PlotPyramid( xy.pop, xx.pop, ylab=agelabels, lxlab="men", rxlab="women",
             main="Australian population pyramid 2002", col=PalHelsana()[c(6,1)])


# Plot violin
PlotViolin(temperature ~ driver, d.pizza, col="brown", bw="SJ")


# PlotPolar
testlen <- c(sin(seq(0, 1.98*pi, length=100))+2+rnorm(100)/10)
testpos <- seq(0, 1.98*pi, length=100)

PlotPolar(testlen, testpos, type="l", main="Test Polygon", col="blue")
PolarGrid(ntheta=9, col="grey", lty="solid", lblradians=FALSE)

# spiderweb
posmat <- matrix(sample(2:9,30,TRUE),nrow=3)
PlotPolar(posmat, type="l", main="Spiderweb plot", col=2:4, lwd=1:3)
PolarGrid( nr=NA, ntheta=ncol(posmat), alabels=paste("X", 1:ncol(posmat)
                                                    , sep=""), col="grey", lty="solid" )

# radarplot
data(mtcars)
d.car <- scale(mtcars[1:6,1:7], center=FALSE)
# let's have a palette with thransparent colors (alpha = 32)
cols <- paste(colorRampPalette(c("red","yellow","green","blue"), space = "rgb")(6), "32", sep="")
PlotPolar(d.car, type="l", fill=cols, main="Cars in radar")
PolarGrid(nr=NA, ntheta=ncol(d.car), alabels=colnames(d.car), lty="solid", col="black")
par(old.par)
```

```
# PlotBag: Two-dimensional Boxplot
d.frm <- d.pizza[complete.cases(d.pizza[,c("temperature","delivery_min")]),]
PlotBag( x=d.frm$delivery_min, y=d.frm$temperature
         , xlab="delivery_min", ylab="temperature", main="Two-dimensional Boxplot")

# Chernoff faces
par(old.par)
m <- data.frame( lapply( d.pizza[,c("temperature","price","delivery_min","wine_ordered","weekday")]
                         , tapply, d.pizza$driver, mean, na.rm=TRUE))
PlotFaces(m, main = "Driver's characteristics")


# PlotWeb
m <- PairApply(d.diamonds[ ,WhichFactors(d.diamonds)[-7]], CramerV)
PlotWeb(m, col="steelblue", xpd=TRUE, main="Diamonds CramerV" )

m <- cor(d.pizza[,WhichNumerics(d.pizza)], use="pairwise.complete.obs")
PlotWeb(m, xpd=TRUE, main="Pizza Correlations" )

PlotCorr(m, cols=colorRampPalette(c("red", "black", "green"), space = "rgb")(20))
mtext("Correlation plot", side=3, line=3, font=2, cex=1.5)


# histograms were yesterday, use marginal densities instead
# would be best with: x11(7.5,4.7)
PlotMarDens( y=d.pizza$temperature, x=d.pizza$delivery_min, grp=d.pizza$area
           , xlab="delivery_min", ylab="temperature", col=c("brown","orange","lightsteelblue")
             , panel.first= grid()
)
layout(matrix(1))
par(old.par)


# just another convenience wrapper for a combination of boxplot and plot.Design
PlotDescNumFact( price ~ area, data=d.pizza)


# describe data.frame
Desc(d.pizza[,c("driver","price","wrongpizza","date","count")])

# just a few groupwise descriptions on the console
Desc( price ~ operator, data=d.pizza)
Desc( driver ~ operator, data=d.pizza)
Desc( driver ~ operator, data=d.pizza, rfrq=("111"))   # show all rel. frequencies

# without data parameter
Desc( d.pizza$delivery_min ~ d.pizza$driver, digits=1)

# use functions and interactions
Desc( sqrt(price) ~ operator : factor(wrongpizza), data=d.pizza, digits=2)
Desc( log(price+1) ~ cut(delivery_min, breaks=seq(10,90,10))
      , data=d.pizza, digits=c(2,3,4,2,0,3,0,0))



## Not run:
```

```
# we can't run this example in CRAN environment, but here's how it would work:
# export results to Word by just declaring a new output device

wrd <- GetNewWrd(header=TRUE)
Desc(d.pizza[,c("driver","price","wrongpizza","date","count")], wrd=wrd)

Desc( price ~ driver + operator, d.pizza, digits=c(1,1,1,1,0,3,0,0), wrd=wrd )
Desc( factor(weekday) ~ driver, d.pizza, digits=c(1,1,1,1,0,3,0,0), wrd=wrd)

Desc( temperature ~ delivery_min, d.pizza, digits=c(1,1,1,1,0,3,0,0), wrd=wrd )
Desc( log(temperature) ~ log(delivery_min), d.pizza, digits=c(1,1,1,1,0,3,0,0), wrd=wrd )

Desc( log(price+1) ~ cut(delivery_min, breaks=seq(10,90,10))
 , data=d.pizza, digits=c(2,2,2,2,0,3,0,0), wrd=wrd)

## End(Not run)
```

AddConnLines *Add Connection Lines to a Barplot*

Description

Add connection lines to a stacked barplot (beside = TRUE is not supported). The function expects exactly the same arguments, that were used to create the barplot.

Usage

```
AddConnLines(..., lcol = 1, lwd = 1, lty = "solid")
```

Arguments

`...`	the arguments used to create the barplot. (The dots are sent directly to barplot).
`lcol`	the line color of the connection lines. Defaults to black.
`lwd`	the line width for the connection lines. Default is 1.
`lty`	the line type for the connection lines. Line types can either be specified as an integer (0=blank, 1=solid (default), 2=dashed, 3=dotted, 4=dotdash, 5=longdash, 6=twodash) or as one of the character strings `"blank"`, `"solid"`, `"dashed"`, `"dotted"`, `"dotdash"`, `"longdash"`, or `"twodash"`. Default is `"solid"`.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

barplot

Examples

```
tab <- with(
  subset(d.pizza, driver %in% c("Carpenter","Miller","Farmer","Butcher")),
  table(factor(driver), Weekday(date, "a", stringsAsFactor=TRUE))
)
tab

barplot(tab, beside=FALSE, space=1.2)
AddConnLines(tab, beside=FALSE, space=1.2, lcol="grey50", lwd=1, lty=2)

barplot(tab, beside=FALSE, space=1.2, horiz=TRUE)
AddConnLines(tab, beside=FALSE, space=1.2, horiz=TRUE, lcol="grey50", lwd=1, lty=2)


cols <- PalHelsana()[1:4]
b <- barplot(tab, beside=FALSE, horiz=FALSE, col=cols)
AddConnLines(tab, beside=FALSE, horiz=FALSE, lcol="grey50", lwd=1, lty=2)

# set some labels
text(x=b, y=t(apply(rbind(0,tab), 2, Midx)), labels=tab,
    col=(matrix(rep(TextContrastColor(cols), each=ncol(tab)), nrow=nrow(tab), byrow=FALSE )))
```

`AddErrBars` *Add Error Bars to an Existing Plot*

Description

Add Error Bars to an Existing Plot.

Usage

```
AddErrBars(from, to = NULL, pos = NULL, mid = NULL, horiz = FALSE, col = par("fg"),
           lty = par("lty"), lwd = par("lwd"), code = 3, length = 0.05,
         pch = NA, cex.pch = par("cex"), col.pch = par("fg"), bg.pch = par("bg") )
```

Arguments

`from`	coordinates of points **from** which to draw (the lower end of the error bars).
`to`	coordinates of points **to** which to draw (the upper end of the error bars).
`pos`	numeric, position of the error bars. This will either be the x-coordinate in case of vertical error bars and the y-coordinate in case of horizontal error bars.
`mid`	numeric, position of midpoints. Defaults to the mean of from and to.
`horiz`	boolean, TRUE (default) if horizontal error bars are needed.
`col`	the line color.
`lty`	the line type.
`lwd`	line width.

code	integer code, determining kind of arrows to be drawn. If code = 1 an arrowhead is drawn at (x0[i], y0[i]) and if code = 2 an arrowhead is drawn at (x1[i], y1[i]). If code = 3 (default) a head is drawn at both ends of the arrow. Unless length = 0, when no head is drawn.
length	the length of the end lines.
pch	plotting character for the midpoints. No points will be plotted if this is set to NA, which is the default.
cex.pch	the character extension for the plotting characters. Default is par("cex")
col.pch	the color of the plotting characters. Default is par("fg")
bg.pch	the background color of the plotting characters (if pch is set to 21:25). Default is par("bg")
...	the dots are passed to the arrows function.

Details

A short wrapper for plotting error bars by means of arrows.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

AddLoess

Examples

```
par(mfrow=c(2,2))
b <- barplot(1:5, ylim=c(0,6))
AddErrBars(from=1:5-rep(0.5,5), to=1:5+rep(0.5,5), pos=b, length=0.3)

# just on one side
b <- barplot(1:5, ylim=c(0,6))
AddErrBars(from=1:5, to=1:5+rep(0.5,5), pos=b, length=0.2, col="red", code=2, lwd=2)

b <- barplot(1:5, xlim=c(0,6), horiz=TRUE)
AddErrBars(from=1:5, to=1:5+rep(0.2,5), pos=b, horiz=TRUE,  length=0.2, col="red", code=2, lwd=2)

par(xpd=FALSE)
dotchart(1:5, xlim=c(0,6))
AddErrBars(from=1:5-rep(0.2,5), to=1:5+rep(0.2,5), horiz=TRUE, length=0.1)
```

AddLm *Add a Linear Regression Line*

Description

Add a linear regression line to an existing plot. The function first calculates the prediction of a lm object for a reasonable amount of points, then adds the line to the plot and inserts a polygon with the confidence and prediction intervals.

Usage

```
## Default S3 method:
AddLm(x, y, ...)
## S3 method for class 'formula'
AddLm(formula, data, ...)
## S3 method for class 'lm'
AddLm(x, col = hblue, lwd = 2, lty = "solid",
         type = "l", n = 100, conf.level = 0.95, args.cband = NULL,
         pred.level = NA, args.pband = NULL, ...)
```

Arguments

x, y	vectors giving the coordinates of the points in the scatter plot.
formula	a formula specifying the numeric response and one numeric predictor. Will be coerced to a formula if necessary.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which the function is called.
col	linecolor of the line. Default is DescTools's lightblue.
lwd	line width of the line.
lty	line type of the line.
type	character indicating the type of plotting; actually any of the types as in plot.default. Type of plot, defaults to "l".
n	number of points used for plotting the fit.
conf.level	confidence level for the confidence interval. Set this to NA, if no confidence band should be plotted. Default is 0.95.
args.cband	list of arguments for the confidence band, such as color or border (see DrawBand).
pred.level	confidence level for the prediction interval. Set this to NA, if no prediction band should be plotted. Default is 0.95.
args.pband	list of arguments for the prediction band, such as color or border (see DrawBand).
...	further arguments are not used specifically.

Details

It's sometimes illuminating to plot a regression line with it's prediction, resp. confidence intervals over an existing xy-plot. This only makes sense, if just a simple regression model y ~ x is to be visualized.

Value

nothing

Author(s)

Andri Signorell <andri@signorell.net>

See Also

AddLoess, lm

Examples

```
par(mfrow=c(1,2))

plot(hp ~ wt, mtcars)
AddLm(hp ~ wt, mtcars, col="steelblue")

# add the prediction intervals in different color
plot(hp ~ wt, mtcars)
AddLm(hp ~ wt, mtcars, col="red", pred.level=0.95, args.pband=list(col=SetAlpha("grey",0.3)) )
```

AddLoess *Add a Loess Smoother*

Description

Add a loess smoother to an existing plot. The function first calculates the prediction of a loess object for a reasonable amount of points, then adds the line to the plot and inserts a polygon with the confidence intervals.

Usage

```
## Default S3 method:
AddLoess(x, y, ...)

## S3 method for class 'formula'
AddLoess(formula, data, col = hblue, lwd = 2, lty = "solid",
                 type = "l", n = 100, conf.level = 0.95, args.band = NULL, ...)
```

Arguments

x, y	vectors giving the coordinates of the points in the scatter plot.
formula	a formula specifying the numeric response and one to four numeric predictors (best specified via an interaction, but can also be specified additively). Will be coerced to a formula if necessary.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which loess is called.

col	linecolor of the smoother. Default is DescTools's lightblue.
lwd	line width of the smoother.
lty	line type of the smoother.
type	type of plot, defaults to "l".
n	number of points used for plotting the fit.
conf.level	confidence level for the confidence interval. Set this to NA, if no confidence band should be plotted. Default is 0.95.
args.band	list of arguments for the confidence band, such as color or border (see DrawBand).
...	further arguments are passed to loess()

Value

The computed object of class "loess".

Note

Loess can result in heavy computational load if there are many points!

Author(s)

Andri Signorell <andri@signorell.net>

See Also

loess, scatter.smooth

Examples

```
par(mfrow=c(1,2))

x <- runif(100)
y <- rnorm(100)
plot(x, y)
AddLoess(x, y)

# the formula interface is implemented as well:
data(d.pizza)
plot(temperature ~ delivery_min, data=d.pizza)
AddLoess(temperature ~ delivery_min, data=d.pizza)

plot(temperature ~ delivery_min, data=d.pizza)
AddLoess(temperature ~ delivery_min, data=d.pizza, conf.level = 0.99,
         args.band = list(col=SetAlpha("red", 0.4), border="black") )

# the default values from scatter.smooth
AddLoess(temperature ~ delivery_min, data=d.pizza,
         span=2/3, degree=1, family="symmetric", col="red")
```

AddMonths *Add a Month to a Date*

Description

Add a number of months to a date. In case the number is negative, the months will be subtracted. A ceiling to the last available day of the month can be set. The function will then yield as.Date('2013-01-31') + 1 month = '2013-02-28'.

Usage

```
AddMonths(x, n, ceiling = TRUE)
```

Arguments

x	the date to which a number of months has to be added. x can be a date or an integer. If it is an integer it will be interpreted as yyyymm.
n	the number of months to be added. If n is negative the months will be subtracted.
ceiling	logic. If set to TRUE (default), a ceiling to the last available day of month will be set.

Value

a vector with the same dimension and type as x, containing the transformed dates.

Author(s)

Andri Signorell <andri@signorell.net>, based on code by Roland Rapold and Antonio

References

Thanks to Antonio: http://stackoverflow.com/questions/14169620/add-a-month-to-a-date

See Also

Date functions, like Year, Month, etc.

Examples

```
AddMonths(as.Date("2013-01-01"), 10)
AddMonths(as.Date("2013-01-01"), -5)

AddMonths(201301, 3)
AddMonths(201301, -5)

AddMonths(as.Date("2013-01-31"), 1)
AddMonths(as.Date("2013-01-31"), -2)
```

`AllDuplicated` *Index Vector of All Values Involved in Ties*

Description

Returns an index vector of all the values in x which are involved in ties.
So `!AllDuplicated` determines all values in a vector x, which appear only once (frequency 1).

Usage

```
AllDuplicated(x)
```

Arguments

x	vector of any type.

Value

logical vector of the same dimension as x.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`unique` returns a unique list of all values in x
`duplicated` returns an index vector flagging all elements, which appeared more than once (leaving out the first appearance!)
`union`(A, B) returns a list with the unique values from A and B
`intersect` returns all elements which appear in A and in B
`setdiff`(A, B) returns all elements appearing in A but not in B
`setequal`(A, B) returns TRUE if A contains exactly the same elements as B

Examples

```
x <- c(1:10, 4:6)

AllDuplicated(x)

x[!AllDuplicated(x)]


# union, intersect and friends...
A <- c(sort(sample(1:20, 9)),NA)
B <- c(sort(sample(3:23, 7)),NA)

# all elements from A and B (no duplicates)
union(A, B)
# all elements appearing in A and in B
intersect(A, B)
```

```
# elements in A, but not in B
setdiff(A, B)
# elements in B, but not in A
setdiff(B, A)
# Does A contain the same elements as B?
setequal(A, B)


# Find ties in a vector x
x <- sample(letters[1:10], 20, replace=TRUE)
ties <- split(x, x)

# count tied groups
sum(unlist(lapply(ties, function(x) length(x)>1)))

# length of tied groups
lapply(ties, length)[lapply(ties, length)>1]

# by means of table
tab <- table(x)
tab[tab>1]

# count elements involved in ties
sum(tab>1)
# count tied groups
sum(tab[tab>1])
```

`AndersonDarlingTest` *Anderson-Darling test for normality*

Description

Performs the Anderson-Darling test for the composite hypothesis of normality, see e.g. Thode (2002, Sec. 5.1.4).

Usage

```
AndersonDarlingTest(x)
```

Arguments

`x`	a numeric vector of data values, the number of which must be greater than 7. Missing values are allowed.

Details

The Anderson-Darling test is an EDF omnibus test for the composite hypothesis of normality. The test statistic is

$$A = -n - \frac{1}{n}\sum_{i=1}^{n}[2i-1][\ln(p_{(i)}) + \ln(1 - p_{(n-i+1)})],$$

where $p_{(i)} = \Phi([x_{(i)} - \overline{x}]/s)$. Here, Φ is the cumulative distribution function of the standard normal distribution, and $\overline{x}$ and s are mean and standard deviation of the data values. The p-value is computed from the modified statistic $Z = A(1.0 + 0.75/n + 2.25/n^2)\backslash$ according to Table 4.9 in Stephens (1986).

Value

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

`statistic`	the value of the Anderson-Darling statistic.
`p.value`	the p-value for the test.
`method`	the character string "Anderson-Darling normality test".
`data.name`	a character string giving the name(s) of the data.

Note

The Anderson-Darling test is the recommended EDF test by Stephens (1986). Compared to the Cramer-von Mises test (as second choice) it gives more weight to the tails of the distribution.

Note

This function was previously published as `ad.test()` in the **nortest** package and has been integrated here without logical changes.

Author(s)

Juergen Gross <gross@statistik.uni-dortmund.de>

References

Stephens, M.A. (1986) *Tests based on EDF statistics. In: D'Agostino, R.B. and Stephens, M.A., eds.: Goodness-of-Fit Techniques*. Marcel Dekker, New York.

Thode Jr., H.C. (2002) *Testing for Normality*. Marcel Dekker, New York.

See Also

`shapiro.test` for performing the Shapiro-Wilk test for normality. `CramerVonMisesTest`, `LillieTest`, `PearsonTest`, `ShapiroFranciaTest` for performing further tests for normality. `qqnorm` for producing a normal quantile-quantile plot.

Examples

```
AndersonDarlingTest(rnorm(100, mean = 5, sd = 3))
AndersonDarlingTest(runif(100, min = 2, max = 4))
```

AreaIdent *Identify Points in Plot Lying within a Rectangle or Polygon*

Description

Find all the points lying either in a rectangle area, spanned by an upper left point and a bottom-right point clicked by the user or in a polygon area defined by the user.

Usage

```
AreaIdent(formula, data, subset, poly = FALSE)
```

Arguments

formula	a formula, such as y ~ x specifying x and y values. Here the formula must be entered that was used to create the scatterplot.
data	a data frame (or list) from which the variables in formula should be taken.
subset	an optional vector specifying a subset of observations to be used.
poly	logical. Defines if a polygon or a rectangle should be used to select the points. Default is rectangle. If a polygon should be used, set this argument to TRUE and select all desired points. The polygon will be closed automatically when finished.

Value

Index vector with the points lying within the selected area. The coordinates are returned as text in the attribute "cond".

Author(s)

Andri Signorell <andri@signorell.net>

See Also

identify, locator

Examples

```
## Not run:
# run the example via copy and paste

plot(temperature ~ delivery_min, data=d.pizza)
idx <- AreaIdent(temperature ~ delivery_min, data=d.pizza)

# you selected the following points
d.pizza[idx,]
points(temperature ~ delivery_min, data = d.pizza[idx,], col="green")

# use the attr("cond") for subsets in code
attr(idx, "cond")
```

```
# create a group variable for the found points
d.pizza$grp <- seq(nrow(d.pizza)) %in% idx

# try the polygon option
idx <- AreaIdent(temperature ~ delivery_min, data=d.pizza, poly=TRUE)
points(temperature ~ delivery_min, data = d.pizza[idx,], col="red")


## End(Not run)
```

AscToChar *Converts ASCII Codes to Characters and Vice Versa*

Description

AscToChar returns a character for each ASCII code (integer) supplied.
CharToAsc returns integer codes in `0:255` for each (one byte) character in `strings`.

Usage

```
AscToChar(i)
CharToAsc(x)
```

Arguments

`i`	numeric (integer) vector of values in `1:255`.
`x`	`character` vector.

Details

Only codes in `1:127` make up the ASCII encoding which should be identical for all R versions, whereas the *'upper'* half is often determined from the ISO-8859-1 (aka "ISO-Latin 1)" encoding, but may well differ, depending on the locale setting, see also `Sys.setlocale`.

Note that `0` is no longer allowed since, R does not allow `\0` aka `nul` characters in a string anymore.

Value

`AscToChar` and `CharToAsc` return a vector of the same length as their argument.

Author(s)

unknown guy out there, help text partly taken from M. Maechler's **sfmisc**.

Examples

```
(x <- CharToAsc("Silvia"))
paste(AscToChar(x), collapse="")
```

Assocs *Association Measures*

Description

Collects a bunch of association measures for nominal and ordinal data.

Usage

```
Assocs(x, conf.level = 0.95, digits = 4)
```

Arguments

x	a 2 dimensional contingency table or a matrix.
conf.level	confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.
digits	integer which determines the number of digits used in formatting the measures of association.

Details

This function wraps the association measures phi, contingency coefficient, Cramer's V, Goodman Kruskal's Gamma, Kendall's Tau-b, Stuart's Tau-c, Somers' Delta, Pearson and Spearman correlation, Guttman's Lambda, Theil's Uncertainty Coefficient and the mutual information.

Value

numeric matrix, dimension [1:17, 1:3]
the first column contains the estimate, the second the lower confidence interval, the third the upper one.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Phi, ContCoef, CramerV, GoodmanKruskalGamma, KendallTauB, StuartTauC, SomersDelta, SpearmanRho, Lambda, UncertCoef, MutInf

Examples

```
# Example taken from: SAS/STAT(R) 9.2 User's Guide, Second Edition, The FREQ Procedure
# http://support.sas.com/documentation/cdl/en/statugfreq/63124/PDF/default/statugfreq.pdf
# Hair-Eye-Color pp. 1816

tob <- as.table(matrix(c(
  69, 28, 68, 51,  6,
  69, 38, 55, 37,  0,
  90, 47, 94, 94, 16
  ), nrow=3, byrow=TRUE))
```

```
Desc(tob)
Assocs(tob)


# Example taken from: http://www.math.wpi.edu/saspdf/stat/chap28.pdf
# pp. 1349

pain <- as.table(matrix(c(
   26,  6,
   26,  7,
   23,  9,
   18, 14,
    9, 23
   ), ncol=2, byrow=TRUE))

Desc(pain)
Assocs(pain)
```

`Atkinson` *Calculate the Atkinson Index*

Description

Compute the Atkinson Index. This inequality measure is useful in determining which end of the distribution contributed most to the observed inequality.

Usage

```
Atkinson(x, n = rep(1, length(x)), parameter = 0.5, na.rm = FALSE)
```

Arguments

`x`	a vector containing at least non-negative elements.
`n`	a vector of frequencies, must be same length as `x`.
`parameter`	parameter of the inequality measure (if set to `NULL` the default parameter of the respective measure is used).
`na.rm`	logical. Should missing values be removed? Defaults to FALSE.

Value

the value of the Akinson Index.

Note

This function was previously published as `ineq()` in the **ineq** package and has been integrated here without logical changes, but with some extensions for NA-handling and the use of weights.

Author(s)

Achim Zeileis <Achim.Zeileis@R-project.org>

References

Cowell, F. A. (2000) Measurement of Inequality in Atkinson, A. B. / Bourguignon, F. (Eds): *Handbook of Income Distribution*. Amsterdam.

Cowell, F. A. (1995) *Measuring Inequality* Harvester Wheatshef: Prentice Hall.

Marshall, Olkin (1979) *Inequalities: Theory of Majorization and Its Applications*. New York: Academic Press.

See Also

See `Herfindahl`, `Rosenbluth` for concentration measures and `ineq()` in the package **ineq** for additional inequality measures

Examples

```
# generate vector (of incomes)
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)

# compute Atkinson coefficient with parameter=0.5
Atkinson(x, parameter=0.5)
```

AUC *Area Under the Curve*

Description

Calculate the area under the curve with a naive algorithm and with a more elaborated spline approach. The curve must be given by vectors of xy-coordinates.

Usage

```
AUC(x, y, method = c("trapezoid", "step", "spline"), na.rm = FALSE)
```

Arguments

`x, y`	the xy-points of the curve
`method`	can be `"trapezoid"` (default), `"step"` or `"spline"`.
`na.rm`	logical, indicating whether NA values should be stripped before the computation proceeds. In this case only the complete.cases of x and y will be used. `na.rm` defaults to `FALSE`.

Details

If method is set to `"trapezoid"` then the curve is formed by connecting all points by a direct line (composite trapezoid rule). If `"step"` is chosen then a stepwise connection of two points is used. For calculating the area under a spline interpolation the `splinefun` function is used in combination with `integrate`.
The AUC function will handle unsorted x values (by sorting x) and ties for the x values (by ignoring duplicates).

Value

Numeric value of the area under the curve.

Author(s)

Andri Signorell <andri@signorell.net>, spline part by Claus Ekstrom <claus@rprimer.dk>

See Also

integrate, splinefun

Examples

```
AUC(x=c(1,3), y=c(1,1))

AUC(x=c(1,2,3), y=c(1,2,4), method="trapezoid")
AUC(x=c(1,2,3), y=c(1,2,4), method="step")

plot(x=c(1,2,2.5), y=c(1,2,4), type="l", col="blue", ylim=c(0,4))
lines(x=c(1,2,2.5), y=c(1,2,4), type="s", col="red")

x <- seq(0, pi, length.out=200)
AUC(x=x, y=sin(x))
AUC(x=x, y=sin(x), method="spline")
```

AxisBreak *Place a Break Mark on an Axis*

Description

Places a break mark on an axis on an existing plot.

Usage

```
AxisBreak(axis = 1, breakpos = NULL, pos = NA, bgcol = "white",
          breakcol = "black", style = "slash", brw = 0.02)
```

Arguments

axis	which axis to break.
breakpos	where to place the break in user units.
pos	position of the axis (see axis).
bgcol	the color of the plot background.
breakcol	the color of the "break" marker.
style	Either 'gap', 'slash' or 'zigzag'.
brw	break width relative to plot width.

Details

The 'pos' argument is not needed unless the user has specified a different position from the default for the axis to be broken.

Note

There is some controversy about the propriety of using discontinuous coordinates for plotting, and thus axis breaks. Discontinuous coordinates allow widely separated groups of values or outliers to appear without devoting too much of the plot to empty space.
The major objection seems to be that the reader will be misled by assuming continuous coordinates. The 'gap' style that clearly separates the two sections of the plot is probably best for avoiding this.

Author(s)

Jim Lemon and Ben Bolker

Examples

```
plot(3:10, main="Axis break test")

# put a break at the default axis and position
AxisBreak()
AxisBreak(2, 2.9, style="zigzag")
```

Between *Between Operators Check, if a Value Lies Within a Given Range*

Description

The between operators are used to check, whether given values x lie within a defined range. The values can be numbers, text or dates. Ordered factors are supported.

Usage

```
x %()% rng
x %(]% rng
x %[)% rng
x %[]% rng
```

Arguments

x	is a variable with at least ordinal scale, usually a numeric value, but can be an ordered factor or a text as well. Texts would be treated alphabetically.
rng	a vector of two values, defining the minimum and maximum of the range for x

Details

The between operators basically combine two conditional statements into one and simplify so the query process.
They are merely a wrapper for: `ifelse (x >= rng[1] & x <= rng[2], TRUE, FALSE)`, where the round bracket (means "strictly greater than >" and the square bracket [means ">=". Numerical values of x will be handled by C-code, which is significantly faster than two comparisons in R (especially when x is huge). .
SQL-guys might like these kind of thinking.

Value

A logical vector of the same length as x.

Author(s)

Andri Signorell <andri@signorell.net> based on C-code by Kevin Ushey <kevinushey@gmail.com>

See Also

`if`, `ifelse`, `Comparison`

Examples

```
x <- 1:9
x %[]% c(3,5)
c(x,NA) %[]% c(3,5)

x %(]% c(3,5)

# no result when from > to:
x %[]% c(5,3)
x %(]% c(5,5)

# no problem:
ordered(x) %[]% c(3,5)

# not meaningful:
factor(x) %[]% c(3,5)

# characters
letters[letters %(]% c("d","h")]

data(d.pizza)
x <- levels(d.pizza$driver)
x %[]% c("C","G")

# select diamonds with a price between 2400 and 2510
data(d.diamonds)
d.diamonds[d.diamonds$price %[]% c(2400,2510),]

# use it with an ordered factor and select all diamonds with
#   symmetry between G (included) and X (excluded).
mean(d.diamonds[d.diamonds$symmetry %[)% c("G","X"),"price"])
```

BinomCI *Confidence Intervals for Binomial Proportions*

Description

Compute confidence intervals for binomial proportions following the most popular methods. (Wald, Wilson, Agresti-Coull, Jeffreys, Clopper-Pearson etc.)

Usage

```
BinomCI(x, n, conf.level = 0.95,
        method = c("wilson", "wald", "agresti-coull", "jeffreys",
                   "modified wilson", "modified jeffreys",
                   "clopper-pearson", "arcsine", "logit", "witting"),
        rand = 123)
```

Arguments

x	number of successes.
n	number of trials.
conf.level	confidence level, defaults to 0.95.
method	character string specifing which method to use; this can be one out of: "wald", "wilson", "agresti-coull", "jeffreys", "modified wilson", "modified jeffreys", "clopper-pearson", "arcsine", "logit" or "witting". Defaults to "wilson". Abbreviation of method are accepted. See details.
rand	seed for random number generator; see details.

Details

All arguments are being recycled.

The Wald interval is obtained by inverting the acceptance region of the Wald large-sample normal test.

The Wilson interval, which is the default, was introduced by Wilson (1927) and is the inversion of the CLT approximation to the family of equal tail tests of p = p0. The Wilson interval is recommended by Agresti and Coull (1998) as well as by Brown et al (2001).

The Agresti-Coull interval was proposed by Agresti and Coull (1998) and is a slight modification of the Wilson interval. The Agresti-Coull intervals are never shorter than the Wilson intervals; cf. Brown et al (2001).

The Jeffreys interval is an implementation of the equal-tailed Jeffreys prior interval as given in Brown et al (2001).

The modified Wilson interval is a modification of the Wilson interval for x close to 0 or n as proposed by Brown et al (2001).

The modified Jeffreys interval is a modification of the Jeffreys interval for `x == 0 | x == 1` and `x == n-1 | x == n` as proposed by Brown et al (2001).

The Clopper-Pearson interval is based on quantiles of corresponding beta distributions. This is sometimes also called exact interval.

The arcsine interval is based on the variance stabilizing distribution for the binomial distribution.

The logit interval is obtained by inverting the Wald type interval for the log odds.

The Witting interval (cf. Beispiel 2.106 in Witting (1985)) uses randomization to obtain uniformly optimal lower and upper confidence bounds (cf. Satz 2.105 in Witting (1985)) for binomial proportions.

For more details we refer to Brown et al (2001) as well as Witting (1985).

Value

A vector with 3 elements for estimate, lower confidence intervall and upper for the upper one.

Note

This function was previously published as `binomCI()` in the **SLmisc** package and has been integrated here with some adaptations concerning the interface, but without any change in the computation logic.

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>, Andri Signorell <andri@signorell.net> (interface issues)

References

A. Agresti and B.A. Coull (1998) Approximate is better than "exact" for interval estimation of binomial proportions. *American Statistician*, **52**, pp. 119-126.

L.D. Brown, T.T. Cai and A. Dasgupta (2001) Interval estimation for a binomial proportion. *Statistical Science*, **16**(2), pp. 101-133.

H. Witting (1985). *Mathematische Statistik I*. Stuttgart: Teubner.

See Also

`binom.test`, `binconf`, `MultinomCI`

Examples

```
BinomCI(x=37, n=43, method=c("wald", "wilson", "agresti-coull", "jeffreys",
  "modified wilson", "modified jeffreys", "clopper-pearson", "arcsine", "logit", "witting")
)


# the confidence interval computed by binom.test
#   corresponds to the Clopper-Pearson interval
BinomCI(x=42, n=43, method="clopper-pearson")
binom.test(x=42, n=43)$conf.int


# all arguments are being recycled:
BinomCI(x=c(42, 35, 23, 22), n=43, method="wilson")
BinomCI(x=c(42, 35, 23, 22), n=c(50, 60, 70, 80), method="jeffreys")
```

BinomDiffCI *Confidence Interval for a Difference of Binomials*

Description

Calculate confidence interval for a risk difference.

Usage

```
BinomDiffCI(x1, n1, x2, n2, conf.level = 0.95,
            method = c("wald", "waldcor", "ac", "exact", "newcombe",
                       "newcombecor", "fm", "ha"))
```

Arguments

x1	number of successes for the first group.
n1	number of trials for the first group.
x2	number of successes for the second group.
n2	number of trials for the second group.
conf.level	confidence level, defaults to 0.95.
method	one out of "wald", "waldcor", "ac", "exact", "newcombe", "newcombecor", "fm", "ha".

Details

Still to follow.

Value

A vector with 3 elements for estimate, lower confidence intervall and upper for the upper one.

Author(s)

Andri Signorell <andri@signorell.net>

References

Fagerland M W, Lydersen S and Laake P (2011) Recommended confidence intervals for two independent binomial proportions, Statistical Mehtods in Medical Research 0(0) 1-31

http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_freq_a0000000642.htm#statug.freq.freqbincp

See Also

BinomCI, MultinomCI, binom.test, prop.test

Examples

```
BinomDiffCI(14, 70, 32, 80, method="wald")
BinomDiffCI(14, 70, 32, 80, method="waldcor")
BinomDiffCI(14, 70, 32, 80, method="ac")
BinomDiffCI(14, 70, 32, 80, method="newcombe")
```

BinToDec *Converts numbers from binmode, octmode or hexmode to decimal and vice versa*

Description

These functions convert numbers from one base to another. There are several solutions for this problem out there, but the naming is quite heterogeneous and so consistent function names might be helpful.

Usage

```
BinToDec(x)
DecToBin(x)
OctToDec(x)
DecToOct(x)
HexToDec(x)
DecToHex(x)
```

Arguments

x	a vector of numbers, resp. alphanumerical representation of numbers (hex), to be converted.

Details

BinToDec converts numbers from binary mode into decimal values. DecToBin does it the other way round.
Oct means octal system and hex hexadecimal.

Value

A numeric or character vector of the same length as x containing the converted values. Binary, octal and decimal values are numeric, hex-values are returned in character mode.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

strtoi

Examples

```
DecToBin(c(17, 25))
BinToDec(c(101, 11101))

DecToOct(c(17, 25))
OctToDec(c(11, 77))

DecToHex(c(17, 25))
```

```
HexToDec(c("FF", "AA", "ABC"))
```

BoxCox *Box Cox Transformation*

Description

BoxCox() returns a transformation of the input variable using a Box-Cox transformation. BoxCoxInv() reverses the transformation.

Usage

```
BoxCox(x, lambda)
BoxCoxInv(x, lambda)
```

Arguments

x	a numeric vector
lambda	transformation parameter

Details

The Box-Cox transformation is given by

$$f_\lambda(x) = \frac{x^\lambda - 1}{\lambda}$$

if $\lambda \neq 0$. For $\lambda = 0$,

$$f_0(x) = \log(x)$$

.

Value

a numeric vector of the same length as x.

Note

These two functions are borrowed from library(forecast).

Author(s)

Rob J Hyndman <rob.hyndman@monash.edu>

References

Box, G. E. P. and Cox, D. R. (1964) An analysis of transformations. *JRSS B* **26** 211–246.

See Also

Use BoxCoxLambda or boxcox in library(MASS) to find optimal lambda values.

Examples

```
# example by Greg Snow
x <- rlnorm(500, 3, 2)

par(mfrow=c(2,2))
qqnorm(x, main="Lognormal")
qqnorm(BoxCox(x, 1/2), main="BoxCox(lambda=0.5)")
qqnorm(BoxCox(x, 0), main="BoxCox(lambda=0)")

hist(BoxCox(x, 0))

bx <- BoxCox( x, lambda = BoxCoxLambda(x) )
```

`BoxCoxLambda` *Automatic selection of Box Cox transformation parameter*

Description

An automatic selection of the Box Cox transformation parameter is estimated with two methods. Guerrero's (1993) method yields a lambda which minimizes the coefficient of variation for subseries of `x`. For method `"loglik"`, the value of lambda is chosen to maximize the profile log likelihood of a linear model fitted to `x`. For non-seasonal data, a linear time trend is fitted while for seasonal data, a linear time trend with seasonal dummy variables is used.

Usage

```
BoxCoxLambda(x, method=c("guerrero","loglik"), lower=-1, upper=2)
```

Arguments

`x`	a numeric vector or time series
`method`	Choose method to be used in calculating lambda.
`lower`	Lower limit for possible lambda values.
`upper`	Upper limit for possible lambda values.

Value

a number indicating the Box-Cox transformation parameter.

Note

This function was previously published as `BoxCox.lambda()` in the **forecast** package and has been integrated here without logical changes.

Author(s)

Leanne Chhay and Rob J Hyndman

References

Box, G. E. P. and Cox, D. R. (1964) An analysis of transformations. *JRSS B* **26** 211–246.

Guerrero, V.M. (1993) Time-series analysis supported by power transformations. *Journal of Forecasting*, **12**, 37–48.

See Also

BoxCox

Examples

```
lambda <- BoxCoxLambda(AirPassengers,lower=0)
```

BoxedText *Add Text in a Box to a Plot*

Description

BoxedText draws the strings given in the vector labels at the coordinates given by x and y, surrounded by a rectangle.

Usage

```
BoxedText(x, y = NULL, labels = seq_along(x), adj = NULL, pos = NULL, offset = 0.5,
          vfont = NULL, cex = 1, txt.col = NULL, font = NULL, srt = 0,
          xpad = 0.2, ypad = 0.2, density = NULL, angle = 45, col = "white",
          border = par("fg"), lty = par("lty"), lwd = par("lwd"), ...)
```

Arguments

x, y	numeric vectors of coordinates where the text labels should be written. If the length of x and y differs, the shorter one is recycled.
labels	a character vector or expression specifying the text to be written. An attempt is made to coerce other language objects (names and calls) to expressions, and vectors and other classed objects to character vectors by as.character. If labels is longer than x and y, the coordinates are recycled to the length of labels.
adj	The value of adj determines the way in which text strings are justified. A value of 0 produces left-justified text, 0.5 (the default) centered text and 1 right-justified text. (Any value in [0, 1] is allowed, and on most devices values outside that interval will also work.) Note that the adj argument of text also allows adj = c(x, y) for different adjustment in x- and y- directions.
pos	a position specifier for the text. If specified this overrides any adj value given. Values of 1, 2, 3 and 4, respectively indicate positions below, to the left of, above and to the right of the specified coordinates.
offset	when pos is specified, this value gives the offset of the label from the specified coordinate in fractions of a character width.

vfont	NULL for the current font family, or a character vector of length 2 for Hershey vector fonts. The first element of the vector selects a typeface and the second element selects a style. Ignored if labels is an expression.
cex	numeric character expansion factor; multiplied by par("cex") yields the final character size. NULL and NA are equivalent to 1.0.
txt.col, font	the color and (if vfont = NULL) font to be used, possibly vectors. These default to the values of the global graphical parameters in par().
srt	The string rotation in degrees.
xpad, ypad	The proportion of the rectangles to the extent of the text within.
density	the density of shading lines, in lines per inch. The default value of NULL means that no shading lines are drawn. A zero value of density means no shading lines whereas negative values (and NA) suppress shading (and so allow color filling).
angle	angle (in degrees) of the shading lines.
col	color(s) to fill or shade the rectangle(s) with. The default NA (or also NULL) means do not fill, i.e., draw transparent rectangles, unless density is specified.
border	color for rectangle border(s). The default is par("fg"). Use border = NA to omit borders (this is the default). If there are shading lines, border = TRUE means use the same colour for the border as for the shading lines.
lty	line type for borders and shading; defaults to "solid".
lwd	line width for borders and shading. Note that the use of lwd = 0 (as in the examples) is device-dependent.
...	additional arguments are passed to the text function.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

SpreadOut, similar function in package **plotrix** boxed.labels (lacking rotation option)

Examples

```
Canvas(xpd=TRUE)

BoxedText(0, 0, adj=0, label="This is boxed text", srt=seq(0,360,20), xpad=.3, ypad=.3)
points(0,0, pch=15)
```

`BreslowDayTest` *Breslow-Day Test for Homogeneity of the Odds Ratios*

Description

Calculates a Breslow-Day test of homogeneity for a $2 \times 2 \times k$ tables, to see if all strata have the same OR. If OR is not given, the Mantel-Haenszel estimate is used.

Usage

```
BreslowDayTest(x, OR = NA, correct = FALSE)
```

Arguments

`x`	a $2 \times 2 \times k$ table.
`OR`	the odds ratio to be tested against. If left undefined (default) the Mantel-Haenszel estimate will be used.
`correct`	If TRUE, the Breslow-Day test with Tarone's adjustment is computed, which subtracts an adjustment factor to make the resulting statistic asymptotically chi-square.

Details

For the Breslow-Day test to be valid, the sample size should be relatively large in each stratum, and at least 80% of the expected cell counts should be greater than 5. Note that this is a stricter sample size requirement than the requirement for the Cochran-Mantel-Haenszel test for tables, in that each stratum sample size (not just the overall sample size) must be relatively large. Even when the Breslow-Day test is valid, it might not be very powerful against certain alternatives, as discussed in Breslow and Day (1980).

Author(s)

Michael Hoehle <hoehle@math.su.se>

References

source: https://onlinecourses.science.psu.edu/stat504/sites/onlinecourses.science.psu.edu.stat504/files/lesson04/breslowday.test_.R

Breslow, N. E., N. E. Day (1980) The Analysis of Case-Control Studies *Statistical Methods in Cancer Research: Vol. 1*. Lyon, France, IARC Scientific Publications.

Tarone, R.E. (1985) On heterogeneity tests based on efficient scores, *Biometrika*, 72, pp. 91-95.

Jones, M. P., O'Gorman, T. W., Lemka, J. H., and Woolson, R. F. (1989) A Monte Carlo Investigation of Homogeneity Tests of the Odds Ratio Under Various Sample Size Configurations *Biometrics*, 45, 171-181

Breslow, N. E. (1996) Statistics in Epidemiology: The Case-Control Study *Journal of the American Statistical Association*, 91, 14-26.

See Also

WoolfTest

Examples

```
migraine <- xtabs(freq ~ .,
            cbind(expand.grid(treatment=c("active","placebo"),
                              response=c("better","same"),
                              gender=c("female","male")),
                  freq=c(16,5,11,20,12,7,16,19))
            )

# get rid of gender
tab <- xtabs(Freq ~ treatment + response, migraine)
Desc(tab)

# only the women
female <- migraine[,,1]
Desc(female)

# .. and the men
male <- migraine[,,2]
Desc(male)

BreslowDayTest(migraine)
BreslowDayTest(migraine, correct = TRUE)


salary <- array(
      c(38,12,102,141,12,9,136,383),
      dim=c(2,2,2),
      dimnames=list(exposure=c("exposed","not"),
                    disease=c("case","control"),
                    salary=c("<1000",">=1000"))
                    )

# common odds ratio = 4.028269
BreslowDayTest(salary, OR = 4.02)
```

Canvas *Canvas for Geometric Plotting*

Description

This is just a wrapper for creating an empty plot with suitable defaults for plotting geometric shapes.

Usage

```
Canvas(xlim = NULL, ylim = xlim, xpd=par("xpd"), mar=c(5.1,5.1,5.1,5.1), asp = 1, ...)
```

Arguments

xlim, ylim	the xlims and ylims for the plot. Default is c(-1, 1).
xpd	expand drawing area, defaults to par("xpd").
mar	set margins. Defaults to c(5.1,5.1,5.1,5.1).
asp	numeric, giving the aspect ratio y/x. (See plot.window for details. Default is 1.
...	additional arguments are passed to the plot() command.

Details

The plot is created with these settings:
asp = 1, xaxt = "n", yaxt = "n", xlab = "", ylab = "", frame.plot = FALSE.

Author(s)

Andri Signorell <andri@signorell.net>

Examples

```
Canvas(7)
text(0, 0, "Hello world!", cex=5)
```

CartToPol *Transform Cartesian to Polar Coordinates and vice versa*

Description

Transform cartesian into polar coordinates.

Usage

```
CartToPol(x, y)
PolToCart(r, theta)
```

Arguments

x, y	two vectors with the xy-coordianates to be transformed.
r	a vector with the radius of the points.
theta	a vector with the angle(s) of the points.

Details

Angles are in radians, not degrees (i.e., a right angle is pi/2). Use DegToRad to convert, if you don't wanna do it by yourself.
All parameters are recycled if necessary.

Value

PolToCart returns a list of x and y coordinates of the points.
CartToPol returns a list of r for the radius and theta for the angles of the given points.

Author(s)

Andri Signorell <andri@signorell.net>

Examples

```
CartToPol(x=1, y=1)
CartToPol(x=c(1,2,3), y=c(1,1,1))
CartToPol(x=c(1,2,3), y=1)


PolToCart(r=1, theta=pi/2)
PolToCart(r=c(1,2,3), theta=pi/2)
```

CatTable *Function to write a table*

Description

CatTable helps printing a table, if is has to be broken into multiple rows. Rowlabels will be repeated after every new break.

Usage

```
CatTable(tab, wcol, nrepchars, width = getOption("width"))
```

Arguments

tab	the rows of a table to be printed, pasted together in one string with constant columnwidth.
wcol	integer, the width of the columns. All columns must have the same width.
nrepchars	integer, the number of characters to be repeated with every break. This is typically the maximum width of the rowlabels.
width	integer, the width of the whole table. Default is the width of the current command window (getOption("width")).

Author(s)

Andri Signorell <andri@signorell.net>

See Also

table, paste

Examples

```
# used in bivariate description functions
Desc(temperature ~ cut(delivery_min, breaks=40), data=d.pizza)


txt <- c(
   paste(sample(letters, 500, replace=TRUE), collapse="")
 , paste(sample(letters, 500, replace=TRUE), collapse="")
 , paste(sample(letters, 500, replace=TRUE), collapse="")
)
txt <- paste(c("aaa","bbb","ccc"), txt, sep="")

CatTable(txt, nrepchars=3, wcol=5)
```

ChooseColorDlg *Display Color Dialog to Choose a Color*

Description

Choose a color by means of the system's color dialog. Nice for looking up RGB-values of any color.

Usage

```
ChooseColorDlg()
```

Value

RGB value of the selected color

Author(s)

Andri Signorell <andri@signorell.net>

See Also

PlotRCol

ClipToVect — *Reformat a Table in the Clipboard as Vector*

Description

A table in the clipboard can not easily be inserted in the R-code. Though it can be accessed by `read.table(clipboard)`, for saving purposes it then has to be saved by `write.table`, which seems clumsy if the file has to be used by others. ClipToVect reformats the table in the way that it can be defined as a data.frame and so be pasted directly in the source code. The option will of course only be interesting for small datasets.

Usage

```
ClipToVect(doubleQuote = TRUE)
```

Arguments

`doubleQuote`	logical. Defines if text should be put in doubleQuotes or singleQuotes. Default is double quotes.

Details

For using this function just copy a cell range in Excel for exmple, then run `ClipToVect` and insert the clipboard in the code file.

Value

the formatted text is copied to clipboard.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`dput`

Clockwise — *Calculates Begin and End Angle From a List of Given Angles in Clockwise Mode*

Description

Transforms given angles in counter clock mode into clockwise angles.

Usage

```
Clockwise(x, start = 0)
```

Arguments

x	a vector of angles
start	the starting angle for the transformation. Defaults to 0.

Details

Sometimes there's need for angles being defined the other way round.

Value

a data.frame with two columns, containing the start and end angles.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

PlotPolar

Examples

```
Clockwise( c(0, pi/4, pi/2, pi))
```

Coalesce *Return the First Element Not Being NA*

Description

Return the first element of a vector, not being NA.

Usage

```
Coalesce(..., method = c("is.na", "is.finite"))
```

Arguments

...	the elements to be evaluated. This can either be a single vector, several vectors of same length, a matrix, a data.frame or a list of vectors (of same length). See examples.
method	one out of "is.na" (default) or "is.finite". The "is.na" option allows Inf values to be in the result, the second one eliminates them.

Details

The evaluation will be rowwise. The first element of the result is the first non NA element of the first elements of all the arguments, the second element of the result is the one of the second elements of all the arguments and so on.
Shorter inputs (of non-zero length) are NOT recycled.
The idea is borrowed from SQL. Might sometimes be useful when preparing data in R instead of in SQL.

Value

return a single vector of the first non NA element(s) of the given data structure.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

is.na, is.finite

Examples

```
Coalesce(c(NA, NA, NA, 5, 3))
Coalesce(c(NA, NULL, "a"))
Coalesce(NULL, 5, 3)

d.frm <- data.frame(matrix(c(
  1,2,NA,4,
  NA,NA,3,1,
  NaN,2,3,1,
  NA,Inf,1,1), nrow=4, byrow=TRUE)
)

Coalesce(d.frm)
Coalesce(as.matrix(d.frm))
Coalesce(d.frm[,2], d.frm[,3:4])
Coalesce(list(d.frm[,1], d.frm[,2]))

# returns the first finite element
Coalesce(d.frm, method="is.finite")
```

CochranArmitageTest *Cochran-Armitage test for trend*

Description

Perform a Cochran Armitage test for trend in binomial proportions across the levels of a single variable. This test is appropriate only when one variable has two levels and the other variable is ordinal. The two-level variable represents the response, and the other represents an explanatory variable with ordered levels. The null hypothesis is the hypothesis of no trend, which means that the binomial proportion is the same for all levels of the explanatory variable.

Usage

```
CochranArmitageTest(x, alternative = c("two.sided", "increasing", "decreasing"))
```

Arguments

x	a frequency table or a matrix.
alternative	alternativea character string specifying the alternative hypothesis, must be one of "two.sided" (default), "increasing" or "decreasing". You can specify just the initial letter.

Value

A list of class htest, containing the following components:

statistic	the Z-statistic of the test.
parameter	the dimension of the table.
p.value	the p-value for the test.
alternative	a character string describing the alternative hypothesis.
method	the character string "Cochran-Armitage test for trend".
data.name	a character string giving the names of the data.

Author(s)

Andri Signorell <andri@signorell.net> strongly based on code from Eric Lecoutre <lecoutre@stat.ucl.ac.be>
http://tolstoy.newcastle.edu.au/R/help/05/07/9442.html

References

Agresti, A. (2002) *Categorical Data Analysis*. John Wiley & Sons

See Also

prop.trend.test

Examples

```
# http://www.lexjansen.com/pharmasug/2007/sp/sp05.pdf, pp. 4
dose <- matrix(c(10,9,10,7, 0,1,0,3), byrow=TRUE, nrow=2, dimnames=list(resp=0:1, dose=0:3))
Desc(dose)
CochranArmitageTest(dose, "increasing")
CochranArmitageTest(dose)
CochranArmitageTest(dose, "decreasing")


# not exactly the same as in package coin:
# independence_test(tumor ~ dose, data = lungtumor, teststat = "quad")
lungtumor <- data.frame(dose = rep(c(0, 1, 2), c(40, 50, 48)),
                        tumor = c(rep(c(0, 1), c(38, 2)),
                                  rep(c(0, 1), c(43, 7)),
                                  rep(c(0, 1), c(33, 15))))
tab <- table(lungtumor$dose, lungtumor$tumor)
CochranArmitageTest(tab)

# but similar to
prop.trend.test(tab[,1], apply(tab,1, sum))
```

`CochranQTest` *Cochran's Q test*

Description

Perform the Cochran's Q test for unreplicated randomized block design experiments with a binary response variable and paired data.

Usage

```
CochranQTest(y, ...)

## Default S3 method:
CochranQTest(y, groups, blocks, ...)

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

Arguments

`y`	either a numeric vector of data values, or a data matrix.
`groups`	a vector giving the group for the corresponding elements of y if this is a vector; ignored if y is a matrix. If not a factor object, it is coerced to one.
`blocks`	a vector giving the block for the corresponding elements of y if this is a vector; ignored if y is a matrix. If not a factor object, it is coerced to one.
`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 matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
`subset`	an optional vector specifying a subset of observations to be used.
`na.action`	a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
`...`	further arguments to be passed to or from methods.

Details

CochranQTest can be used for analyzing unreplicated complete block designs (i.e., there is exactly one binary observation in y for each combination of levels of groups and blocks) where the normality assumption may be violated.

The null hypothesis is that apart from an effect of blocks, the location parameter of y is the same in each of the groups.

If y is a matrix, groups and blocks are obtained from the column and row indices, respectively. NA's are not allowed in groups or blocks; if y contains NA's, corresponding blocks are removed.

Note that Cochran's Q Test is analogue to the Friedman test with 0, 1 coded response. This is used here for a simple implementation.

Value

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

`statistic`	the value of Cochran's chi-squared statistic.
`parameter`	the degrees of freedom of the approximate chi-squared distribution of the test statistic.
`p.value`	the p-value of the test.
`method`	the character string "Cochran's Q-Test".
`data.name`	a character string giving the names of the data.

Author(s)

Andri Signorell <andri@signorell.net>

Examples

```
# example in:
# http://support.sas.com/documentation/cdl/en/statugfreq/63124/PDF/default/statugfreq.pdf
# pp. S. 1824

# use expand.grid, xtabs and Untable to create the dataset
d.frm <- Untable(xtabs(c(6,2,2,6,16,4,4,6) ~ .,
    expand.grid(rep(list(c("F","U")), times=3))),
    colnames = LETTERS[1:3])

# rearrange to long shape
d.long <- reshape(d.frm, varying=1:3, times=names(d.frm)[c(1:3)],
                  v.names="resp", direction="long")


# after having done the hard work of data organisation, performing the test is a piece of cake....
CochranQTest(resp ~ time | id, data=d.long)
```

CohenKappa *Cohen's Kappa and Weighted Kappa*

Description

Computes the agreement rates Cohen's kappa and weighted kappa and their confidence intervals.

Usage

```
CohenKappa(x, y = NULL, weights = c("Unweighted", "Equal-Spacing", "Fleiss-Cohen"),
           conf.level = NA, ...)
```

Arguments

x	can either be a numeric vector or a confusion matrix. In the latter case x must be a square matrix.
y	NULL (default) or a vector with compatible dimensions to x. If y is provided, table(x, y, ...) is calculated. In order to get a square matrix, x and y are coerced to factors with synchronized levels.
weights	either one of the character strings given in the default value, or a user-specified matrix with same dimensions as x.
conf.level	confidence level of the interval. If set to NA (which is the default) no confidence intervals will be calculated.
...	further arguments are passed to the function table, allowing i.e. to set useNA. This refers only to the vector interface.

Details

Cohen's kappa is the diagonal sum of the (possibly weighted) relative frequencies, corrected for expected values and standardized by its maximum value.
The equal-spacing weights are defined by 1 - abs(i - j) / (r - 1), r number of columns/rows, and the Fleiss-Cohen weights by 1 - abs(i - j)^2 / (r - 1)^2. The latter one attaches greater importance to near disagreements.

Value

a single numeric value if no confidence intervals are requested,
and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Note

The vector interface (x, y) is only supported for the calculation of unweighted kappa. For 2 factors with different levels we cannot ensure a reproducible construction of a confusion table, which is independent of the order of x and y. All weights would lead to inconsistent results. The function will raise an error in such cases.

Author(s)

David Meyer <david.meyer@r-project.org>, some slight changes Andri Signorell <andri@signorell.net>

References

Cohen, J. (1960) A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20, 37-46.

Everitt, B.S. (1968), Moments of statistics kappa and weighted kappa. *The British Journal of Mathematical and Statistical Psychology*, 21, 97-103.

Fleiss, J.L., Cohen, J., and Everitt, B.S. (1969), Large sample standard errors of kappa and weighted kappa. *Psychological Bulletin*, 72, 332-327.

See Also

CronbachAlpha, Kappam

Examples

```
# from Bortz et. al (1990) Verteilungsfreie Methoden in der Biostatistik, Springer, pp. 459
m <- matrix(c(53,5,2,  11,14,5,  1,6,3), nrow=3, byrow=TRUE,
            dimnames = list(rater1 = c("V","N","P"), rater2 = c("V","N","P")) )

# confusion matrix interface
CohenKappa(m, weight="Unweighted")

# vector interface
x <- Untable(m)
CohenKappa(x$rater1, x$rater2, weight="Unweighted")

# pairwise Kappa
rating <- data.frame(
  rtr1 = c(4,2,2,5,2, 1,3,1,1,5, 1,1,2,1,2, 3,1,1,2,1, 5,2,2,1,1, 2,1,2,1,5),
  rtr2 = c(4,2,3,5,2, 1,3,1,1,5, 4,2,2,4,2, 3,1,1,2,3, 5,4,2,1,4, 2,1,2,3,5),
  rtr3 = c(4,2,3,5,2, 3,3,3,4,5, 4,4,2,4,4, 3,1,1,4,3, 5,4,4,4,4, 2,1,4,3,5),
  rtr4 = c(4,5,3,5,4, 3,3,3,4,5, 4,4,3,4,4, 3,4,1,4,5, 5,4,5,4,4, 2,1,4,3,5),
  rtr5 = c(4,5,3,5,4, 3,5,3,4,5, 4,4,3,4,4, 3,5,1,4,5, 5,4,5,4,4, 2,5,4,3,5),
  rtr6 = c(4,5,5,5,4, 3,5,4,4,5, 4,4,3,4,5, 5,5,2,4,5, 5,4,5,4,5, 4,5,4,3,5)
)

PairApply(rating, FUN=CohenKappa)

# Weighted Kappa
cats <- c("<10%", "11-20%", "21-30%", "31-40%", "41-50%", ">50%")
m <- matrix(c(5,8,1,2,4,2, 3,5,3,5,5,0, 1,2,6,11,2,1,
              0,1,5,4,3,3, 0,0,1,2,5,2, 0,0,1,2,1,4), nrow=6, byrow=TRUE,
            dimnames = list(rater1 = cats, rater2 = cats) )
CohenKappa(m, weight="Equal-Spacing")


# supply an explicit weight matrix
ncol(m)
(wm <- outer(1:ncol(m), 1:ncol(m), function(x, y) {
        1 - ((abs(x-y)) / (ncol(m)-1)) } ))
CohenKappa(m, weight=wm, conf.level=0.95)


# however, Fleiss, Cohen and Everitt weight similarities
fleiss <- matrix(c(
  106, 10,4,
  22,28, 10,
  2, 12,  6
  ), ncol=3, byrow=TRUE)

#Fleiss weights the similarities
weights <- matrix(c(
 1.0000, 0.0000, 0.4444,
 0.0000, 1.0000, 0.6666,
 0.4444, 0.6666, 1.0000
 ), ncol=3)

CohenKappa(fleiss, weights)
```

`ColorLegend` *Add a ColorLegend to a Plot*

Description

Add a color legend, an image of a sequence of colors, to a plot.

Usage

```
ColorLegend(x, y = NULL, cols = rev(heat.colors(100)), labels = NULL,
            width = NULL, height = NULL, horiz = FALSE,
          xjust = 0, yjust = 1, inset = 0, border = NA, frame = NA, cntrlbl = FALSE,
            adj = ifelse(horiz, c(0.5, 1), c(1, 0.5)), cex = 1, ...)
```

Arguments

`x`	the left x-coordinate to be used to position the colorlegend. See 'Details'.
`y`	the top y-coordinate to be used to position the colorlegend. See 'Details'.
`cols`	the color appearing in the colorlegend.
`labels`	a vector of labels to be placed at the right side of the colorlegend.
`width`	the width of the colorlegend.
`height`	the height of the colorlegend.
`horiz`	logical indicating if the colorlegend should be horizontal; default FALSE means vertical alignment.
`xjust`	how the colorlegend is to be justified relative to the colorlegend x location. A value of 0 means left justified, 0.5 means centered and 1 means right justified.
`yjust`	the same as `xjust` for the legend y location.
`inset`	inset distance(s) from the margins as a fraction of the plot region when colorlegend is placed by keyword.
`border`	defines the bordor color of each rectangle. Default is none (`NA`).
`frame`	defines the bordor color of the frame around the whole colorlegend. Default is none (`NA`).
`cntrlbl`	defines, whether the labels should be printed in the middle of the color blocks or start at the edges of the colorlegend. Default is `FALSE`, which will print the extreme labels centered on the edges.
`adj`	text alignment, horizontal and vertical.
`cex`	character extension for the labels, default 1.0.
`...`	further arguments are passed to the function `text`.

Details

The labels are placed at the right side of the colorlegend and are reparted uniformly between y and y - height.

The location may also be specified by setting x to a single keyword from the list `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"top"`, `"topright"`, `"right"` and `"center"`. This

places the colorlegend on the inside of the plot frame at the given location. Partial argument matching is used. The optional inset argument specifies how far the colorlegend is inset from the plot margins. If a single value is given, it is used for both margins; if two values are given, the first is used for x- distance, the second for y-distance.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

legend, FindColor

Examples

```
plot(1:15,, xlim=c(0,10), type="n", xlab="", ylab="", main="Colorstrips")
ColorLegend(x="right", inset=0.1, labels=c(1:10))

# Center the labels
ColorLegend(x=1, y=9, height=6, col=colorRampPalette(c("blue", "white", "red"),
  space = "rgb")(5), labels=1:5, cntrlbl = TRUE)

ColorLegend(x=3, y=9, height=6, col=colorRampPalette(c("blue", "white", "red"),
  space = "rgb")(5), labels=1:4, frame="grey")

ColorLegend(x=5, y=9, height=6, col=colorRampPalette(c("blue", "white", "red"),
  space = "rgb")(10), labels=sprintf("%.1f",seq(0,1,0.1)), cex=0.8)

ColorLegend(x=1, y=2, width=6, height=0.2, col=rainbow(500), labels=1:5,horiz=TRUE)

ColorLegend(x=1, y=14, width=6, height=0.5, col=colorRampPalette(
  c("red","yellow","green","blue","black"), space = "rgb")(100), horiz=TRUE)

ColorLegend(x=1, y=12, width=6, height=1, col=colorRampPalette(c("red","yellow",
  "green","blue","black"), space = "rgb")(10), horiz=TRUE, border="black")
```

ColToGrey *Convert Colors to Grey/Grayscale*

Description

Convert colors to grey/grayscale so that you can see how your plot will look after photocopying or printing to a non-color printer.

Usage

```
ColToGrey(col)
ColToGray(col)
```

Arguments

col	vector of any of the three kind of R colors, i.e., either a color name (an element of colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see rgb), or an integer i meaning palette()[i]. Non-string values are coerced to integer.

Details

Converts colors to greyscale using the formula grey = 0.3*red + 0.59*green + 0.11*blue. This allows you to see how your color plot will approximately look when printed on a non-color printer or photocopied.

Value

A vector of colors (greys) corresponding to the input colors.

Note

These function was previously published as `Col2Grey()` in the **TeachingDemos** package and has been integrated here without logical changes.

Author(s)

Greg Snow <greg.snow@imail.org>

See Also

`grey`, `ColToRgb`, dichromat package

Examples

```
par(mfcol=c(2,2))
tmp <- 1:3
names(tmp) <- c('red','green','blue')

barplot(tmp, col=c('red','green','blue'))
barplot(tmp, col=ColToGrey(c('red','green','blue')))

barplot(tmp, col=c('red','#008100','#3636ff'))
barplot(tmp, col=ColToGrey(c('red','#008100','#3636ff')))
```

ColToHex *Convert a Color into Hex String*

Description

Convert a color given by name, by its palette index or by rgb-values into a string of the form "#rrggbb" or "#rrggbbaa".

Usage

```
ColToHex(col, alpha = 1)
```

Arguments

col	vector of any of either a color name (an element of colors()), or an integer i meaning palette()[i]. Non-string values are coerced to integer.
alpha	the alpha value to be used. This can be any value from 0 (fully transparent) to 1 (opaque). Default is 1.

Value

Returns the colorvalue in #rrggbb" or #rrggbbaa" format. (character)

Author(s)

Andri Signorell <andri@signorell.net>

See Also

HexToCol, ColToRgb, colors

Examples

```
ColToHex(c("lightblue", "salmon"))

x <- ColToRgb("darkmagenta")
x[2,] <- x[2,] + 155
RgbToCol(x)
```

ColToHsv *R Color to HSV Conversion*

Description

ColToHsv transforms colors from R color into HSV space (hue/saturation/value).

Usage

```
ColToHsv(col, alpha = FALSE)
```

Arguments

col	vector of any of the three kind of R colors, i.e., either a color name (an element of colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa", or an integer i meaning palette()[i]. Non-string values are coerced to integer.
alpha	logical value indicating whether alpha channel (opacity) values should be returned.

Details

Converts a color first into RGB an from there into HSV space by means of the functions rgb2hsv and col2rgb.

Value (brightness) gives the amount of light in the color. Hue describes the dominant wavelength. Saturation is the amount of Hue mixed into the color.

An HSV colorspace is relative to an RGB colorspace, which in R is sRGB, which has an implicit gamma correction.

Value

A matrix with a column for each color. The three rows of the matrix indicate hue, saturation and value and are named "h", "s", and "v" accordingly.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

rgb2hsv, ColToRgb

Examples

```
ColToHsv("peachpuff")
ColToHsv(c(blu = "royalblue", reddish = "tomato")) # names kept

ColToHsv(1:8)
```

ColToRgb *Color to RGB Conversion*

Description

R color to RGB (red/green/blue) conversion.

Usage

```
ColToRgb(col, alpha = FALSE)
```

Arguments

col	vector of any of the three kind of R colors, i.e., either a color name (an element of colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa", or an integer i meaning palette()[i]. Non-string values are coerced to integer.
alpha	logical value indicating whether alpha channel (opacity) values should be returned.

Details

This is merely a wrapper to col2rgb, defined in order to follow this package's naming conventions.

Value

A matrix with a column for each color. The three rows of the matrix indicate red, green and blue value and are named "red", "green", and "blue" accordingly. The matrix might have a 4th row if an alpha channel is requested.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

col2rgb

Examples

```
ColToRgb("peachpuff")
ColToRgb(c(blu = "royalblue", reddish = "tomato")) # names kept

ColToRgb(1:8)
```

ConDisPairs *Concordant and Discordant Pairs*

Description

This function counts concordant and discordant pairs for two variables x, y with at least ordinal scale, aggregated in a 2way table. This is the base for many association measures like Goodman Kruskal's gamma, but also all tau measures.

Usage

```
ConDisPairs(x)
```

Arguments

x	a 2-dimensional table. The column and the row order must be the logical one.

Details

The code is so far implemented in R (O(n^2)) and therefore slow for large sample sizes (>5000). An O(n log(n)) implementation is on track.

Value

a list with the number of concordant pairs, the number of discordant pairs and the matrix

Author(s)

Andri Signorell <andri@signorell.net>

References

Agresti, A. (2002) *Categorical Data Analysis*. John Wiley & Sons, pp. 57-59.

Goodman, L. A., & Kruskal, W. H. (1954) Measures of association for cross classifications. *Journal of the American Statistical Association*, 49, 732-764.

Goodman, L. A., & Kruskal, W. H. (1963) Measures of association for cross classifications III: Approximate sampling theory. *Journal of the American Statistical Association*, 58, 310-364.

http://support.sas.com/onlinedoc/913/getDoc/en/statug.hlp/freq_sect18.htm
http://support.sas.com/onlinedoc/913/getDoc/en/statug.hlp/freq_sect20.htm

See Also

Association measures:
GoodmanKruskalTauA (tau-a), cor (method="kendall") for tau-b, StuartTauC (tau-c), SomersDelta Lambda, UncertCoef, MutInf

Examples

```
tab <- as.table(rbind(c(26,26,23,18,9),c(6,7,9,14,23)))
ConDisPairs(tab)
```

CramerV — *Cramer's V, Pearson's Contingency Coefficient and Phi Coefficient Yule's Q and Y, Tschuprow's T*

Description

Calculate Cramer's V, Pearson's contingency coefficient and phi, Yule's Q and Y and Tschuprow's T for a table, a matrix or a data.frame.

Usage

```
Phi(x, y = NULL, ...)
ContCoef(x, y = NULL, correct = FALSE, ...)
CramerV(x, y = NULL, conf.level = NA, ...)

YuleQ(x, y = NULL, ...)
YuleY(x, y = NULL, ...)
TschuprowT(x, y = NULL, ...)
```

Arguments

x	can be a numeric vector, a matrix or a table.
y	NULL (default) or a vector with compatible dimensions to x. If y is provided, table(x, y) is calculated.
conf.level	confidence level of the interval. This is only implemented for Cramer's V. If set to NA (which is the default) no confidence interval will be calculated. See examples for how to compute bootstrap intervals.

correct	logical. The argument is only used by ContCoef and indicates, whether the Sakoda's adjusted Pearson's C should be returned or not. Default is FALSE.
...	further arguments are passed to the function table, allowing i.e. to set useNA.

Details

For x either a matrix or two vectors x and y are expected. In latter case table(x, y, ...) is calculated. The function handles NAs the same way the table function does, so tables are by default calculated with NAs omitted.

A provided matrix is interpreted as a contingency table, which seems in the case of frequency data the natural interpretation (this is what chisq.test expects).

Use the function PairApply (pairwise apply) if the measure should be calculated pairwise for all columns. This allows matrices of association measures to be calculated the same way cor does. NAs are by default omitted pairwise, which corresponds to the pairwise.complete option of cor. Use complete.cases, if only the complete cases of a data.frame are to be used. (see examples)

The maximum value for Phi is sqrt(min(r, c) - 1), for the corrected contingency coefficient and for Cramer's V it's 1. The minimum value for all is 0 under statistical independence.

Value

a single numeric value if no confidence intervals are requested,
and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net>, Michael Smithson (confidence interval for Cramer V)

References

Yule, G. Uday (1912) On the methods of measuring association between two attributes. *Journal of the Royal Statistical Society, LXXV*, 579-652

Tschuprow, A. A. (1939) *Principles of the Mathematical Theory of Correlation*, translated by M. Kantorowitsch. W. Hodge & Co.

Cramer, H. (1946) *Mathematical Methods of Statistics*. Princeton University Press

Agresti, Alan (1996) *Introduction to categorical data analysis*. NY: John Wiley and Sons

Sakoda, J.M. (1977) Measures of Association for Multivariate Contingency Tables, *Proceedings of the Social Statistics Section of the American Statistical Association* (Part III), 777-780.

Smithson, M.J. (2003) *Confidence Intervals, Quantitative Applications in the Social Sciences Series*, No. 140. Thousand Oaks, CA: Sage. pp. 39-41

See Also

table, PlotCorr, PairApply

Examples

```
data(d.pizza)

tab <- table(d.pizza$driver, d.pizza$wine_delivered)
Phi(tab)
ContCoef(tab)
CramerV(tab)
TschuprowT(tab)

# just x and y
CramerV(d.pizza$driver, d.pizza$wine_delivered)

# data.frame
PairApply(d.pizza[,c("driver","operator","area")], CramerV)


# useNA is passed to table
PairApply(d.pizza[,c("driver","operator","area")], CramerV, useNA="ifany")

d.frm <- d.pizza[,c("driver","operator","area")]
PairApply(d.frm[complete.cases(d.frm),], CramerV)


m <- as.table(matrix(c(2,4,1,7), nrow=2))
YuleQ(m)
YuleY(m)


# Bootstrap confidence intervals for Cramer's V
# http://support.sas.com/documentation/cdl/en/statugfreq/63124/PDF/default/statugfreq.pdf, p. 1821

tab <- as.table(rbind(
  c(26,26,23,18, 9),
  c( 6, 7, 9,14,23)))
d.frm <- Untable(tab)

n <- 1000
idx <- matrix(sample(nrow(d.frm), size=nrow(d.frm) * n, replace=TRUE), ncol=n, byrow=FALSE)
v <- apply(idx, 2, function(x) CramerV(d.frm[x,1], d.frm[x,2]))
quantile(v, probs=c(0.025,0.975))

# compare this to the analytical ones
CramerV(tab, conf.level=0.95)
```

`CramerVonMisesTest` *Cramer-von Mises test for normality*

Description

Performs the Cramer-von Mises test for the composite hypothesis of normality, see e.g. Thode (2002, Sec. 5.1.3).

Usage

```
CramerVonMisesTest(x)
```

Arguments

x	a numeric vector of data values, the number of which must be greater than 7. Missing values are allowed.

Details

The Cramer-von Mises test is an EDF omnibus test for the composite hypothesis of normality. The test statistic is

$$W = \frac{1}{12n} + \sum_{i=1}^{n}(p_{(i)} - \frac{2i-1}{2n}),$$

where $p_{(i)} = \Phi([x_{(i)} - \overline{x}]/s)$. Here, Φ is the cumulative distribution function of the standard normal distribution, and $\overline{x}$ and s are mean and standard deviation of the data values. The p-value is computed from the modified statistic $Z = W(1.0 + 0.5/n)$ according to Table 4.9 in Stephens (1986).

Value

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

statistic	the value of the Cramer-von Mises statistic.
p.value	the p-value for the test.
method	the character string "Cramer-von Mises normality test".
data.name	a character string giving the name(s) of the data.

Author(s)

Juergen Gross <gross@statistik.uni-dortmund.de>

References

Stephens, M.A. (1986) Tests based on EDF statistics In: D'Agostino, R.B. and Stephens, M.A., eds.: *Goodness-of-Fit Techniques*. Marcel Dekker, New York.

Thode Jr., H.C. (2002) *Testing for Normality* Marcel Dekker, New York.

See Also

shapiro.test for performing the Shapiro-Wilk test for normality. AndersonDarlingTest, LillieTest, PearsonTest, ShapiroFranciaTest for performing further tests for normality. qqnorm for producing a normal quantile-quantile plot.

Examples

```
CramerVonMisesTest(rnorm(100, mean = 5, sd = 3))
CramerVonMisesTest(runif(100, min = 2, max = 4))
```

CronbachAlpha *Cronbach's Coefficient Alpha*

Description

Computes Cronbach's alpha. Cronbach's alpha determines the internal consistency or average correlation of items in a survey instrument to gauge its reliability. This reduces to KR-20 when the columns of the data matrix are dichotomous.

Usage

```
CronbachAlpha(x, conf.level = NA, cond = FALSE)
```

Arguments

x	A data frame or matrix with item responses.
conf.level	confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.
cond	logical. If set to TRUE, alpha is additionally calculated for the dataset with each item left out.

Value

Either a numeric value or
a named vector of 3 columns if confidence levels are required (estimate, lower and upper ci) or

a list containing the following components, if the argument cond is set to TRUE:

unconditional	Cronbach's Alpha, either the single value only or with confidence intervals
condCronbachAlpha	The alpha that would be realized if the item were excluded

Author(s)

Andri Signorell <andri@signorell.net>, based on code of Harold C. Doran

References

Cohen, J. (1960), A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20, 37-46.

See Also

CohenKappa, Kappam

Examples

```
set.seed(1234)
tmp <- data.frame(
  item1 = sample(c(0,1), 20, replace=TRUE),
  item2 = sample(c(0,1), 20, replace=TRUE),
  item3 = sample(c(0,1), 20, replace=TRUE),
  item4 = sample(c(0,1), 20, replace=TRUE),
  item5 = sample(c(0,1), 20, replace=TRUE)
  )

CronbachAlpha(tmp[,1:4], cond=FALSE, conf.level=0.95)
CronbachAlpha(tmp[,1:4], cond=TRUE, conf.level=0.95)

CronbachAlpha(tmp[,1:4], cond=FALSE)
CronbachAlpha(tmp[,1:2], cond=TRUE, conf.level=0.95)

## Not run:
# Calculate bootstrap confidence intervals for CronbachAlpha
library(boot)
cronbach.boot <- function(data,x) {CronbachAlpha(data[x,])[[3]]}
res <- boot(datafile, cronbach.boot, 1000)
quantile(res$t, c(0.025,0.975))  # two-sided bootstrapped confidence interval of Cronbach's alpha
boot.ci(res, type="bca")        # adjusted bootstrap percentile (BCa) confidence interval (better)

## End(Not run)
```

CutQ *Build Quantile Groups for a Numeric Variable*

Description

Divides the range of x into some quantiles and codes the values in x according to which interval they fall.

Usage

```
CutQ(x, quant = c(.25,.5,.75,1), labels = gettextf("Q%s", 1:4), na.rm = FALSE)
```

Arguments

x	a numeric vector which is to be converted to a factor by cutting.
quant	the breaks for the quantiles, default is quartiles.
labels	labels for the levels of the resulting category. By default, labels are defined as Q1, Q2, Q3 and Q4. The parameter ist passed to cut, so if labels = FALSE, simple integer codes are returned instead of a factor.
na.rm	logical, indicating whether NA values should be stripped before the computation of the quantile proceeds. Defaults to FALSE. NAs are reported as NA afterwards.

Value

A factor is returned, unless `labels = FALSE` which results in an integer vector of level codes.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

cut, quantile

Examples

```
boxplot(temperature ~ CutQ(temperature, na.rm = TRUE), data=d.pizza)
```

`d.diamonds` *Data diamonds*

Description

As I suppose, an artificial dataset

Usage

```
data(d.diamonds)
```

Format

A data frame with 440 observations on the following 10 variables.

`index` a numeric vector
`carat` a numeric vector
`colour` a factor with levels `D E F G H I J K L`
`clarity` an ordered factor with levels `I2 < I1 < SI3 < SI2 < SI1 < VS2 < VS1 < VVS2 < VVS1`
`cut` an ordered factor with levels `F < G < V < X < I`
`certification` a factor with levels `AGS DOW EGL GIA IGI`
`polish` an ordered factor with levels `F < G < V < X < I`
`symmetry` an ordered factor with levels `F < G < V < X < I`
`price` a numeric vector
`wholesaler` a factor with levels `A B C`

Details

P Poor F Fair G Good V Very good X Excellent I Ideal

Source

somewhere from the net...

Examples

```
data(d.diamonds)
str(d.diamonds)
```

d.pizza *Data pizza*

Description

An artificial dataset inspired by a similar dataset pizza.sav in *Arbeitsbuch zur deskriptiven und induktiven Statistik* by Toutenburg et.al.
The dataset contains data of three pizza services in three districts. Every record defines an order and the according properties. A pizza is supposed to taste good, if it's temperature is high enough, say 45 Celsius. So it might be interesting for the pizza delivery service to minimize the delivery time.

Usage

```
data(d.pizza)
```

Format

A data frame with 1209 observations on the following 17 variables.

`index` a numeric vector, indexing the records (no missings here).

`date` Date, the delivery date

`week` integer, the weeknumber

`weekday` integer, the weekday

`area` factor with levels `Westminster` `Kensington` `Notting Hill`

`count` integer, the number of pizzas delivered

`rabate` numeric, a potential rabate

`price` numeric, the total price of delivered pizza(s)

`operator` a factor with levels `Allanah` `Christal` `Rhonda`

`driver` a factor with levels `Carpenter` `Carter` `Taylor` `Butcher` `Hunter` `Miller` `Farmer`

`delivery_min` numeric, the delivery time in minutes (decimal)

`temperature` numeric, the temperature of the pizza in degrees Celsius when delivered to the customer

`wine_ordered` boolean, TRUE if wine was ordered

`wine_delivered` boolean, TRUE if wine was delivered

`wrongpizza` logical, TRUE if a wrong pizza was delivered

`quality` ordered factor with levels `low` < `mid` < `hot`, defining the quality of the pizza when delivered

Details

The dataset contains NAs randomly scattered.

References

Toutenburg H, Schomaker M, Wissmann M, Heumann C (2009): *Arbeitsbuch zur deskriptiven und induktiven Statistik* Springer, Berlin Heidelberg

Examples

```
str(d.pizza)
head(d.pizza)

Desc(d.pizza)
```

`d.world` *Some World Population and Geographic Data*

Description

Some data about the population in the world.

Usage

```
data(d.world)
```

Format

A data frame with 30 observations on the following 4 variables.

`country` a factor with levels `Angola` `Argentina` `Australia` `Bangladesh` `Brazil` `Canada` `Colombia` `Egypt` `France` `Germany` `Iceland` `India` `Indonesia` `Iran` `Italy` `Japan` `Kenya` `Luxembourg` `Mexico` `Nigeria` `Norway` `Pakistan` `PR China` `Russia` `South Africa` `Turkey` `United Kingdom` `United States` `Zimbabwe`

`x` a numeric vector

`y` a numeric vector

`pop` a numeric vector

Details

There is just the population and the coordinates.

Source

Somewhere out there....

Examples

```
data(d.world)
```

`Date` *Create a Date from Numeric Representation*

Description

Create a date out of either year, month and day supplied by single values or out of one single numeric valu in the form yyyymmdd.

Usage

```
Date(year, month = NA, day = NA)
```

Arguments

`year, month, day`
: numerical values to specify a day. If month and day are omitted the function tries to interpret year as yearmonthday (yyyymmdd) long integer value.

Details

All arguments are recycled if necessary.

The function is a convenience wrapper for ISOdate, which yields a datetime object, which again is an overkill, if only dates are needed.

Value

a vector with the same length as the input vector of class `"Date"`.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`ISOdate`

Examples

```
Date(2011, 3, 1:15)

Date(20120305:20120315)
```

day.name *Build-in Constants Extension*

Description

There's a small number of built-in constants in R. We have month.name and month.abb but nothing similar for days. Here it is.

Usage

```
day.name
day.abb
```

See Also

month.name, month.abb

DegToRad *Convert Degrees to Radians and vice versa*

Description

Convert degrees to radians (and back again).

Usage

```
DegToRad(deg)
RadToDeg(rad)
```

Arguments

deg	a vector of angles in degrees.
rad	a vector of angles in radians.

Value

DegToRad returns a vector of the same length as deg with the angles in radians.
RadToDeg returns a vector of the same length as rad with the angles in degrees.

Author(s)

Andri Signorell <andri@signorell.net>

Examples

```
DegToRad(c(90,180,270))

RadToDeg( c(0.5,1,2) * pi)
```

Desc *Describe Data*

Description

Produce summaries of various types of variables. Calculate descriptive statistics for x and use Word as reporting tool for the numeric results and for descriptive plots. The appropriate statistics are chosen depending on the class of x. The general intention is to simplify the description process for lazy typers and return a quick, but rich summary.

Usage

```
Desc(x, ..., wrd = NULL)
## Default S3 method:
Desc(x, ...)
```

Arguments

x	the object to be described.
wrd	a pointer to a running Word instance. If this is NULL then output will be directed to console, else to Word.
...	the dots are passed to the specific function.

Details

More details can be found in the type-specific help texts.

Value

partly results are returned

Author(s)

Andri Signorell

See Also

See the specific object methods:
Desc.logical, Desc.factor, Desc.ordered, Desc.integer, Desc.numeric, Desc.Date, Desc.table, Desc.data.frame, Desc.list, Desc.formula, DescFactFact, DescNumFact, DescNumNum

Examples

```
data(d.pizza)

# implemented classes:
Desc(d.pizza$wrongpizza)     # logical
Desc(d.pizza$driver)         # factor
Desc(d.pizza$quality)        # ordered factor
Desc(d.pizza$week)           # integer
Desc(d.pizza$delivery_min)   # numeric
Desc(d.pizza$date)           # Date
```

```
Desc(d.pizza$wrongpizza, xname="The wrong pizza delivered", digits=5)


# just selected bivariate analysis on the console
Desc( price ~ operator, data=d.pizza)                  # numeric ~ factor
Desc( driver ~ operator, data=d.pizza)                 # factor ~ factor
Desc( driver ~ area + operator, data=d.pizza)          # factor ~ several factors
Desc( driver + area ~ operator, data=d.pizza)          # several factors ~ factor
Desc( driver ~ week, data=d.pizza )                   # factor ~ integer will be changed
                                                  #   into:   week (int) ~ driver (fact)

Desc( driver ~ operator, data=d.pizza, rfrq=("111"))   # alle rel. frequencies
# Desc( driver ~ operator, data=d.pizza
#  , rfrq=("000"), show.mutinf=TRUE)               # no rel. frequencies, but mutual information

Desc( price ~ delivery_min, data=d.pizza )             # numeric ~ numeric
Desc( price + delivery_min ~ operator + driver + wrongpizza
  , data=d.pizza, digits=c(2,2,2,2,0,3,0,0) )

Desc( week ~ driver, data=d.pizza, digits=c(2,2,2,2,0,3,0,0) )   # define digits

Desc( delivery_min + weekday ~ driver, data=d.pizza )


# without defining data-parameter
Desc( d.pizza$delivery_min ~ d.pizza$driver)
# Desc( d.pizza$delivery_min + d.pizza$operator ~ d.pizza$driver, show.mutinf=TRUE)


# with functions and interactions
Desc( sqrt(price) ~ operator : factor(wrongpizza), data=d.pizza)
Desc( log(price+1) ~ cut(delivery_min, breaks=seq(10,90,10))
  , data=d.pizza, digits=c(2,2,2,2,0,3,0,0))


# internal functions (not meant to be used by the enduser):
Desc.factor( d.pizza$driver, ord="n" )  # ordered by name
Desc.factor( d.pizza$driver, ord="l" )  # ordered by level
Desc.logical(d.pizza$wrongpizza)
Desc.integer( d.pizza$weekday)
Desc.integer( d.pizza$weekday, maxlevels=3 )
Desc.numeric( d.pizza$count, highlow=FALSE )
Desc.numeric( d.pizza$count, highlow=TRUE )

DescNumFact( x=d.pizza$delivery_min, grp=d.pizza$operator )
DescFactFact( x=d.pizza$driver, grp=d.pizza$operator)
DescNumNum( x=d.pizza$delivery_min, y=d.pizza$price )
```

`Desc.data.frame` *Describe a data.frame, a list or a matrix*

Description

Describes all the columns of a data.frame or a matrix, resp. all elements of a list, according to their class.

Usage

```
## S3 method for class 'data.frame'
Desc(x, sep = paste(rep("-",
                (as.numeric(options("width")) - 2)), collapse = ""), ...)
```

Arguments

x	the data.frame, the matrix or the list to be described
sep	character. The separator for the title.
...	the dots are passed to the child functions.

Details

See detailed information in the description of the according object interfaces.

Value

No results are returned.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Desc, PlotDesc

Examples

```
Desc(d.pizza[,c("temperature","count","driver","date","wine_delivered")] )
```

Desc.Date *Describe a Date Vector*

Description

Description interface for dates. We do here what seems resonable for describing dates. We start with a short summary about length, number of NAs and extreme values, before we describe the frequencies of the weekdays and months, rounded up by a chi-square test.

Usage

```
## S3 method for class 'Date'
Desc(x, xname = NULL, maxrows = 10,
          digits = 3, plotit = FALSE, ...)
```

Arguments

x	the Date vector to be described.
xname	the caption of the output.
maxrows	numeric. Defines the maximum number of rows to be reported. Default is 10 (most frequent ones). If maxrows < 1 then just as many rows, as the maxrows% most frequent factors are shown. Say if maxrows is set to 0.8 then as many rows are shown, that the highest cumulative relative frequency is the first going beyond 0.8.
digits	integer. With how many digits shoud the relative frequencies be formatted? Default is 3.
plotit	boolean. Should a plot be created? The factor is plotted with the factor interface of PlotDesc. Default is FALSE.
...	further argument to be passed to methods.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Desc, PlotDesc

Examples

```
Desc(d.pizza$date)
```

Desc.factor *Describe a factor, an ordered factor or a character vector*

Description

This reports a rich description of a factor, consisting out of length, number of NAs, number of levels and detailed frequencies of all levels. The order of the frequency table can be chosen between descending, ascending frequency, labels or levels. For ordered factors the order default is "level". Character vectors are treated as unordered factors.

Usage

```
## S3 method for class 'factor'
Desc(x, xname = NULL, ord = c("desc", "asc", "name", "level"),
     maxrows = 12, digits = 3, plotit = FALSE, ...)

## S3 method for class 'ordered'
Desc(x, xname = NULL, ...)

## S3 method for class 'character'
Desc(x, xname = NULL, ...)
```

Arguments

x	a single factor, an ordered factor or a character vector to be described.
xname	the caption of the output. If this is set to NULL (which is the default) the name of the factor and its class will be printed. Use NA if no caption should be printed at all.
ord	the order for the frequency table. Factors (and character vectors) are by default orderd by their descending frequencies, ordered factors by their natural order.
maxrows	numeric. Defines the maximum number of rows to be reported. For factors with lots of levels it is often not interesting to see all the levels. Default is 12 (most frequent ones). If maxrows < 1 then just as many rows, as the maxrows% most frequent factors are shown. Say if maxrows is set to 0.8 then as many rows are shown, that the highest cumulative relative frequency is the first going beyond 0.8.
digits	integer. With how many digits shoud the relative frequencies be formatted? Default is 3.
plotit	boolean. Should a plot be created? The factor is plotted with PlotDesc.factor. Default is FALSE.
...	further argument to be passed to methods. For ordered factors and character vectors they are passed to Desc.factor.

Details

Desc.char converts x to a factor an processes x as factor.

Desc.ordered only changes the standard order for the frequencies to it's intrinsic order, which means order "level" instead of "desc" in the factor case.

Value

A list containing the following components:

length	the length of the vector
n	the valid entries (NAs are excluded)
NAs	number of NAs
levels	number of levels
unique	number of unique values. Note that need not be the same, as there might be empty levels. Of course unique values can never be less than the number of levels.
dupes	boolean saying whether there are any duplicates in the vector.
frq	a data.frame of absolute and relative frequencies given by Freq

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Desc, PlotDesc

Examples

```
# unordered factor
Desc(d.diamonds$colour)

# ordered factor
Desc(d.diamonds$clarity)

# just the 5 first groups of the factor
Desc(d.diamonds$colour, maxrows = 5)

# just as much rows, as the most frequent 80% of the factors use
Desc(d.diamonds$colour, maxrows = 0.8)
```

Desc.formula *Describe Variables by Groups*

Description

Formula interface for describing data by groups.

Usage

```
## S3 method for class 'formula'
Desc(formula, data = parent.frame(), subset = TRUE, ...)
```

Arguments

formula	a formula of the form lhs ~ rhs where lhs gives the data values and rhs the corresponding groups.
data	an optional matrix or 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 argument to be passed to methods.

Details

The formula interface accepts the formula operators +, :, *, I(), 1 and evaluates any function. The left hand side and right hand side of the formula are evaluated the same way. The variable pairs are processed in dependency of their classes by the functions DescFactFact, DescNumFact and DescNumNum.

Value

just printed summary, no value returned

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Desc.logical, Desc.factor, Desc.ordered, Desc.integer, Desc.numeric, Desc.Date, Desc.table, Desc.data.frame, Desc.list

Examples

```
# univariate evaluation of temperature and driver
Desc(temperature + driver ~ 1, data=d.pizza, digits=1)

# temperature by driver
Desc(temperature ~ driver, data=d.pizza, digits=1)

# functions are evaluated
Desc(I(temperature^2) + sqrt(temperature) ~ interaction(driver, area), data=d.pizza, digits=1)
```

Desc.integer *Describe an integer variable*

Description

Describing an integer, means typically count data, is sometimes the same as describing an ordered factor, and somethimes, when there are many levels, it is like describing a numeric value.

Usage

```
## S3 method for class 'integer'
Desc(x, xname = NULL, maxrows = 12, freq = NULL,
             digits = 3, plotit = FALSE, ...)
```

Arguments

x	a single integer vector to be described.
xname	the caption for the output.
maxrows	integer. This indicates, up to how many levels the detailed frequencies should be reported. Default is 12. Set maxlevels to NA, if no restriction is to be applied, say the frequency table should contain all existing levels.
freq	logical, indicating if a frequency table should be plotted. Default is NULL, which means a frequency table will be printed, if there are less than 13 unique values. Else there will be a high-low description produced by HighLow.
digits	integer. With how many digits shoud the relative frequencies be formatted? Default is 3.
plotit	boolean. Should a plot be created? The vector would be plotted by means of PlotDesc.numeric, which again basically is PlotFdist. Default is FALSE.
...	further argument to be passed to methods.

Details

A horizontal barplot would be suitable as well here.

Value

A list containing the following components:

length	the length of the vector (n + NAs).
n	the valid entries (NAs are excluded)
NAs	number of NAs
unique	number of unique values.
0s	number of zeros
mean	arithmetic mean
MeanSE	standard error of the mean, as calculated by MeanSE.
quant	a table of quantiles, as calculated by quantile(x, probs = c(.05,.10,.25,.5,.75,.9,.95), na.
sd	standard deviation
vcoef	coefficient of variation: mean(x) / sd(x)
mad	median absolute deviation (mad)
IQR	interquartile range
skew	skewness, as calculated by Skew.
kurt	kurtosis, as calculated by Kurt.
highlow	the lowest and the highest values, reported with their frequencies in brackets, if > 1.
frq	a data.frame of absolute and relative frequencies given by Freq if maxlevels > unique values in the vector.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Desc.factor

Examples

```
# default
Desc(d.pizza$count)

# with frequency table
Desc(d.pizza$count, maxlevels=15)

# Result object
res <- Desc(d.pizza$count, maxlevels=15)
res
```

Desc.logical *Describe a dichotomous variable*

Description

Description of a dichotomous variable. This can either be a boolean vector, a factor with two levels or a numeric variable with only two unique values.

Usage

```
## S3 method for class 'logical'
Desc(x, xname = NULL, digits = 3,
     conf.level = 0.95, plotit = FALSE, ...)
```

Arguments

x	the dichotomous vector to be described.
xname	the caption for the output.
digits	integer. With how many digits shoud the relative frequencies be formatted? Default is 3.
conf.level	confidence level of the interval.
plotit	boolean. Should a plot be created? The plot looks like a horizontal bar plot. Default is FALSE.
...	the dots are passed to the table command within Desc.logical.

Details

The confidence levels for the relative frequencies are calculated by BinomCI, method "Wilson" on a confidence level defined by conf.level.

Value

A list containing the following components:

length	the length of the vector (n + NAs).
n	the valid entries (NAs are excluded)
NAs	number of NAs
unique	number of unique values.
frq	a data.frame of absolute and relative frequencies given by Freq if maxlevels > unique values in the vector.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Desc.factor, Desc.integer, Desc.numeric, Desc.data.frame

Examples

```
Desc(d.pizza$wine_delivered)
```

`Desc.numeric` *Describe a numeric vector*

Description

This reports a rich description of a numeric vector, consisting of the most common descriptive measures for location and variability.

Usage

```
## S3 method for class 'numeric'
Desc(x, xname = NULL, highlow = TRUE, plotit = FALSE, ...)
```

Arguments

`x`	a single numeric vector to be described.
`xname`	the caption for the output.
`highlow`	boolean. Should the highest and the lowest values be reported. This is usually a good idea and so the default is `TRUE`.
`plotit`	boolean. Should a plot be created? The vector would be plotted by means of `PlotDesc.numeric`, which again basically is `PlotFdist`. Default is `FALSE`.
`...`	further argument to be passed to methods.

Details

The plot function used here is `PlotFdist`.

Value

A list containing the following components:

`length`	the length of the vector (n + NAs).
`n`	the valid entries (NAs are excluded)
`NAs`	number of NAs
`unique`	number of unique values.
`0s`	number of zeros
`mean`	arithmetic mean
`meanSE`	standard error of the mean, as calculated by `MeanSE`.
`quant`	a table of quantiles, as calculated by `quantile` with probs set to c(.05, .10, .25, .5, .75, .9, .95).
`sd`	standard deviation
`vcoef`	coefficient of variation: mean(x) / sd(x)

mad	median absolute deviation (mad)
IQR	interquartile range
skew	skewness, as calculated by Skew.
kurt	kurtosis, as calculated by Kurt.
highlow	the lowest and the highest values, reported with their frequencies in brackets, if > 1.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Desc, PlotDesc

Examples

```
Desc(d.pizza$temperature)

# contains all results of:
quantile(d.pizza$temperature, na.rm=TRUE)
```

Desc.table *Describe a contingency table*

Description

A table will be described with it's relative frequencies, a short summary containing the total cases, the dimensions of the table, chi-square tests and phi-coefficient, contingency coefficient and Cramer's V as association measures.

Usage

```
## S3 method for class 'table'
Desc(x, xname = NULL, rfrq = NULL,
          margins = c(1, 2), plotit = FALSE, verb = c("med","lo","hi"), ...)
## S3 method for class 'matrix'
Desc(x, xname = NULL, rfrq = NULL,
           margins = c(1,2), plotit = FALSE, verb = c("med","lo","hi"), ...)
```

Arguments

x	a 2-dimensional table or a matrix
xname	the caption for the output.

rfrq	a string with 3 characters, each of them being 1 or 0. The first position means total percentages, the second means row percentages and the third column percentages. "011" produces a table output with row and column percentages. If set to NULL rfrq is defined in dependency of verb (verb = "lo" sets rfrq to "000" and else to "111", latter meaning all percentages will be reported.)
margins	a vector, consisting out of 1 and/or 2. Defines the margin sums to be included. Row margins are reported if margins is set to 1. Set it to 2 for column margins and c(1,2) for both. Default is NULL (none).
plotit	logical. Should a plot be created? Default is FALSE. The table would be plotted by PlotDesc.table.
verb	character defining the verbosity of the reported results. One out of c("med","lo","hi"), "med" being the default.
...	further argument to be passed to methods.

Details

Note that NAs cannot be handled by this interface, as tables in general come in "as.is", say basically as a matrix without any further information about potentially cleared NAs.

Value

no results are returned.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Desc.logical, Desc.factor, Desc.ordered, Desc.integer, Desc.numeric, Desc.Date, Desc.table, Desc.data.frame, Desc.formula

Examples

```
Desc(table(d.pizza$driver, Weekday(d.pizza$date)), rfrq="100", plotit=TRUE)


tab <- t(rbind(c(549,212,54),
          c(93,124,54),
          c(233,78,33),
          c(119,42,13),
          c(225,41,46),
          c(455,12,7),
          c(402,132,153)
    ))

# taciturn
Desc(tab, verb="lo")

# talkative
Desc(tab, verb="hi", expected=TRUE, res=TRUE)
```

DescWrd *Use Word as Reporting Tool for Describing Data*

Description

Calculates descriptive statistics for x and uses Word as reporting tool.

Usage

```
## Default S3 method:
DescWrd(x, wrd, caption = deparse(substitute(x)), ...)
## S3 method for class 'data.frame'
DescWrd(x, wrd, caption = deparse(substitute(x)),
                univar = TRUE, colpairs = FALSE, notch = TRUE,
                col0 = "#9A0941FF", col1 = "#8296C4FF", width = 100,
                fontname = "Consolas", fontsize = 7, ...)
## S3 method for class 'formula'
DescWrd(formula, data = parent.frame(), wrd, ...)
```

Arguments

x	the object to be described.
caption	the title of the decription. Default is name and class of the variable.
wrd	the pointer to a word instance. Can be a new one, created by GetNewWrd() or an existing one, created by GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").
formula	a formula of the form lhs ~ rhs where lhs gives the data values and rhs the corresponding groups.
data	an optional matrix or data frame containing the variables in the formula formula. By default the variables are taken from environment(formula).
univar	logical. Specifying if only univariate descriptions should be produced. Defaults to TRUE.
colpairs	logical. Specifying if pairwise bivariate descriptions should be produced. Defaults to FALSE.
notch	if notch is TRUE, a notch is drawn in each side of the boxes in a boxplot. See boxplot for the details.
col0	the first color used in mosaicplots.
col1	the last color used in mosaicplots.
width	integer. The linewidth of the results in characters. Defaults to 80.
fontname	the font of the text.
fontsize	the fontsize of the text.
...	further argument to be passed to methods.

Details

This function is not thought of being directly run by the enduser. It will normally be called automatically, when a pointer to a Word instance is passed to the function Desc.
However DescWrd takes some more specific arguments concerning the Word output (like font or fontsize), which can make it necessary to call the function directly.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Desc

Examples

```
# Output into word document ------------------------------------------------------

## Not run: # Windows-specific example

wrd <- GetNewWrd(header=TRUE)   # create a new word instance and insert title and contents

# let's have a subset
d.sub <- d.pizza[,c("driver","date","operator","price","wrongpizza")]

# do just the univariate analysis
Desc(d.sub, wrd=wrd)

# do a full report
Desc(d.sub, wrd=wrd, colpairs=TRUE)

# do just bivariate analysis
Desc( d.sub, univar=FALSE, colpairs=TRUE, wrd=wrd)

# selected bivariate analysis into word document
Desc(week ~ driver, data=d.pizza, wrd=wrd)
Desc(price ~ operator, data=d.pizza, digits=c(2,2,2,2,0,3,0,0), wrd=wrd )
Desc(driver ~ operator, data=d.pizza, wrd=wrd)
Desc(price ~ operator + driver + wrongpizza, data=d.pizza
     , digits=c(2,2,2,2,0,3,0,0), wrd=wrd)

Desc(price ~ delivery_min, data=d.pizza, wrd=wrd )


# internal functions (not meant to be used by the enduser):
Desc.factor(d.pizza$driver, ord="n" )  # ordered by name
Desc.factor(d.pizza$driver, ord="l" )  # ordered by level
Desc.logical(d.pizza$wrongpizza)
Desc.integer(d.pizza$weekday, maxlevels=NA)
Desc.integer(d.pizza$weekday, maxlevels=3)
Desc.numeric(d.pizza$count, highlow=FALSE)
Desc.numeric(d.pizza$count, highlow=TRUE)

DescNumFact( x=d.pizza$delivery_min, grp=d.pizza$operator )
DescFactFact( x=d.pizza$driver, grp=d.pizza$operator)
```

```
DescNumNum( x=d.pizza$delivery_min, y=d.pizza$price )


## End(Not run)
```

DivCoef *Rao's diversity coefficient also called quadratic entropy*

Description

Calculates Rao's diversity coefficient within samples.

Usage

```
DivCoef(df, dis, scale)
```

Arguments

df	a data frame with elements as rows, samples as columns, and abundance, presence-absence or frequencies as entries
dis	an object of class dist containing distances or dissimilarities among elements. If dis is NULL, Gini-Simpson index is performed.
scale	a logical value indicating whether or not the diversity coefficient should be scaled by its maximal value over all frequency distributions.

Value

Returns a data frame with samples as rows and the diversity coefficient within samples as columns

Note

This function was previously published as divc() in the **ade4** package and has been integrated here without logical changes.

Author(s)

Sandrine Pavoine <pavoine@biomserv.univ-lyon1.fr>

References

Rao, C.R. (1982) Diversity and dissimilarity coefficients: a unified approach. *Theoretical Population Biology*, **21**, 24–43.

Gini, C. (1912) Variabilita e mutabilita. *Universite di Cagliari III*, Parte II.

Simpson, E.H. (1949) Measurement of diversity. *Nature*, **163**, 688.

Champely, S. and Chessel, D. (2002) Measuring biological diversity using Euclidean metrics. *Environmental and Ecological Statistics*, **9**, 167–177.

Examples

```
# data(ecomor)
# dtaxo <- dist.taxo(ecomor$taxo)
# DivCoef(ecomor$habitat, dtaxo)

# data(humDNAm)
# DivCoef(humDNAm$samples, sqrt(humDNAm$distances))
```

`DivCoefMax` *Maximal value of Rao's diversity coefficient also called quadratic entropy*

Description

For a given dissimilarity matrix, this function calculates the maximal value of Rao's diversity coefficient over all frequency distribution. It uses an optimization technique based on Rosen's projection gradient algorithm and is verified using the Kuhn-Tucker conditions.

Usage

```
DivCoefMax(dis, epsilon, comment)
```

Arguments

`dis`	an object of class `dist` containing distances or dissimilarities among elements.
`epsilon`	a tolerance threshold : a frequency is non null if it is higher than epsilon.
`comment`	a logical value indicating whether or not comments on the optimization technique should be printed.

Value

Returns a list

`value`	the maximal value of Rao's diversity coefficient.
`vectors`	a data frame containing four frequency distributions : sim is a simple distribution which is equal to $\frac{D1}{1^t D1}$, pro is equal to $\frac{z}{1^t z1}$, where z is the nonnegative eigenvector of the matrix containing the squared dissimilarities among the elements, met is equal to z^2, num is a frequency vector maximizing Rao's diversity coefficient.

Author(s)

Stéphane Champely <Stephane.Champely@univ-lyon1.fr>
Sandrine Pavoine <pavoine@biomserv.univ-lyon1.fr>

References

Rao, C.R. (1982) Diversity and dissimilarity coefficients: a unified approach. *Theoretical Population Biology*, **21**, 24–43.

Gini, C. (1912) Variabilita e mutabilita. *Universite di Cagliari III*, Parte II.

Simpson, E.H. (1949) Measurement of diversity. *Nature*, **163**, 688.

Champely, S. and Chessel, D. (2002) Measuring biological diversity using Euclidean metrics. *Environmental and Ecological Statistics*, **9**, 167–177.

Pavoine, S., Ollier, S. and Pontier, D. (2005) Measuring diversity from dissimilarities with Rao's quadratic entropy: are any dissimilarities suitable? *Theoretical Population Biology*, **67**, 231–239.

Examples

```
## Not run:
par.safe <- par()$mar
data(elec88)
par(mar = c(0.1, 0.1, 0.1, 0.1))
# Departments of France.
area.plot(elec88$area)

# Dissimilarity matrix.
d0 <- dist(elec88$xy)

# Frequency distribution maximizing spatial diversity in France
# according to Rao's quadratic entropy.
France.m <- DivCoefMax(d0)
w0 <- France.m$vectors$num
v0 <- France.m$value
(1:94) [w0 > 0]

# Smallest circle including all the 94 departments.
# The squared radius of that circle is the maximal value of the
# spatial diversity.
w1 = elec88$xy[c(6, 28, 66), ]
w.c = apply(w1 * w0[c(6, 28, 66)], 2, sum)
symbols(w.c[1], w.c[2], circles = sqrt(v0), inc = FALSE, add = TRUE)
s.value(elec88$xy, w0, add.plot = TRUE)
par(mar = par.safe)


# Maximisation of Rao's diversity coefficient
# with ultrametric dissimilarities.
data(microsatt)
mic.genet <- count2genet(microsatt$tab)
mic.dist <- dist.genet(mic.genet, 1)
mic.phylog <- hclust2phylog(hclust(mic.dist))
plot.phylog(mic.phylog)
mic.maxpond <- DivCoefMax(mic.phylog$Wdist)$vectors$num
dotchart.phylog(mic.phylog, mic.maxpond)

## End(Not run)
```

DrawAnnulus *Draw One or Several Annuli*

Description

Draw an annulus (or a sequence of annuli) with given center coordinates, fill and border colors on an existing plot using classical graphics.

Usage

```
DrawAnnulus(x = 0, y = x, radius.in = 1, radius.out = 2, nv = 100,
            border = par("fg"), col = par("bg"), lty = par("lty"),
            lwd = par("lwd"), plot = TRUE)
```

Arguments

x, y	a vector (or scalar) of xy-coordinates of the center(s).
radius.in	a vector (or scalar) of the inner radius of the annulus(i).
radius.out	a vector (or scalar) of the outer radius of the annulus(i).
nv	number of vertices to draw the circles. Default is 100.
border	color for the border(s). The default is par("fg"). Use border = NA to omit borders. If there are shading lines, border = TRUE means use the same colour for the border as for the shading lines.
col	color(s) to fill or shade the annulus with. The default NA (or also NULL) means do not fill, i.e., draw transparent rectangles, unless density is specified.
lty	line type for borders and shading; defaults to "solid".
lwd	line width for borders and shading.
plot	logical. If TRUE the structure will be plotted. If FALSE only the xy-points are calculated and returned. Use this option if you want to combine several geometric structures to a single polygon.

Details

All geometric arguments are recycled if necessary.

Value

DrawAnnulus invisibly returns a list of the calculated coordinates for all shapes.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

polygon, DrawRegPolygon, DrawCircle, DrawArc

Examples

```
Canvas(0.5)
DrawRegPolygon(nv=4, rot=pi/4, col="lightblue")
DrawAnnulus(radius.in=0.3, radius.out=0.45, col="lightgrey", border="darkgrey", lwd=5)
```

DrawAnnulusSector *Draw a Sector of an Annulus*

Description

Draw one or more annulus sectors with given centers, radii, angles, fill- and border colors on an existing plot using classical graphics.

Usage

```
DrawAnnulusSector(x = 0, y = x, radius.in = 1, radius.out = 2,
                  angle.beg = 0, angle.end = pi, nv = 100,
                  border = par("fg"), col = par("bg"), lty = par("lty"),
                  lwd = par("lwd"), plot = TRUE)
```

Arguments

x, y	a vector (or scalar) of xy-coordinates of the center(s).
radius.in	a vector (or scalar) of the inner radius of the annulus(i).
radius.out	a vector (or scalar) of the outer radius of the annulus(i).
angle.beg	a vector (or scalar) of the starting angle(s). The sectors are built counterclockwise.
angle.end	a vector (or scalar) of the ending angle(s).
nv	number of vertices to draw the arcs.
border	color for borders. The default is par("fg"). Use border = NA to omit borders. If there are shading lines, border = TRUE means use the same colour for the border as for the shading lines.
col	color(s) to fill or shade the annulus sector with. The default NA (or also NULL) means do not fill (say draw transparent).
lty	line type for borders and shading; defaults to "solid".
lwd	line width for borders and shading.
plot	logical. If TRUE the structure will be plotted. If FALSE only the points are calculated and returned. Use this if you want to combine several geometric structures to a single polygon.

Value

DrawAnnulusSector invisibly returns a list of the calculated coordinates for all shapes.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

polygon, DrawAnnulus, DrawRegPolygon, DrawCircle, DrawArc

Examples

```
par(mfrow=c(1,2))

angles <- seq( 0,2 * pi, pi/4)  # the angles
mycol <- rainbow(8)             # colors of the sector annuli
d <- 0.1                        # the gap between the sectors in radians

plot(1:10, type="n", asp=1, xlab="", ylab="")
res <- sapply( 1:(length(angles)-1),
  function(i) DrawAnnulusSector(x = 6, y = 6, radius.in = 2, radius.out = 3,
              angle.beg = angles[i] + d/2, angle.end = angles[i+1] - d/2, col = mycol[i])
)


# Produce a clockplot
x <- c(15,9,75,90,1,1,11,5,9,8,33,11,11,20,14,13,10,28,33,21,24,25,11,33)
# plot clockwise, starting from 12 o'clock
angles <- (rev(seq(0,2*pi, pi/12) + pi/2))

Canvas(xlim=c(-100,100), main="Number of visitors to web site for each hour of a day")
PolarGrid(nr=c(0,90), ntheta=24, rlabels=NA, alabels=c(6:0, 23:7) )
DrawAnnulusSector(radius.in=0, radius.out=x, angle.beg = angles[-1],
  angle.end = angles[-length(angles)], col=rainbow(24))
```

DrawArc *Draw Elliptic or Circular Arc(s)*

Description

Draw one or more elliptic or circular arcs from angle.beg to angle.end on an existing plot using classic graphics.

Usage

```
DrawArc(x = 0, y = x, radius.x = 1, radius.y = radius.x,
        angle.beg = 0, angle.end = pi, nv = 100,
        col = par("col"), lty = par("lty"), lwd = par("lwd"),
        plot = TRUE)
```

Arguments

x, y	a vector (or scalar) of xy-coordinates of the center(s) of the arc(s).
radius.x	a scalar or a vector giving the semi-major axis of the ellipse for the arc(s)
radius.y	a scalar or a vector giving the semi-minor axis of the ellipse for the arc(s). Default is radius.x which will result in a circle arc with radius.x.

angle.beg	a scalar or a vector of starting angles in radians.
angle.end	a scalar or a vector of ending angles in radians.
nv	number of vertices used to plot the arc. Scalar or vector.
col	color for the arc(s). Scalar or vector.
lty	line type used for drawing.
lwd	line width used for drawing.
plot	logical. If TRUE the structure will be plotted. If FALSE only the xy-points are calculated and returned. Use this if you want to combine several geometric structures to a single polygon.

Details

All parameters are recycled if necessary.
Be sure to use an aspect ratio of 1 as shown in the example to avoid distortion.

Value

DrawArc invisibly returns a list of the calculated coordinates for all shapes.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

DrawCircle, DrawAnnulusSector, polygon

Examples

```
curve(sin(x), 0, pi, col="blue", asp=1)
DrawArc( x = pi/2, y = 0, radius.x = 1, angle.beg = pi/4, angle.end = 3*pi/4, col="red")
```

DrawBand *Draw Confidence Band*

Description

Draw a (confidence) band. Just a wrapper for polygon.

Usage

```
DrawBand(x, y, col = SetAlpha("grey", 0.5), border = NA)
```

Arguments

x	a vector with x coordinates for the band.
y	a vector with y coordinates for the band.
col	the color of the band.
border	the border color of the band.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

polygon

Examples

```
set.seed(18)
x <- rnorm(15)
y <- x + rnorm(15)

new <- seq(-3, 3, 0.5)
pred.w.plim <- predict(lm(y ~ x), newdata=data.frame(x=new), interval="prediction")
pred.w.clim <- predict(lm(y ~ x), newdata=data.frame(x=new), interval="confidence")

plot(y ~ x)
DrawBand(y = c(pred.w.plim[,2], rev(pred.w.plim[,3])),
  x=c(new, rev(new)), col= SetAlpha("grey90", 0.5))
DrawBand(y = c(pred.w.clim[,2], rev(pred.w.clim[,3])),
  x=c(new, rev(new)), col= SetAlpha("grey80", 0.5))

abline(lm(y~x), col="brown")
```

DrawBezier *Draw a Bezier Curve*

Description

Draw a Bezier curve.

Usage

```
DrawBezier(x = 0, y = x, nv = 100, col = par("col"), lty = par("lty"),
           lwd = par("lwd"), plot = TRUE)
```

Arguments

x, y	a vector of xy-coordinates to define the Bezier curve. Should at least contain 3 points.
nv	number of vertices to draw the curve.
col	color(s) for the curve. Default is par("fg").
lty	line type for borders and shading; defaults to "solid".
lwd	line width for borders and shading.
plot	logical. If TRUE the structure will be plotted. If FALSE only the xy-points are calculated and returned. Use this if you want to combine several geometric structures to a single polygon.

Details

Bezier curves appear in such areas as mechanical computer aided design (CAD). They are named after P. Bezier, who used a closely related representation in Renault's UNISURF CAD system in the early 1960s (similar, unpublished, work was done by P. de Casteljau at Citroen in the late 1950s and early 1960s). The 1970s and 1980s saw a flowering of interest in Bezier curves, with many CAD systems using them, and many important developments in their theory. The usefulness of Bezier curves resides in their many geometric and analytical properties. There are elegant and efficient algorithms for evaluation, differentiation, subdivision of the curves, and conversion to other useful representations. (See: Farin, 1993)

Value

`DrawBezier` invisibly returns a list of the calculated coordinates for all shapes.

Author(s)

Frank E Harrell Jr <f.harrell@vanderbilt.edu>

References

G. Farin (1993) *Curves and surfaces for computer aided geometric design. A practical guide*, Acad. Press

See Also

`polygon`, `DrawRegPolygon`, `DrawCircle`, `DrawArc`

Examples

```
Canvas(xlim=c(0,1))
grid()
DrawBezier( x=c(0,0.5,1), y=c(0,0.5,0), col="blue", lwd=2)
DrawBezier( x=c(0,0.5,1), y=c(0,1,0), col="red", lwd=2)
DrawBezier( x=c(0,0.25,0.5,0.75,1), y=c(0,1,1,1,0), col="darkgreen", lwd=2)
```

DrawCircle *Draw a Circle*

Description

Draw one or several circle on an existing plot.

Usage

```
DrawCircle(x = 0, y = x, radius = 1, rot = 0, nv = 100,
           border = par("fg"), col = par("bg"), lty = par("lty"),
           lwd = par("lwd"), plot = TRUE)
```

Arguments

x, y	a vector (or scalar) of xy-coordinates for the center(s) of the circle(s).
radius	a scalar or a vector giving the radius of the circle(s)
rot	rotation angle for the geometric structure in radiants.
nv	number of vertices to draw the circle.
border	color for annulus borders. The default is par("fg"). Use border = NA to omit borders.
col	color(s) to fill or shade the circle(s) with. The default NA (or also NULL) means do not fill, i.e., draw transparent rectangles, unless density is specified.
lty	line type for borders and shading; defaults to "solid".
lwd	line width for borders and shading.
plot	logical. If TRUE the structure will be plotted. If FALSE only the points are calculated and returned. Use this option if you want to combine several geometric structures to a polygon.

Details

All geometric arguments will be recycled.

Value

The function invisibly returns a list of the calculated coordinates for all shapes.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

polygon, DrawRegPolygon, DrawEllipse, DrawArc, DrawAnnulus

Examples

```
Canvas(xlim=c(-5,5))
DrawCircle( radius=4:1, col=c("white","steelblue2","white","red"), lwd=3, nv=300)


x <- seq(-3,3, length.out=18)

par(bg="black")
plot( x=c(-5,5), y=c(-5,5), asp=1, type="n", xaxt="n", yaxt="n", xlab="", ylab="")

sapply( (0:12) * pi/6, function(theta) {
  xy <- Rotate( x, y=0, theta=theta )
  DrawCircle( x=xy$x, y=xy$y, radius=2.4, border="white", col="transparent" )
} )
```

DrawEllipse *Draw an Ellipse*

Description

Draw one or several ellipses on an existing plot.

Usage

```
DrawEllipse(x = 0, y = x, radius.x = 1, radius.y = 0.5, rot = 0,
            nv = 100, border = par("fg"), col = par("bg"),
            lty = par("lty"), lwd = par("lwd"), plot = TRUE)
```

Arguments

x, y	the x and y co-ordinates for the centre(s) of the ellipse(s).
radius.x	a scalar or a vector giving the semi-major axis of the ellipse.
radius.y	a scalar or a vector giving the semi-minor axis of the ellipse.
rot	angle of rotation in radians.
nv	number of vertices to draw the ellipses.
border	color for borders. The default is par("fg"). Use border = NA to omit borders.
col	color(s) to fill or shade the annulus sector with. The default NA (or also NULL) means do not fill (say draw transparent).
lty	line type for borders and shading; defaults to "solid".
lwd	line width for borders and shading.
plot	logical. If TRUE the structure will be plotted. If FALSE only the points are calculated and returned. Use this if you want to combine several geometric structures to a single polygon.

Details

Use DegToRad if you want to define rotation angle in degrees.

Value

The function invisibly returns a list of the calculated coordinates for all shapes.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

polygon, DrawRegPolygon, DrawCircle, DrawArc, DrawAnnulus

Examples

```
par(mfrow=c(1,2))

Canvas()
DrawEllipse(rot = c(1:3) * pi/3, col=SetAlpha(c("blue","red","green"), 0.5) )


plot(cars)
m <- var(cars)
eig <- eigen(m)
eig.val <- sqrt(eig$values)
eig.vec <- eig$vectors

DrawEllipse(x=mean(cars$speed), y=mean(cars$dist), radius.x=eig.val[1] , radius.y=eig.val[2]
  , rot=acos(eig.vec[1,1]), border="blue", lwd=3)
```

DrawRegPolygon *Draw Regular Polygon(s)*

Description

Draw a regular polygon with n corners. This is the workhorse function for drawing regular polygons. Drawing a circle can be done by setting the vertices to a value of say 100.

Usage

```
DrawRegPolygon(x = 0, y = x, radius.x = 1, radius.y = radius.x, rot = 0,
            nv = 3, border = par("fg"), col = par("bg"), lty = par("lty"),
            lwd = par("lwd"), plot = TRUE)
```

Arguments

x, y	a vector (or scalar) of xy-coordinates of the center(s) of the regular polygon(s).
radius.x	a scalar or a vector giving the semi-major axis of the ellipse for the polygon(s).
radius.y	a scalar or a vector giving the semi-minor axis of the ellipse for the polygon(s). Default is radius.x which will result in a polygon with radius.x.
rot	angle of rotation in radians.
nv	number of vertices to draw the polygon(s).
border	color for borders. The default is par("fg"). Use border = NA to omit borders.
col	color(s) to fill or shade the shape with. The default NA (or also NULL) means do not fill (say draw transparent).
lty	line type for borders and shading; defaults to "solid".
lwd	line width for borders and shading.
plot	logical. If TRUE the structure will be plotted. If FALSE only the points are calculated and returned. Use this if you want to combine several geometric structures to a polygon.

Details

All geometric arguments will be recycled.

Value

The function invisibly returns a list of the calculated coordinates for all shapes.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

polygon, DrawAnnulus, DrawCircle, DrawArc

Examples

```
# Draw 4 triangles (nv = 3) with different rotation angles
plot(c(0,1),c(0,1), asp=1, type="n", xaxt="n", yaxt="n", xlab="", ylab="")
DrawRegPolygon(x = 0.5, y = 0.5, rot = (1:4)*pi/6, radius.x = 0.5, nv = 3,
  col = SetAlpha("yellow",0.5))


# Draw several polygons
plot(c(0,1),c(0,1), asp=1, type="n", xaxt="n", yaxt="n", xlab="", ylab="")
DrawRegPolygon(x = 0.5, y = 0.5, radius.x=seq(50, 5, -10) * 1 /100,
  rot=0, nv = c(50, 10, 7, 4, 3), col=SetAlpha("blue",seq(0.2,0.7,0.1)))


# Combine several polygons by sorting the coordinates
# Calculate the xy-points for two concentric pentagons
d.pts <- do.call("rbind", lapply(DrawRegPolygon(radius.x=c(1,0.38), nv=5,
  rot=c(pi/2, pi/2+pi/5), plot=FALSE ), data.frame))

# prepare plot
plot(c(-1,1),c(-1,1), asp=1, type="n", xaxt="n", yaxt="n", xlab="", ylab="")

# .. and draw the polygon with reordered points
polygon( d.pts[order(rep(1:6, times=2), rep(1:2, each=6)), c("x","y")], col="yellow")



# Move the center
plot(c(0,1),c(0,1), asp=1, type="n", xaxt="n", yaxt="n", xlab="", ylab="")
theta <- seq(0, pi/6, length.out=5)
xy <- PolToCart( exp(theta) /2, theta)
DrawRegPolygon(x=xy$x, y=xy$y + 0.5, radius.x=seq(0.5, 0.1, -0.1),
  nv=4, rot=seq(0, pi/2, length.out=5), col=rainbow(5) )


# Plot a polygon with a "hole"
plot(c(-1,1),c(-1,1), asp=1, type="n", xaxt="n", yaxt="n", xlab="", ylab="")
DrawRegPolygon(nv = 4, rot=pi/4, col="red" )
text(x=0,y=0, "Polygon", cex=6, srt=45)
```

```
# Calculate circle and hexagon, but do not plot
pts <- DrawRegPolygon(radius.x=c(0.7, 0.5), nv = c(100, 6), plot=FALSE )

# combine the 2 shapes and plot the new structure
polygon(x = unlist(lapply(pts, "[", "x")),
  y=unlist(lapply(pts, "[", "y")), col="green", border=FALSE)
```

Dummy *Generate Dummy Codes for a Factor*

Description

Generates a matrix of dummy codes (class indicators) for a given factor.

Usage

```
Dummy(x, method = c("treatment", "sum", "helmert", "poly", "full"), base = 1)
```

Arguments

x	factor or vector of classes for cases.
method	defines the method of the contrasts being formed. Can be one out of "treatment", "sum", "helmert", "poly", "full", whereas "treatment" is the default one. Abbreviations are accepted. The option "full" returns a full set of class indicators, say a dummy factor for EACH level of x. Note that this would be redundant for lm and friends!
base	an integer specifying which group is considered the baseline group.

Value

a matrix with the dummy codes. The rows correspond to the number of elements in x and the columns to it's levels.

Author(s)

Andri Signorell <andri@signorell.net>

References

Venables, W N and Ripley, B D (2002): *Modern Applied Statistics with S*. Fourth edition. Springer.

See Also

model.frame, contrasts, class.ind in the package **nnet**

Examples

```
x <- c("red","blue","green","blue","green","red","red","blue")
Dummy(x)
Dummy(x, base = 2)

Dummy(x, method = "sum")


y <- c("Max","Max","Max","Max","Max","Bill","Bill","Bill")

Dummy(y)
Dummy(y, base = "Max")

Dummy(y, base = "Max", method="full")


# "Undummy" (revert the dummy coding)
m <- Dummy(y, method="full")
m
z <- apply(m, 1, function(x) colnames(m)[x==1])
z
identical(y, as.vector(z))

m <- Dummy(y)
m
z <- apply(m, 1, function(x) ifelse(sum(x)==0, attr(m,"base"), colnames(m)[x==1]))
z
```

`Entropy` *Shannon Entropy and Mutual Information*

Description

Computes Shannon entropy and the mutual information of two variables. The entropy quantifies the expected value of the information contained in a vector. The mutual information is a quantity that measures the mutual dependence of the two random variables.

Usage

```
Entropy(x, y = NULL, base = 2, ...)

MutInf(x, y, base = 2, ...)
```

Arguments

`x`	a vector or a matrix of numerical or categorical type. If only x is supplied it will be interpreted as contingency table.
`y`	a vector with the same type and dimension as x. If y is not NULL then the entropy of `table(x, y, ...)` will be calculated.
`base`	base of the logarithm to be used, defaults to 2.
`...`	further arguments are passed to the function `table`, allowing i.e. to set `useNA`.

Details

The Shannon entropy equation provides a way to estimate the average minimum number of bits needed to encode a string of symbols, based on the frequency of the symbols.
It is given by the formula H = - sum(pi log(pi)) where pi is the probability of character number i showing up in a stream of characters of the given "script".
The entropy is ranging from 0 to Inf.

Value

a numeric value.

Author(s)

Andri Signorell <andri@signorell.net>

References

Shannon, Claude E. (July/October 1948). A Mathematical Theory of Communication, *Bell System Technical Journal* 27 (3): 379-423.

Ihara, Shunsuke (1993) *Information theory for continuous systems*, World Scientific. p. 2. ISBN 978-981-02-0985-8.

See Also

package **entropy** which implements various estimators of entropy

Examples

```
Entropy(as.matrix(rep(1/8, 8)))

# http://r.789695.n4.nabble.com/entropy-package-how-to-compute-mutual-information-td4385339.html
x <- as.factor(c("a","b","a","c","b","c"))
y <- as.factor(c("b","a","a","c","c","b"))

Entropy(table(x), base=exp(1))
Entropy(table(y), base=exp(1))
Entropy(x, y, base=exp(1))

# Mutual information is
Entropy(table(x), base=exp(1)) + Entropy(table(y), base=exp(1)) - Entropy(x, y, base=exp(1))
MutInf(x, y, base=exp(1))

Entropy(table(x)) + Entropy(table(y)) - Entropy(x, y)
MutInf(x, y, base=2)

# http://en.wikipedia.org/wiki/Cluster_labeling
tab <- matrix(c(60,10000,200,500000), nrow=2, byrow=TRUE)
MutInf(tab, base=2)

d.frm <- Untable(as.table(tab))
str(d.frm)
MutInf(d.frm[,1], d.frm[,2])

table(d.frm[,1], d.frm[,2])
```

```
MutInf(table(d.frm[,1], d.frm[,2]))


# Ranking mutual information can help to describe clusters
#
#   r.mi <- MutInf(x, grp)
#   attributes(r.mi)$dimnames <- attributes(tab)$dimnames
#
#   # calculating ranks of mutual information
#   r.mi_r <- apply( -r.mi, 2, rank, na.last=TRUE )
#   # show only first 6 ranks
#   r.mi_r6 <- ifelse( r.mi_r < 7, r.mi_r, NA)
#   attributes(r.mi_r6)$dimnames <- attributes(tab)$dimnames
#   r.mi_r6
```

`ExpFreq` *Expected frequencies*

Description

Calculate the expected frequencies of an n-way table assuming independence.

Usage

```
ExpFreq(x, freq = c("abs", "rel"))
```

Arguments

`x`	a table.
`freq`	indicates, whether absolute or relative frenquencies should be computed. Can either be `"abs"` or `"rel"`. Partial matching is supported.

Value

A table with either the absolute or the relative expected frequencies.

Note

This is a copy of the function `independence_table` in **vcd**.

Author(s)

David Meyer <David.Meyer@R-project.org>

See Also

`chisq.test`

Examples

```
ExpFreq(Titanic)

ExpFreq(UCBAdmissions, freq="r")
```

Factorize *Prime Factorization of Integers*

Description

Compute the prime factorization(s) of integer(s) n.

Usage

```
Factorize(n)
```

Arguments

n	vector of integers to factorize.

Details

works via Primes, currently in a cheap way, sub-optimal for large composite n.

Value

A named list of the same length as n, each element a 2-column matrix with column "p" the prime factors and column~"m" their respective exponents (or multiplities), i.e., for a prime number n, the resulting matrix is cbind(p = n, m = 1).

Author(s)

Martin Maechler, Jan. 1996.

See Also

Primes.

For factorization of moderately or really large numbers, see the **gmp** package, and its factorize().

Examples

```
Factorize(47)
Factorize(seq(101, 120, by=2))
```

FctArgs *Retrieve a Functions' Arguments*

Description

Retrieve a functions' arguments and default values in a list.

Usage

```
FctArgs(name, sort = FALSE)
```

Arguments

name	name of the function.
sort	logical. Should the function arguments be sorted? Defaults to FALSE.

Value

a data.frame with the default in the first columns and with row.names as argument names.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

formalArgs just returns the name of the arguments, but not their defaults.

Examples

```
formalArgs(PlotFdist)

# compare:
FctArgs(PlotFdist)
```

Fibonacci *Fibonacci Series*

Description

Generates single Fibonacci numbers or a Fibonacci sequence.

Usage

```
Fibonacci(n, sequence = TRUE)
```

Arguments

n	integer.
sequence	logical; default is TRUE.

Details

Generates the n-th Fibonacci number, or the whole Fibonacci sequence from the first to the n-th number.

The recursive version (sequence=FALSE) is extremely slow for values n>=30. To get the n-th Fibonacci number for larger n values, use Fibonacci(n)[n].

Value

A single integer, or a vector of integers.

Note

This function stems from the library **numbers** (fibonacci).

Author(s)

Hans W Borchers <hwborchers at googlemail.com>

Examples

```
Fibonacci(0)                            # 1
Fibonacci(2)                            # 2
Fibonacci(2, sequence = TRUE)           # 1 2


# Golden ratio
F <- Fibonacci(25, sequence = TRUE)     # ... 75025 121393
f25 <- F[25]/F[24]                      #     1.618033989
phi <- (sqrt(5) + 1)/2
abs(f25 - phi)                          # 7.945178e-11

# Fibonacci numbers w/o iteration
  fibo <- function(n) {
    phi <- (sqrt(5) + 1)/2
    fib <- (phi^(n+1) - (1-phi)^(n+1)) / (2*phi - 1)
    round(fib)
  }
fibo(30:33)                             # 1346269 2178309 3524578 5702887

# Compare recursive with iterative approach:
# system.time(F30 <- Fibonacci(30, sequence = FALSE)      # user: 17.075 s
# system.time(F30 <- Fibonacci(30, sequence = TRUE)[30])  # user:  0.006 s
```

FindColor *Get Color on a Defined Color Range*

Description

Find a color on a defined color range depending on the value of x. This is helpful for colorcoding numeric values.

Usage

```
FindColor(x, cols = rev(heat.colors(100)),
          min.x = min(pretty(x)), max.x = max(pretty(x)),
          all.inside = FALSE)
```

Arguments

x	numeric.
cols	all the colors in defined range.
min.x	the x-value for the first color.
max.x	the x-value for the last color.
all.inside	logical; if true, the returned indices are coerced into 1, ..., N-1, i.e., 0 is mapped to 1 and N to N-1.

Details

For the selection of colors the option rightmost.closed in the used function findInterval is set to TRUE. This will ensure that all values on the right edge of the range are assigned a color. How values outside the boundaries of min.x and max.x are handled can be controlled by all.inside. Set this value to TRUE, if those values should get the colors at the edges or set it to FALSE, if they should remain white (which is the default).

Author(s)

Andri Signorell <andri@signorell.net>

See Also

findInterval

Examples

```
Canvas(7, main="Use of function FindColor()")

# get some data
x <- c(23,56,96)
# get a color range from blue via white to red
cols <- colorRampPalette(c("blue","white","red"))(100)
ColorLegend(x="bottomleft", cols=cols, xlab=rev(seq(0,100,10)), cex=0.8)
```

```
# and now the color coding of x:
xcols <- FindColor(x, cols, min.x=0, max.x=100 )

# how does it look like?
text(x=1, y=c(3), labels="Color coding of x:")
text(x=1.5, y=c(-5,-2,1), labels=x)
DrawRegPolygon(x=3, y=c(-5,-2,1), nv=4, rot=pi/4, col=xcols)
text(x=6, y=c(-5,-2,1), labels=xcols)
```

FisherZ *Fisher r to z and z to r and confidence intervals*

Description

Convert a correlation to a z score or z to r using the Fisher transformation or find the confidence intervals for a specified correlation.

Usage

```
FisherZ(rho)
FisherZInv(z)
CorCI(rho, n, conf.level = 0.95, twotailed = TRUE)
```

Arguments

rho	a Pearson r
z	A Fisher z
n	Sample size for confidence intervals
conf.level	Confidence interval
twotailed	Treat p as twotailed p

Value

z value corresponding to r (FisherZ) \ r corresponding to z (FisherZInv) \ rho, lower and upper confidence intervals (CorCI) \

Author(s)

William Revelle <revelle@northwestern.edu>

Examples

```
cors <- seq(-.9,.9,.1)

zs <- FisherZ(cors)
rs <- FisherZInv(zs)
round(zs, 2)
n <- 30
r <- seq(0,.9,.1)
```

```
rc <- t(sapply(r, CorCI, n=n))
t <- r*sqrt(n-2)/sqrt(1-r^2)
p <- (1-pt(t,n-2))/2

r.rc <- data.frame(r=r, z=FisherZ(r), lower=rc[,2], upper=rc[,3], t=t, p=p)

round(r.rc,2)
```

FixToTab *Text to Table*

Description

Convert a text to a table by using complete columns of spaces (or any other separator) as delimiting point.

Usage

```
FixToTab(txt, sep = " ", delim = "\t", trim = TRUE, header = TRUE)
```

Arguments

txt	the text to be partitioned. Works best, if txt is a matrix.
sep	the separator to use. Will frequently be " ".
delim	the new delimiter to insert. (default tab)
trim	logical. Should the separated text be trimmed from whitespace? Defaults to TRUE.
header	logical. Should the first line be interpreted as header?

Details

Only a complete appearance of the separator character in the same position over all rows will be accepted as column delimiter.

Value

a matrix of the separated text.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

StrChop

Examples

```
# let's get some tabbed text
txt <- matrix(capture.output(Titanic[,,2,1]))
FixToTab(txt[-1,])
```

`FormatFix` *Format to a fixed format representation*

Description

`FormatFix` formats to fixed point number format. It writes x with sign (" " or "-") and `"before"` decimals before the "." and with `"after"` decimals after the ".". If `"after"` == 0 then the "." will be omitted.

There will always be at least one decimal digit before the "."

If `"before"` is too small to represent `x` then if `extend == TRUE`, the string will be extended, else a string consisting of "*" of length `before + after` will be given.

If `abs(x) >= 10^8` values very near `10^k` cannot be represented exactly, so the normal `format` will be used.

Names are retained.

Usage

```
FormatFix(x, after, before = 2,extend = TRUE)
```

Arguments

`x`	the number to be represented.
`after`	the number of decimals after ".".
`before`	the minimum number of decimals before ".".
`extend`	extend string if necessary.

Value

The string representing the fixed point format of `x`.

Note

This function was previously published as `formatFix()` in the **cwhmisc** package and has been integrated here without logical changes.

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>

Examples

```
xxbig <- c(1.2e9,3.51e23,6.72e120,NaN)
xx    <- c(0.001,92,exp(1),1000*pi)
FormatFix(c(-rev(xxbig),-rev(xx),0,NA,xx,xxbig),0,3)
#> [1] "    NaN" "-7e+120" " -4e+23" " -1e+09" "  -3142" "     -3" "    -92"
#> [8] "     -0" "      0" "     NA" "      0" "     92" "      3" "   3142"
#> [15] "  1e+09" "  4e+23" " 7e+120" "    NaN"
FormatFix(c(-rev(xxbig),-rev(xx),0,NA,xx,xxbig),0,3,FALSE)
#> [1] "NaN" "***" "***" "***" "***" " -3" "-92" " -0" "  0" " NA" "  0" " 92"
#> [13] "  3" "***" "***" "***" "***" "NaN"
FormatFix(c(-rev(xxbig),-rev(xx),0,NA,xx,xxbig),6,3)
#> [1] "         NaN" "  -6.72e+120" "   -3.51e+23" "    -1.2e+09" "-3141.592654"
#> [6] "   -2.718282" "  -92.000000" "   -0.001000" "    0.000000" "          NA"
#> [11] "    0.001000" "   92.000000" "    2.718282" " 3141.592654" "     1.2e+09"
#> [16] "    3.51e+23" "   6.72e+120" "         NaN"
FormatFix(c(-rev(xxbig),-rev(xx),0,NA,xx,xxbig),6,3,FALSE)
#> [1] "       NaN" "-6.72e+120" " -3.51e+23" "  -1.2e+09" "**********"
#> [6] " -2.718282" "-92.000000" " -0.001000" "  0.000000" "        NA"
#> [11] "  0.001000" " 92.000000" "  2.718282" "**********" "   1.2e+09"
#> [16] "  3.51e+23" " 6.72e+120" "       NaN"
```

`FormatSig` *Significance Representation of a P-Value*

Description

Create a significance representation of a p-value consisting of * and . according to the used defaults e.g. in `lm`.

Usage

```
FormatSig(x, breaks = c(0, 0.001, 0.01, 0.05, 0.1, 1),
          labels = c("***", "**", "*", ".", " "))
```

Arguments

`x`	a numerical vector.
`breaks`	the breaks for the significance groups.
`labels`	the labels for the significance groups.

Value

The string representing the significance format of `x`.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`FormatFix`, `format`

Examples

```
FormatSig(x=c(0.12, 0.07,0.03, 0.008, 0.000001))
```

Frac *Return the Fractional Part of a Numeric Value*

Description

Return the fractional part of a numeric value.

Usage

```
Frac(x, dpwr = NA)
```

Arguments

x	the numeric value (or a vector of numerics), whose fractional part is to be calculated.
dpwr	if dpwr is not missing, the fractional part will be multiplied by 10^dpwr and returned rounded to integer. Defaults to NA.

Value

return the fractional part of x.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Ndec

Examples

```
x <- rnorm(5)*100
x
Frac(x)

# multiply by 10^4
Frac(x, dpwr=4)
```

`Freq` *Frequency Table*

Description

Calculates absolute and relative frequencies of a vector x. Continuous variables will be cut using the logic applied by the function `hist`. Categorical variables will be aggregated by `table`. The result will contain single and cumulative frequencies for both, absolute values and percentages.

Usage

```
Freq(x, breaks = hist(x, plot = FALSE)$breaks, include.lowest = TRUE,
     ord = c("level", "desc", "asc", "name"),
     useNA = c("no", "ifany", "always"), ...)

## S3 method for class 'Freq'
print(x, digits = 3, ...)
```

Arguments

`x`	the variable to be described, x can be numeric or a(n) (ordered) factor.
`breaks`	either a numeric vector of two or more cut points or a single number (greater than or equal to 2) giving the number of intervals into which x is to be cut. Default taken from the function `hist()`. This is ignored if x is a factor.
`include.lowest`	logical, indicating if an x[i] equal to the lowest (or highest, for `right = FALSE`) `"breaks"` value should be included.
`ord`	how should the result be ordered? Default is `"level"`, other choices are by frequency (`"descending"` or `"ascending"`) or by name of the levels (`"name"`). The argument can be abbreviated. This is ignored if x is numeric.
`useNA`	one out of `"no"`, `"ifany"`, `"always"`. Defines whether to include extra NA levels in the table. Defaults to `"no"` which is the `table()` default too.
`digits`	integer, determining the number of digits used to format the relative frequencies.
`...`	further arguments are passed to the function `cut()`. Use `dig.lab` to control the format of numeric group names. Use the argument `right` to define if the intervals should be closed on the right (and open on the left) or vice versa. In print.Freq the dots are not used.

Details

When breaks is specified as a single number, the range of the data is divided into breaks pieces of equal length, and then the outer limits are moved away by 0.1 within the break intervals. (If x is a constant vector, equal-length intervals are created that cover the single value.)

Value

an object of type `"Freq"`, which is basically a data.frame with 5 columns (earning a specific print routine), containing the following components:

`level`	factor. The levels of the grouping variable.

freq	integer. The absolute frequencies.
perc	numeric. The relative frequencies (percent).
cumfreq	integer. The cumulative sum of the absolute frequencies.
cumperc	numeric. The cumulative sum of the relative frequencies.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

PercTable, cut, hist, cumsum, table, prop.table

Examples

```
data(d.pizza)

# result is a data.frame
d.freq <- Freq(d.pizza$price)
d.freq

# it is printed by default with 3 digits for the percent values,
# but the number of digits can be defined in the print function
print(d.freq, digits=5)

# sorted by frequency
Freq(d.pizza$driver, ord="desc")

# sorted by name, including NAs
Freq(d.pizza$driver, ord="name", useNA="ifany")
```

GCD, LCM *Greatest Common Divisor and Least Common Multiple*

Description

Calculates the greatest common divisor (GCD) and least common multiple (LCM).

Usage

```
GCD(x)
LCM(x)
```

Arguments

x	a vector of integers.

Details

The computation is based on the Euclidean algorithm without using the extended version.The greatest common divisor for all numbers in the integer vector x will be computed (the multiple GCD).

Value

A numeric (integer) value.

Note

The following relation is always true:

```
n * m = GCD(n, m) * LCM(n, m)
```

Note

This functions stem from the library **numbers** and are a combination of GCD, LCM, mGCD, mLCM.

Author(s)

Hans W Borchers <hwborchers@googlemail.com>
combined by Andri Signorell <andri@signorell.net>

See Also

Factorize, Primes

Examples

```
GCD(c(12, 10))
GCD(c(46368, 75025))  # Fibonacci numbers are relatively prime to each other

LCM(c(12, 10))
LCM(c(46368, 75025))  # = 46368 * 75025

GCD(c(2, 3, 5, 7) * 11)
GCD(c(2*3, 3*5, 5*7))
LCM(c(2, 3, 5, 7) * 11)
LCM(c(2*3, 3*5, 5*7))
```

GetAllSubsets *Get All Subsets out of a List of Elements*

Description

Returns a list with all the subsets that can be built based on the elements given in x.
The number of elements used in the combinations can be limited by setting min.n and max.n.

Usage

```
GetAllSubsets(x, min.n = 1, max.n = length(x))
```

Arguments

x	a vector with elements
min.n	the minimum count of elements to be drawn. This defaults to 1.
max.n	the maximum count of elements to be drawn. Default is "all elements in x".

Value

a list with the subsets.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetPairs, PairApply

Examples

```
x <- LETTERS[1:6]

GetAllSubsets(x)

#get all sets of size 2 and 3
GetAllSubsets(x, min.n = 2, max.n = 3)
```

GetCurrWrd *Get a Handle to a Running Word Instance*

Description

Look for a running Word instance and return its handle. NULL is returned if nothing's found.

Usage

```
GetCurrWrd()
GetCurrXL()
```

Value

a handle (pointer) to the running Word, resp. Excel instance.

Note

Closing an instance does not update the value of the pointer. So it may contain an invalid address. Whether the pointer is still valid can be checked by IsValidWrd.

Note

This does unfortunately not work with RDCOMClient (but it would with rcom)! Any better idea out there?

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetNewWrd, IsValidWrd

Examples

```
## Not run: # Windows-specific example

x <- rnorm(100)

wrd <- GetCurrWrd()

if(IsValidWrd(wrd)){
  Desc(x, wrd=wrd)
} else {
  print("GetCurrWrd: no running word instance found...")
}

## End(Not run)
```

GetNewPP *Create a new PowerPoint Instance*

Description

Start a new instance of PowerPoint and return its handle. A new presentation with one empty slide will be created. The handle is needed for addressing the presentation afterwards.

Usage

```
GetNewPP(visible = TRUE, template = "Normal")
```

Arguments

visible	logical, should PowerPoint made visible? Defaults to TRUE.
template	the name of the template to be used for creating a new presentation.

Value

a handle (pointer) to the created PowerPoint instance.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetNewXL, GetNewWrd, PpPlot

Examples

```
## Not run:  # Windows-specific example
# get a handle to a new PowerPoint instance
# (this will be used further to export R-Objects to PowerPoint)
pp <- GetNewPP()

## End(Not run)
```

GetNewWrd *Create a new Word Instance*

Description

Start a new instance of Word and return its handle. This handle allows controlling word afterwards.

Usage

```
GetNewWrd(visible = TRUE, template = "Normal", header = FALSE,
          main = "Descriptive report")
```

Arguments

visible	logical, should Word made visible? Defaults to TRUE.
template	the name of the template to be used for creating a new document.
header	logical, should a caption and a list of contents be inserted? Default is FALSE.
main	the main title of the report

Details

Note that the list of contents has to be refreshed by hand after inserting text (if inserted by `header = TRUE`.

Value

a handle (pointer) to the created Word instance.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetNewXL, GetNewPP

Examples

```
## Not run:  # Windows-specific example

wrd <- GetNewWrd()
Desc(d.pizza[,1:4], wrd=wrd)

wrd <- GetNewWrd(header=TRUE)
Desc(d.pizza[,1:4], wrd=wrd)

## End(Not run)
```

GetNewXL *Create a new Excel Instance*

Description

Start a new instance of Excel and return its handle. This is needed to address XL afterwards.

Usage

```
GetNewXL(visible = TRUE)
```

Arguments

visible	logical, should Excel made visible? Defaults to TRUE.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

XLView, XLGetRange, XLGetWorkbook

Examples

```
## Not run:  # Windows-specific example
# get a handle to a new excel instance
xl <- GetNewXL()

## End(Not run)
```

GetPairs *Get All Pairs out of one or two Sets of Elements*

Description

Returns all combinations of 2 out of the elements in x or x and y (if defined). Combinations of the same elements will be dropped (no replacing).

Usage

```
GetPairs(x, y = NULL)
```

Arguments

x	a vector of elements
y	a vector of elements, need not be same dimension as x. If y is not NULL then all combination x and y are returned.

Details

If y = NULL then all combination of 2 out of x are returned.
If y is defined then all combinations of x and y are calculated.

Value

GetPairs returns a data.frame with 2 columns X1 and X2.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

combn, expand.grid, outer, lower.tri

Examples

```
GetPairs(letters[1:4])
GetPairs(x = letters[1:4], y = LETTERS[1:2])

data(d.pizza)
# get all pairs of combinations between factors and numerics out of a data.frame
GetPairs(WhichNumerics(d.pizza), WhichFactors(d.pizza))
```

`Gini` *Gini Coefficient*

Description

Compute the Gini coefficient.

Usage

```
Gini(x, n = rep(1, length(x)), unbiased = TRUE,
     conf.level = NA, R = 1000, type = "bca", na.rm = FALSE)
```

Arguments

`x`	a vector containing at least non-negative elements.
`n`	a vector of frequencies (weights), must be same length as x.
`unbiased`	logical. In order for G to be an unbiased estimate of the true population value, calculated gini is multiplied by n/(n-1). Default is TRUE. (See Dixon, 1987)
`conf.level`	confidence level for the returned confidence interval, restricted to lie between 0 and 1. If set to TRUE the bootstrap confidence intervals are calculated. If set to NA, which is the default, no confidence intervals are returned.
`R`	The number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case R would be a vector of integers where each component gives the number of resamples from each of the rows of weights. This is ignored if no confidence intervals are to be calculated.
`type`	A vector of character strings representing the type of intervals required. The value should be any subset of the values c("norm","basic", "stud", "perc", "bca") or simply "all" which will compute all five types of intervals. This is ignored if no confidence intervals are to be calculated.
`na.rm`	logical. Should missing values be removed? Defaults to FALSE.

Details

The small sample variance properties of the Gini coefficient are not known, and large sample approximations to the variance of the coefficient are poor (Mills and Zandvakili, 1997; Glasser, 1962; Dixon et al., 1987), therefore confidence intervals are calculated via bootstrap re-sampling methods (Efron and Tibshirani, 1997).

Two types of bootstrap confidence intervals are commonly used, these are percentile and bias-corrected (Mills and Zandvakili, 1997; Dixon et al., 1987; Efron and Tibshirani, 1997). The bias-corrected intervals are most appropriate for most applications. This is set as default for the `type` argument (`"bca"`). Dixon (1987) describes a refinement of the bias-corrected method known as 'accelerated' - this produces values very closed to conventional bias corrected intervals.

(Iain Buchan (2002) *Calculating the Gini coefficient of inequality*, see: http://www.statsdirect.com/help/nonparametric_methods/gini_coefficient.htm)

Value

If conf.level is NA then the result will be a single numeric value.
If conf.level is provided the result will be a vector with 3 elements for estimate, lower confidence intervall and upper for the upper one.

Author(s)

Andri Signorell <andri@signorell.net>

References

Cowell, F. A. (2000) Measurement of Inequality in Atkinson, A. B. / Bourguignon, F. (Eds): *Handbook of Income Distribution*. Amsterdam.

Cowell, F. A. (1995) *Measuring Inequality* Harvester Wheatshef: Prentice Hall.

Marshall, Olkin (1979) *Inequalities: Theory of Majorization and Its Applications*. New York: Academic Press.

Glasser C. (1962) Variance formulas for the mean difference and coefficient of concentration. *Journal of the American Statistical Association* 57:648-654.

Mills JA, Zandvakili A. (1997). Statistical inference via bootstrapping for measures of inequality. *Journal of Applied Econometrics* 12:133-150.

Dixon, PM, Weiner J., Mitchell-Olds T, Woodley R. (1987) Boot-strapping the Gini coefficient of inequality. *Ecology* 68:1548-1551.

Efron B, Tibshirani R. (1997) Improvements on cross-validation: The bootstrap method. *Journal of the American Statistical Association* 92:548-560.

See Also

See `Herfindahl`, `Rosenbluth` for concentration measures, `Lc` for the Lorenz curve
`ineq()` in the package **ineq** contains additional inequality measures

Examples

```
# generate vector (of incomes)
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)

# compute Gini coefficient
Gini(x)

# working with weights
fl <- c(2.5,7.5,15,35,75,150)    # midpoints of classes
n  <- c(25,13,10,5,5,2)          # frequencies

Gini(fl, n, conf.level=0.95, unbiased=FALSE)

# some special cases
x <- c(10,10,0,0,0)
plot(Lc(x))

Gini(x, unbiased=FALSE)

# the same with weights
Gini(x=c(10,0), n=c(2,3), unbiased=FALSE)
```

```
# perfect balance
Gini(c(10,10,10))
```

`GiniSimpson` *Compute Gini-Simpson Coefficient*

Description

Calculate the Gini-Simpson coefficient.

Usage

```
GiniSimpson(x, na.rm = FALSE)
```

Arguments

`x`	a vector containing at least non-negative elements.
`na.rm`	logical. Should missing values be removed? Defaults to FALSE.

Details

The original Simpson index λ equals the probability that two entities taken at random from the dataset of interest (with replacement) represent the same type. The Simpson index was introduced in 1949 by Edward H. Simpson to measure the degree of concentration when individuals are classified into types. The same index was rediscovered by Orris C. Herfindahl in 1950. The square root of the index had already been introduced in 1945 by the economist Albert O. Hirschman. As a result, the same measure is usually known as the Simpson index in ecology, and as the Herfindahl index or the Herfindahl-Hirschman index (HHI) in economics.
Its transformation 1 - λ therefore equals the probability that the two entities represent different types. This measure is also known in ecology as the probability of interspecific encounter (PIE) and the Gini-Simpson index.

Value

a numeric value.

Author(s)

Andri Signorell <andri@signorell.net>

References

Cover Thomas M. and Thomas Joy A. (1991) *Elements of Information Theory*. Wiley.

See Also

`DivCoef`, `Entropy`, `Gini`, `Herfindahl`

Examples

```
x <- c(261,29,33,15,39,28,95,5,6,28,69,8,105,38,15)

GiniSimpson(x)

# is the same as
1 - Herfindahl(x)

GiniSimpson(c(783,121,112,70,201,153,425,19,37,126,325,51,442,193,41))
```

Gmean *Geometric Mean and Standard Deviation*

Description

Calculates the geometric mean and the geometric standard deviation of a vector x.

Usage

```
Gmean(x, na.rm = FALSE)

Gsd(x, na.rm = FALSE)
```

Arguments

x	a positive numeric vector. An object which is not a vector is coerced (if possible) by as.vector.
na.rm	logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Details

The geometric mean and sd are restricted to positive inputs (because otherwise the answer can have an imaginary component).
Use sapply to calculate the measures from data frame, resp. from a matrix.

Value

a numeric value.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

mean, Hmean

Examples

```
x <- runif(5)
Gmean(x)

m <- matrix(runif(50), nrow = 10)
apply(m, 2, Gmean)

sapply(as.data.frame(m), Gmean)
```

GoodmanKruskalGamma *Goodman Kruskal's Gamma*

Description

Calculate Goodman Kruskal's Gamma statistic, a measure of association for ordinal factors in a two-way table.

Usage

```
GoodmanKruskalGamma(x, y = NULL, conf.level = NA, ...)
```

Arguments

x	a numeric vector or a contingency table. A matrix will be treated as a table.
y	NULL (default) or a vector with compatible dimensions to x. If y is provided, table(x, y, ...) is calculated.
conf.level	confidence level of the interval. If set to NA (which is the default) no confidence intervals will be calculated.
...	further arguments are passed to the function table, allowing i.e. to set useNA. This refers only to the vector interface.

Details

The estimator of γ is based only on the number of concordant and discordant pairs of observations. It ignores tied pairs (that is, pairs of observations that have equal values of X or equal values of Y). Gamma is appropriate only when both variables lie on an ordinal scale.
It has the range [-1, 1]. If the two variables are independent, then the estimator of gamma tends to be close to zero. For 2x2 tables, gamma is equivalent to Yule's Q (YuleQ).
Gamma is estimated by G = (P-Q)/(P+Q), where P equals twice the number of concordances and Q twice the number of discordances.

Value

a single numeric value if no confidence intervals are requested,
and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

References

Agresti, A. (2002) *Categorical Data Analysis*. John Wiley & Sons, pp. 57-59.

Goodman, L. A., & Kruskal, W. H. (1954) Measures of association for cross classifications. *Journal of the American Statistical Association*, 49, 732-764.

Goodman, L. A., & Kruskal, W. H. (1963) Measures of association for cross classifications III: Approximate sampling theory. *Journal of the American Statistical Association*, 58, 310-364.

http://support.sas.com/onlinedoc/913/getDoc/en/statug.hlp/freq_sect18.htm
http://support.sas.com/onlinedoc/913/getDoc/en/statug.hlp/freq_sect20.htm

See Also

There's another implementation of gamma in **vcdExtra** GKgamma
ConDisPairs yields concordant and discordant pairs

Other association measures:
GoodmanKruskalTauA (tau-a), cor (method="kendall") for tau-b, StuartTauC (tau-c), SomersDelta
Lambda, UncertCoef, MutInf

Examples

```
# example in:
# http://support.sas.com/documentation/cdl/en/statugfreq/63124/PDF/default/statugfreq.pdf
# pp. S. 1821

tab <- as.table(rbind(
  c(26,26,23,18, 9),
  c( 6, 7, 9,14,23))
  )

GoodmanKruskalGamma(tab, conf.level=0.95)
```

GoodmanKruskalTauA *Goodman Kruskal's Tau a*

Description

Calculate Goodman Kruskal's tau-a statistic, a measure of association for ordinal factors in a two-way table.
The function has interfaces for a table (matrix) and for single vectors.

Usage

```
GoodmanKruskalTauA(x, y = NULL, direction = c("row", "column"), conf.level = NA, ...)
```

Arguments

x	a numeric vector or a table. A matrix will be treated as table.
y	NULL (default) or a vector with compatible dimensions to x. If y is provided, table(x, y, ...) is calculated.
direction	direction of the calculation. Can be "row" (default) or "column", where "row" calculates Goodman Kruskal's tau-a (R|C) ("column dependent").
conf.level	confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.
...	further arguments are passed to the function table, allowing i.e. to set useNA. This refers only to the vector interface.

Details

Goodman and Kruskal's tau-a is a measure of categorical association which is based entirely on the observed data and possesses a clear interpretation in terms of proportional reduction in error. It gives the probabilities of correctly assigning cases to one set of categories improved by the knowledge of another set of categories. The statistic is asymmetric and yields different results predicting row assignments based on columns than from column assignments based on rows.
The range of Goodman Kruskal's tau is [0, 1] and is computed as
TauA(C|R) = (P-Q)/(n(n-1)/2)), where P equals twice the number of concordances and Q twice the number of discordances. Goodman Kruskal tau reduces to Phi^2 in the 2x2-table case.
(Note that Goodman Kruskal tau-a does not take into consideration any ties, which makes it unpractical.)

Value

a single numeric value if no confidence intervals are requested,
and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net>, based on code from Antti Arppe <antti.arppe@helsinki.fi>

References

Agresti, A. (2002) *Categorical Data Analysis*. John Wiley & Sons, pp. 57-59.

Goodman, L. A., & Kruskal, W. H. (1954) Measures of association for cross classifications. *Journal of the American Statistical Association*, 49, 732-764.

Somers, R. H. (1962) A New Asymmetric Measure of Association for Ordinal Variables, *American Sociological Review*, 27, 799-811.

Goodman, L. A., & Kruskal, W. H. (1963) Measures of association for cross classifications III: Approximate sampling theory. *Journal of the American Statistical Association*, 58, 310-364.

http://support.sas.com/onlinedoc/913/getDoc/en/statug.hlp/freq_sect18.htm
http://support.sas.com/onlinedoc/913/getDoc/en/statug.hlp/freq_sect20.htm

See Also

ConDisPairs yields concordant and discordant pairs

Other association measures:
GoodmanKruskalTauA (Tau a), cor (method="kendall") for Tau b, StuartTauC, GoodmanKruskalGamma
Lambda, UncertCoef, MutInf

Examples

```
# example in:
# http://support.sas.com/documentation/cdl/en/statugfreq/63124/PDF/default/statugfreq.pdf
# pp. S. 1821

tab <- as.table(rbind(c(26,26,23,18,9),c(6,7,9,14,23)))

# Goodman Kruskal's tau-a C|R
GoodmanKruskalTauA(tab, direction="column", conf.level=0.95)
# Goodman Kruskal's tau-a R|C
GoodmanKruskalTauA(tab, direction="row", conf.level=0.95)

# http://support.sas.com/documentation/cdl/en/statugfreq/63124/PDF/default/statugfreq.pdf
# pp. 1814 (143)
tab <- as.table(cbind(c(11,2),c(4,6)))

GoodmanKruskalTauA(tab, direction="row", conf.level=0.95)
GoodmanKruskalTauA(tab, direction="column", conf.level=0.95)
# reduces to:
Phi(tab)^2
```

Herfindahl *Concentration Measures*

Description

Computes the concentration within a vector according to the specified concentration measure.

Usage

```
Herfindahl(x, n = rep(1, length(x)), parameter = 1, na.rm = FALSE)
Rosenbluth(x, n = rep(1, length(x)), na.rm = FALSE)
```

Arguments

x	a vector containing non-negative elements
n	a vector of frequencies (weights), must be same length as x.
parameter	parameter of the concentration measure (if set to NULL the default parameter of the respective measure is used)
na.rm	logical. Should missing values be removed? Defaults to FALSE.

Value

the value of the concentration measure

Note

The same measure is usually known as the Simpson index in ecology, and as the Herfindahl index or the Herfindahl-Hirschman index (HHI) in economics.

Note

These functions were previously published as `conc()` in the **ineq** package and have been integrated here without logical changes. NA and weights support was added.

Author(s)

Achim Zeileis <achim.zeileis@r-project.org>

References

Cowell, F. A. (2000) Measurement of Inequality, in Atkinson, A. B., Bourguignon, F. *Handbook of Income Distribution*. (Eds) Amsterdam

Cowell, F. A. (1995) *Measuring Inequality*. Prentice Hall/Harvester Wheatshef

Hall, M., Tidemann, N. (1967) *Measures of Concentration*, JASA 62, 162-168.

See Also

See `Gini`, `Atkinson` and `ineq()` for additional inequality measures

Examples

```
# generate vector (of sales)
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)

# compute Herfindahl coefficient with parameter 1
Herfindahl(x)

# compute coefficient of Hall/Tiedemann/Rosenbluth
Rosenbluth(x)

# Some more examples
Herfindahl(c(261,29,33,15,39,28,95,5,6,28,69,8,105,38,15))
Herfindahl(c(783,121,112,70,201,153,425,19,37,126,325,51,442,193,41))
```

HexToCol *Identify closest match to a color given by a hexadecimal string*

Description

Given a color as a hex string #rrggbb, find the closest match in the table of known (named) colors.

Usage

```
HexToCol(hexstr, method = "rgb", metric = "euclidean")
```

Arguments

hexstr	a color or a vector of colors specified as hexadecimal string of the form "#rrggbb" or "#rrggbbaa"
method	character string specifying the color space to be used. Can be "rgb" (default) or "hsv".
metric	character string specifying the metric to be used for calculating distances between the colors. Available options are "euclidean" (default) and "manhattan". Euclidean distances are root sum-of-squares of differences, and manhattan distances are the sum of absolute differences.

Details

Finds the color with the minimum squared distance in RGB space.

Value

The colorname(s) of the closest match(es) (if more than one).

Author(s)

Ben Bolker, vector support Andri Signorell <andri@signorell.net>

See Also

ColToHex, ColToRgb, colors

Examples

```
ColToHex(c("lightblue", "salmon"))

HexToCol(c("#ADD8E6", "#FA1572"))
HexToCol(PalHelsana())

x <- ColToRgb("darkmagenta")
x[2,] <- x[2,] + 155
RgbToCol(x)
```

HexToRgb — *Convert a Hexstring Color to a Matrix With Three Red/Green/Blue Rows*

Description

Converts a hexstring color to matrix with 3 red/green/blue rows.

Usage

```
HexToRgb(hex)
```

Arguments

hex	a color or a vector of colors specified as hexadecimal string of the form "#rrggbb" or "#rrggbbaa"

Value

a matrix with 3 rows.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

HexToCol

Examples

```
HexToRgb(c("#ADD8E6", "#FA1572"))
```

HighLow — *Return the Lowest and the Highest Values and Their Frequencies*

Description

A printing routine for the highest and the lowest values of x. It enumerates the according values and their frequencies (in brackets).

Usage

```
HighLow(x, nlow = 5, nhigh = nlow, na.rm = FALSE)
```

Arguments

x	a numeric vector or an ordered factor.
nlow	a single integer. The number of the smallest elements of a vector to be printed. Defaults to 5.
nhigh	a single integer. The number of the greatest elements of a vector to be printed. Defaults to the number of nlow.
na.rm	logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Details

This is used for describing univariate variables and is interesting for checking the ends of the vector, where in real data often wrong values accumulate.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

min, max, table

Examples

```
cat(HighLow(d.pizza$temperature, na.rm=TRUE))
```

Hmean *Harmonic mean*

Description

Calculates the harmonic mean of a vector x.

Usage

```
Hmean(x, na.rm = FALSE)
```

Arguments

x	a positive numeric vector. An object which is not a vector is coerced (if possible) by as.vector.
na.rm	logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Details

Use sapply to calculate the measures from data frame, resp. from a matrix.

Value

a numeric value.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Gmean

Examples

```
x <- runif(5)
Hmean(x)

m <- matrix(runif(50), nrow = 10)
apply(m, 2, Hmean)

sapply(as.data.frame(m), Hmean)
```

HoeffD *Matrix of Hoeffding's D Statistics*

Description

Computes a matrix of Hoeffding's (1948) D statistics for all possible pairs of columns of a matrix. D is a measure of the distance between F(x,y) and G(x)H(y), where F(x,y) is the joint CDF of X and Y, and G and H are marginal CDFs. Missing values are deleted in pairs rather than deleting all rows of x having any missing variables. The D statistic is robust against a wide variety of alternatives to independence, such as non-monotonic relationships. The larger the value of D, the more dependent are X and Y (for many types of dependencies). D used here is 30 times Hoeffding's original D, and ranges from -0.5 to 1.0 if there are no ties in the data. print.HoeffD prints the information derived by HoeffD. The higher the value of D, the more dependent are x and y.

Usage

```
HoeffD(x, y)
## S3 method for class 'HoeffD'
print(x, ...)
```

Arguments

x	a numeric matrix with at least 5 rows and at least 2 columns (if y is absent), or an object created by HoeffD
y	a numeric vector or matrix which will be concatenated to x
...	ignored

Details

Uses midranks in case of ties, as described by Hollander and Wolfe. P-values are approximated by linear interpolation on the table in Hollander and Wolfe, which uses the asymptotically equivalent Blum-Kiefer-Rosenblatt statistic. For `P<.0001` or `>0.5`, `P` values are computed using a well-fitting linear regression function in `log P` vs. the test statistic. Ranks (but not bivariate ranks) are computed using efficient algorithms (see reference 3).

Value

a list with elements `D`, the matrix of D statistics, `n` the matrix of number of observations used in analyzing each pair of variables, and `P`, the asymptotic P-values. Pairs with fewer than 5 non-missing values have the D statistic set to NA. The diagonals of `n` are the number of non-NAs for the single variable corresponding to that row and column.

Author(s)

Frank Harrell <f.harrell@vanderbilt.edu>
Department of Biostatistics
Vanderbilt University

References

Hoeffding W. (1948): A non-parametric test of independence. *Ann Math Stat* 19:546–57.

Hollander M. and Wolfe D.A. (1973): *Nonparametric Statistical Methods*, pp. 228–235, 423. New York: Wiley.

Press WH, Flannery BP, Teukolsky SA, Vetterling, WT (1988): *Numerical Recipes in C* Cambridge: Cambridge University Press.

See Also

`rcorr`, `varclus`

Examples

```
x <- c(-2, -1, 0, 1, 2)
y <- c(4,   1, 0, 1, 4)
z <- c(1,   2, 3, 4, NA)
q <- c(1,   2, 3, 4, 5)

HoeffD(cbind(x, y, z, q))


# Hoeffding's test can detect even one-to-many dependency
set.seed(1)
x <- seq(-10, 10, length=200)
y <- x * sign(runif(200, -1, 1))
plot(x, y)

HoeffD(x, y)
```

HuberM *Safe (generalized) Huber M-Estimator of Location*

Description

(Generalized) Huber M-estimator of location with MAD scale, being sensible also when the scale is zero where `huber()` returns an error.

Usage

```
HuberM(x, k = 1.5, weights = NULL, tol = 1e-06,
       mu = if(is.null(weights)) median(x) else wgt.himedian(x, weights),
       s =  if(is.null(weights)) mad(x, center=mu)
    else wgt.himedian(abs(x - mu), weights),
       se = FALSE,
       warn0scale = getOption("verbose"))
```

Arguments

`x`	numeric vector.
`k`	positive factor; the algorithm winsorizes at `k` standard deviations.
`weights`	numeric vector of non-negative weights of same length as `x`, or `NULL`.
`tol`	convergence tolerance.
`mu`	initial location estimator.
`s`	scale estimator held constant through the iterations.
`se`	logical indicating if the standard error should be computed and returned (as `SE` component). Currently only available when `weights` is `NULL`.
`warn0scale`	logical; if true, and `s` is 0 and `length(x) > 1`, this will be warned about.

Details

Note that currently, when non-`NULL` `weights` are specified, the default for initial location `mu` and scale `s` is `wgt.himedian`, where strictly speaking a weighted "non-hi" median should be used for consistency. Since `s` is not updated, the results slightly differ, see the examples below.

When `se = TRUE`, the standard error is computed using the τ correction factor but no finite sample correction.

Value

list of location and scale parameters, and number of iterations used.

`mu`	location estimate
`s`	the `s` argument, typically the `mad`.
`it`	the number of "Huber iterations" used.

Author(s)

Martin Maechler, building on the MASS code mentioned.

References

Huber, P. J. (1981) *Robust Statistics.* Wiley.

See Also

hubers (and huber) in package **MASS**; mad.

Examples

```
HuberM(c(1:9, 1000))
mad   (c(1:9, 1000))
mad   (rep(9, 100))
HuberM(rep(9, 100))

## When you have "binned" aka replicated observations:
set.seed(7)
x <- c(round(rnorm(1000),1), round(rnorm(50, m=10, sd = 10)))
t.x <- table(x) # -> unique values and multiplicities
x.uniq <- as.numeric(names(t.x)) ## == sort(unique(x))
x.mult <- unname(t.x)
str(Hx  <- HuberM(x.uniq, weights = x.mult), digits = 7)
str(Hx. <- HuberM(x, s = Hx$s, se=TRUE), digits = 7) ## should be ~= Hx
stopifnot(all.equal(Hx[-4], Hx.[-4]))
str(Hx2 <- HuberM(x, se=TRUE), digits = 7)## somewhat different, since 's' differs

## Confirm correctness of std.error :

system.time(
SS <- replicate(10000, vapply(HuberM(rnorm(400), se=TRUE), as.double, 1.))
) # ~ 12.2 seconds
rbind(mean(SS["SE",]), sd(SS["mu",]))# both ~ 0.0508
stopifnot(all.equal(mean(SS["SE",]),
                    sd ( SS["mu",]), tol= 0.002))
```

ICC *Intraclass Correlations (ICC1, ICC2, ICC3 from Shrout and Fleiss)*

Description

The Intraclass correlation is used as a measure of association when studying the reliability of raters. Shrout and Fleiss (1979) outline 6 different estimates, that depend upon the particular experimental design. All are implemented and given confidence limits.

Usage

```
ICC(x, missing = TRUE, alpha = 0.05)
## S3 method for class 'ICC'
print(x, digits = 2, ...)
```

Arguments

x	a matrix or dataframe of ratings
missing	if TRUE, remove missing data – work on complete cases only
alpha	the alpha level for significance for finding the confidence intervals
digits	number of digits to use in printing
...	further arguments to be passed to or from methods.

Details

Shrout and Fleiss (1979) consider six cases of reliability of ratings done by k raters on n targets.

ICC1: Each target is rated by a different judge and the judges are selected at random. (This is a one-way ANOVA fixed effects model and is found by (MSB- MSW)/(MSB+ (nr-1)*MSW))

ICC2: A random sample of k judges rate each target. The measure is one of absolute agreement in the ratings. Found as (MSB- MSE)/(MSB + (nr-1)*MSE + nr*(MSJ-MSE)/nc)

ICC3: A fixed set of k judges rate each target. There is no generalization to a larger population of judges. (MSB - MSE)/(MSB+ (nr-1)*MSE)

Then, for each of these cases, is reliability to be estimated for a single rating or for the average of k ratings? (The 1 rating case is equivalent to the average intercorrelation, the k rating case to the Spearman Brown adjusted reliability.)

ICC1 is sensitive to differences in means between raters and is a measure of absolute agreement.

ICC2 and ICC3 remove mean differences between judges, but are sensitive to interactions of raters by judges. The difference between ICC2 and ICC3 is whether raters are seen as fixed or random effects.

ICC1k, ICC2k, ICC3K reflect the means of k raters.

The intraclass correlation is used if raters are all of the same "class". That is, there is no logical way of distinguishing them. Examples include correlations between pairs of twins, correlations between raters. If the variables are logically distinguishable (e.g., different items on a test), then the more typical coefficient is based upon the inter-class correlation (e.g., a Pearson r) and a statistic such as alpha or omega might be used.

Value

results	A matrix of 6 rows and 8 columns, including the ICCs, F test, p values, and confidence limits
summary	The anova summary table
stats	The anova statistics
MSW	Mean Square Within based upon the anova

Note

The results for the Lower and Upper Bounds for ICC(2,k) do not match those of SPSS 9 or 10, but do match the definitions of Shrout and Fleiss. SPSS seems to have been using the formula in McGraw and Wong, but not the errata on p 390. They seem to have fixed it in more recent releases (15).

Author(s)

William Revelle

References

Shrout, Patrick E. and Fleiss, Joseph L. Intraclass correlations: uses in assessing rater reliability. *Psychological Bulletin*, 1979, 86, 420-3428.

McGraw, Kenneth O. and Wong, S. P. (1996), Forming inferences about some intraclass correlation coefficients. *Psychological Methods*, 1, 30-46. + errata on page 390.

Revelle, W. (in prep) *An introduction to psychometric theory with applications in R* Springer. (working draft available at http://personality-project.org/r/book/

Examples

```
sf <- matrix(c(
      9, 2, 5, 8,
      6, 1, 3, 2,
      8, 4, 6, 8,
      7, 1, 2, 6,
      10,5, 6, 9,
      6, 2, 4, 7),
      ncol=4, byrow=TRUE,
      dimnames=list(paste("S", 1:6, sep=""), paste("J", 1:4, sep="")))
)

sf  #example from Shrout and Fleiss (1979)
ICC(sf)
```

`identify.formula` *Identify points in a plot using a formula.*

Description

The function `identify` reads the position of the graphics pointer when the (first) mouse button is pressed. It then searches the coordinates given in x and y for the point closest to the pointer. If this point is close enough to the pointer, its index will be returned as part of the value of the call.

Usage

```
 ## S3 method for class 'formula'
identify(x, data = NULL, ...)
```

Arguments

`x`	A formula of the form `y ~ x`.
`data`	The data frame from which the formula should be evaluated.
`...`	Other arguments to be passed to `identify`.

Details

This function is meant to make it easier to call `identify` after `plot` has been called using a formula and the `data` argument.

A two dimensional plot must be active and the vectors in `x` and data frame in `data` must correspond to the x- and y-axes and the data of the plot.

Value

If pos is FALSE, an integer vector containing the indices of the identified points, in the order they were identified. If pos is TRUE, a list containing a component ind, indicating which points were identified and a component pos, indicating where the labels were placed relative to the identified points (1=below, 2=left, 3=above, 4=right and 0=no offset, used if atpen = TRUE).

Author(s)

Derek Ogle <dogle@northland.edu>

See Also

identify, locator, text
http://www.rforge.net/NCStats/files/

Examples

```
## Not run:
## Copy and try in an interactive R session

df <- data.frame(x=runif(10), y=runif(10), grp=factor(rep(c("Yes","No"), each=5)))
plot(y ~ x, data=df)
identify(y ~ x, data=df)

## End(Not run)
```

ImportDlg *Get Path of a Data File to Be Opened*

Description

Handling of pathnames is tedious in Windows because of the backslashes, that prevent simple pasting of a copied path into the source code. ImportDlg displays the FileOpen-Dialog for picking a file interactively. When done backslashes in the path are replaced by slashes and the result is being copied into the clipboard, from where it can easily be pasted in any code editor.

Usage

```
ImportDlg(fmt = 1)
```

Arguments

fmt	the format, in which the filename parts should be returned. Default ist path\filename.ext and coded as "\"%path%\%fname%.%fxt%\"". See examples for time saving alternative definitions.

Value

ImportDlg() invisibly returns the path of the chosen file in the defined format. The result is additionally being copied to the clipboard.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

file.choose

Examples

```
## Not run:
# choose a file
fn <- ImportDlg()
print(gettextf("You chose the file: %s ", fn))


# the path and filename can as well be nested in a command,
#   done here to build a read.table command that can be well inserted into the code:
ImportDlg(fmt="d.%fname% <- read.table(file = \"%path%\%fname%.%fxt%\",
  header = TRUE, sep = \";\", na.strings = c(\"NA\",\"NULL\"), strip.white = TRUE)")

# go to your editor and paste...



## End(Not run)
```

InDots *Is a Specific Argument in the Dots-Arguments?*

Description

Returns TRUE if a specific named argument was given in the dots.

Usage

```
InDots(..., arg, default)
```

Arguments

...	the dots arguments of the function whose arguments are to be checked.
arg	the name of argument to test for.
default	the default value to return, if the argument arg does not exist in the dots.

Value

the value of the argument, if it exists else the specified default value.

Author(s)

Andri Signorell <andri@signorell.net>

IsDate *Check if an Object is of Type Date*

Description

Check if the given x is of any known Date type.

Usage

```
IsDate(x, what = c("either", "both", "timeVaries"))
```

Arguments

x	a vector or values to be checked.
what	can be any value out of "either" (default), "both" or "timeVaries".

Details

This checks for many known Date and Time classes: "POSIXt", "POSIXct", "dates", "times", "chron", "Date".

Value

logical vector of the same dimension as x.

Author(s)

Frank E Harrell

See Also

Year, Month, etc.

Examples

```
IsDate(as.Date("2013-04-10"))

IsDate(31002)
```

IsEuclid *Is a Distance Matrix Euclidean?*

Description

Confirmation of the Euclidean nature of a distance matrix by the Gower's theorem.
IsEuclid is used in summary.dist.

Usage

```
IsEuclid(distmat, plot = FALSE, print = FALSE, tol = 1e-07)
```

Arguments

distmat	an object of class 'dist'
plot	a logical value indicating whether the eigenvalues bar plot of the matrix of the term $-\frac{1}{2}d_{ij}^2$ centred by rows and columns should be diplayed
print	a logical value indicating whether the eigenvalues of the matrix of the term $-\frac{1}{2}d_{ij}^2$ centred by rows and columns should be printed
tol	a tolerance threshold : an eigenvalue is considered positive if it is larger than -tol*lambda1 where lambda1 is the largest eigenvalue.

Value

returns a logical value indicating if all the eigenvalues are positive or equal to zero

Author(s)

Daniel Chessel
Stephane Dray <dray@biomserv.univ-lyon1.fr>

References

Gower, J.C. and Legendre, P. (1986) Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification*, **3**, 5–48.

Examples

```
w <- matrix(runif(10000), 100, 100)
w <- dist(w)
summary(w)
IsEuclid (w) # TRUE
```

IsPrime *IsPrime Property*

Description

Vectorized version, returning for a vector or matrix of positive integers a vector of the same size containing 1 for the elements that are prime and 0 otherwise.

Usage

```
IsPrime(x)
```

Arguments

x	vector or matrix of nonnegative integers

Details

Given an array of positive integers returns an array of the same size of 0 and 1, where the i indicates a prime number in the same position.

Value

array of elements 0, 1 with 1 indicating prime numbers

Author(s)

Hans W. Borchers <hwborchers@googlemail.com>

See Also

Factorize, Primes

Examples

```
x <- matrix(1:10, nrow=10, ncol=10, byrow=TRUE)
x * IsPrime(x)

# Find first prime number octett:
octett <- c(0, 2, 6, 8, 30, 32, 36, 38) - 19
while (TRUE) {
    octett <- octett + 210
    if (all(IsPrime(octett))) {
        cat(octett, "\n", sep="  ")
        break
    }
}
```

IsValidWrd *Check Word Pointer*

Description

Check if a pointer points to a valid and running Word instance. The function does this by trying to get the current selection of the Word instance and returns FALSE if it's NULL.

Usage

```
IsValidWrd(wrd = getOption("lastWord"))
```

Arguments

wrd	the pointer to a word instance as created by GetNewWrd() or GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").

Value

logical value

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetCurrWrd()

IsWhole *Is x a Whole Number?*

Description

Test if x contains integer numbers.

Usage

```
IsWhole(x, tol = .Machine$double.eps^0.5, na.rm = FALSE)
```

Arguments

x	a (non-empty) numeric vector of data values.
tol	tolerance to be used
na.rm	logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Details

This is the suggested solution for doing the job, as is.integer tests for (class(x) == "integer") and does not test if x contains integer numbers. (Why not simply implement it in **base**?)

Value

logical vector of the same dimension as x.

Author(s)

R-Core

See Also

`is.integer`

Examples

```
(x <- seq(1,5, by=0.5))
IsWhole( x ) #-->  TRUE FALSE TRUE ...
```

`JarqueBeraTest` *Jarque–Bera Test*

Description

Tests the null of normality for `x` using the Jarque-Bera test statistic.

Usage

```
JarqueBeraTest(x)
```

Arguments

`x`	a numeric vector or time series.

Details

This test is a joint statistic using skewness and kurtosis coefficients.

Missing values are not allowed.

Value

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

`statistic`	the value of the test statistic.
`parameter`	the degrees of freedom.
`p.value`	the p-value of the test.
`method`	a character string indicating what type of test was performed.
`data.name`	a character string giving the name of the data.

Note

This function was previously published as `jarque.bera.test()` in the **tseries** package and has been integrated here without logical changes.

Author(s)

Adrian Trapletti

References

Cromwell, J B, Labys, W C and Terraza, M (1994): *Univariate Tests for Time Series Models*, Sage, Thousand Oaks, CA, pages 20–22.

Examples

```
x <- rnorm(100)  # null
JarqueBeraTest(x)

x <- runif(100)  # alternative
JarqueBeraTest(x)
```

`JonckheereTerpstraTest` *Exact Version of Jonckheere-Terpstra Test*

Description

Jonckheere-Terpstra test to test for ordered differences among classes.

Usage

```
JonckheereTerpstraTest(x, ...)

## Default S3 method:
JonckheereTerpstraTest(x, g, alternative = c("two.sided", "increasing", "decreasing"),
                       nperm = NULL, ...)

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

Arguments

`x`	a numeric vector of data values, or a list of numeric data vectors.
`g`	a vector or factor object giving the group for the corresponding elements of x. Ignored if x is a list.
`alternative`	means are monotonic (`two.sided`), `increasing`, or `decreasing`
`nperm`	number of permutations for the reference distribution. The default is `NULL` in which case the permutation p-value is not computed. It's recommended to set `nperm` to 1000 or higher if permutation p-value is desired.
`formula`	a formula of the form `lhs ~ rhs` where `lhs` gives the data values and rhs the corresponding groups.
`data`	an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.

subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
...	further argument to be passed to methods.

Details

JonckheereTerpstraTest is the exact (permutation) version of the Jonckheere-Terpstra test. It uses the statistic

$$\sum_{k<l}\sum_{ij} I(X_{ik} < X_{jl}) + 0.5I(X_{ik} = X_{jl}),$$

where i, j are observations in groups k and l respectively. The asymptotic version is equivalent to cor.test(x, g, method="k"). The exact calculation requires that there be no ties and that the sample size is less than 100. When data are tied and sample size is at most 100 permutation p-value is returned.

If x is a list, its elements are taken as the samples to be compared, and hence have to be numeric data vectors. In this case, g is ignored, and one can simply use JonckheereTerpstraTest(x) to perform the test. If the samples are not yet contained in a list, use JonckheereTerpstraTest(list(x, ...)).

Otherwise, x must be a numeric data vector, and g must be a vector or factor object of the same length as x giving the group for the corresponding elements of x.

Note

The function was previously published as jonckheere.test() in the **clinfun** package and has been integrated here without logical changes. Some argument checks and a formula interface were added.

Author(s)

Venkatraman E. Seshan <seshanv@mskcc.org>, minor adaptations Andri Signorell

References

Jonckheere, A. R. (1954). A distribution-free k-sample test again ordered alternatives. *Biometrika* 41:133-145.

Terpstra, T. J. (1952). The asymptotic normality and consistency of Kendall's test against trend, when ties are present in one ranking. *Indagationes Mathematicae* 14:327-333.

Examples

```
set.seed(1234)
g <- ordered(rep(1:5, rep(10,5)))
x <- rnorm(50) + 0.3 * as.numeric(g)

JonckheereTerpstraTest(x, g)

x[1:2] <- mean(x[1:2]) # tied data

JonckheereTerpstraTest(x, g)
JonckheereTerpstraTest(x, g, nperm=5000)
```

```
# Duller, S. 222
coffee <- data.frame(
  time=c(
  447,396,383,410,
  438,521,468,391,504,472,
  513,543,506,489,407),
  grp=Untable(c(4,6,5), type="ordered")[,1]
)

JonckheereTerpstraTest(coffee$time, coffee$grp)
```

Kappam *Kappa for m raters*

Description

Computes kappa as an index of interrater agreement between m raters on categorical data.

Usage

```
Kappam(x, method = c("Fleiss", "Conger", "Light"), conf.level = NA)
```

Arguments

x	n*m matrix or dataframe, n subjects m raters.
method	a logical indicating whether the exact Kappa (Conger, 1980), the Kappa described by Fleiss (1971) or Light's Kappa (1971) should be computed.
conf.level	confidence level of the interval. If set to NA (which is the default) no confidence intervals will be calculated.

Details

Missing data are omitted in a listwise way.
The coefficient described by Fleiss (1971) does not reduce to Cohen's Kappa (unweighted) for m=2 raters. Therefore, the exact Kappa coefficient, which is slightly higher in most cases, was proposed by Conger (1980).
Light's Kappa equals the average of all possible combinations of bivariate Kappas between raters.
The confidence levels can only be reported using Fleiss' formulation of Kappa.

Value

a single numeric value if no confidence intervals are requested,
and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Note

This function was previously published as `kappam.fleiss()` in the **irr** package and has been integrated here with some changes in the interface.

Author(s)

Matthias Gamer, with some modifications by Andri Signorell <andri@signorell.net>

References

Conger, A.J. (1980): Integration and generalisation of Kappas for multiple raters. *Psychological Bulletin*, 88, 322-328

Fleiss, J.L. (1971): Measuring nominal scale agreement among many raters *Psychological Bulletin*, 76, 378-382

Fleiss, J.L., Levin, B., & Paik, M.C. (2003): *Statistical Methods for Rates and Proportions*, 3rd Edition. New York: John Wiley & Sons

Light, R.J. (1971): Measures of response agreement for qualitative data: Some generalizations and alternatives. *Psychological Bulletin*, 76, 365-377.

See Also

CohenKappa

Examples

```
statement <- data.frame(
  A=c(2,3,1,3,1,2,1,2,3,3,3,3,3,2,1,3,3,2,2,1,
      2,1,3,3,2,2,1,2,1,1,2,3,3,3,3,3,1,2,1,1),
  B=c(2,2,2,1,1,2,1,2,3,3,2,3,1,3,1,1,3,2,1,2,
      2,1,3,2,2,2,3,2,1,1,2,2,3,3,3,3,2,2,2,3),
  C=c(2,2,2,1,1,2,1,2,3,3,2,3,3,3,3,2,2,2,2,3,
      2,2,3,3,2,2,3,2,2,2,2,3,3,3,3,3,3,2,2,2),
  D=c(2,2,2,1,1,2,1,2,3,3,2,3,3,3,3,3,2,2,2,2,
      3,1,3,2,2,2,1,2,2,1,2,3,3,3,3,3,3,2,2,1),
  E=c(2,2,2,3,3,2,3,1,3,3,2,3,3,3,3,3,2,2,2,3,
      2,3,3,2,2,2,3,2,1,3,2,3,3,1,3,3,3,2,2,1),
  F=c(2,3,3,1,1,2,1,2,3,3,3,3,1,3,2,1,2,1,1,3,
      2,1,1,3,2,2,1,2,1,3,2,3,3,2,1,3,3,2,2,1),
  G=c(2,2,3,2,1,2,1,3,3,3,2,1,3,3,3,2,3,3,2,3,
      2,2,3,3,2,2,2,2,1,1,2,2,3,2,3,3,3,2,2,1),
  H=c(2,2,2,2,1,2,1,1,1,3,2,2,3,2,1,2,1,2,1,1,
      2,1,3,2,2,2,2,2,1,3,2,3,3,1,3,2,3,2,2,1),
  I=c(2,2,2,2,1,2,1,2,3,3,2,2,1,1,1,2,2,2,2,2,
      2,1,3,3,2,2,1,2,1,1,2,3,3,2,1,2,2,2,2,1),
  J=c(2,2,2,2,3,2,1,2,3,3,2,3,1,3,2,2,2,2,2,3,
      2,1,3,3,2,2,1,2,1,1,2,3,3,2,3,3,1,3,2,1)
)

Kappam(statement)

Kappam(statement, method="Conger")   # Exact Kappa
Kappam(statement, conf.level=0.95)   # Fleiss' Kappa and confidence intervals

Kappam(statement, method="Light")   # Exact Kappa
```

KendallTauB *Kendall tau-b*

Description

Calculate Kendall's tau-b. The estimator could also be calculated with `cor(..., method="kendall")`. The calculation of confidence intervals however would not be found there.

Usage

```
KendallTauB(x, y = NULL, conf.level = NA, ...)
```

Arguments

`x`	a numeric vector, matrix or data.frame.
`y`	NULL (default) or a vector with compatible dimensions to `x`. If y is provided, `table(x, y, ...)` is calculated.
`conf.level`	confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.
`...`	further arguments are passed to the function `table`, allowing i.e. to set useNA. This refers only to the vector interface.

Value

a single numeric value if no confidence intervals are requested,
and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

References

Agresti, A. (2002) *Categorical Data Analysis*. John Wiley & Sons, pp. 57-59.

http://support.sas.com/onlinedoc/913/getDoc/en/statug.hlp/freq_sect18.htm
http://support.sas.com/onlinedoc/913/getDoc/en/statug.hlp/freq_sect20.htm

See Also

`ConDisPairs` yields concordant and discordant pairs

Other association measures:
`GoodmanKruskalTauA` (tau-a), `cor` (method="kendall") for tau-b, `StuartTauC` (tau-c), `SomersDelta` `Lambda`, `UncertCoef`, `MutInf`

Examples

```
# example in:
# http://support.sas.com/documentation/cdl/en/statugfreq/63124/PDF/default/statugfreq.pdf
# pp. S. 1821

tab <- as.table(rbind(c(26,26,23,18,9),c(6,7,9,14,23)))

KendallTauB(tab, conf.level=0.95)
```

KendallW *Kendall's Coefficient of Concordance W*

Description

Computes Kendall's coefficient of concordance as an index of interrater reliability of ordinal data. The coefficient could be corrected for ties within raters.

Usage

```
KendallW(ratings, correct = FALSE, test = FALSE)
```

Arguments

ratings	n*m matrix or dataframe, n subjects m raters.
correct	a logical indicating whether the coefficient should be corrected for ties within raters.
test	a logical indicating whether the test statistic and p-value should be reported.

Details

Missing data are omitted in a listwise way.
Kendall's W should be corrected for ties if raters did not use a true ranking order for the subjects.
A test for the significance of Kendall's W is only valid for large samples.

Value

Either a single value if test is set to FALSE or else

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

statistic	the value of the chi-square statistic.
p.value	the p-value for the test.
method	the character string "Kendall's coefficient of concordance W".
data.name	a character string giving the name(s) of the data.
estimate	the coefficient of concordance.
parameter	the degrees of freedom df, the number of subjects examined and the number of raters.

Note

This function was previously published as `kendall()` in the **irr** package and has been integrated here without logical changes, but with some adaptations in the result structure.

Author(s)

Matthias Gamer

References

Kendall, M.G. (1948) Rank correlation methods. London: Griffin.

See Also

`cor`

Examples

```
anxiety <- data.frame(rater1=c(3,3,3,4,5,5,2,3,5,2,2,6,1,5,2,2,1,2,4,3),
                      rater2=c(3,6,4,6,2,4,2,4,3,3,2,3,3,3,2,2,1,3,3,4),
                      rater3=c(2,1,4,4,3,2,1,6,1,1,1,2,3,3,1,1,3,3,2,2))

KendallW(anxiety, TRUE)

# with test results
KendallW(anxiety, TRUE, test=TRUE)
```

KrippAlpha *Krippendorff's Alpha Reliability Coefficient*

Description

Calculate the alpha coefficient of reliability proposed by Krippendorff.

Usage

```
KrippAlpha(x, method=c("nominal","ordinal","interval","ratio"))
```

Arguments

`x`	classifier x object matrix of classifications or scores
`method`	data level of x

Value

A list with class '"irrlist"' containing the following components:

`$method`	a character string describing the method.
`$subjects`	the number of data objects.
`$raters`	the number of raters.
`$irr.name`	a character string specifying the name of the coefficient.

$value	value of alpha.
$stat.name	here "nil" as there is no test statistic.
$statistic	the value of the test statistic (NULL).
$p.value	the probability of the test statistic (NULL).
cm	the concordance/discordance matrix used in the calculation of alpha
data.values	a character vector of the unique data values
levx	the unique values of the ratings
nmatchval	the count of matches, used in calculation
data.level	the data level of the ratings ("nominal","ordinal", "interval","ratio")

Note

Krippendorff's alpha coefficient is particularly useful where the level of measurement of classification data is higher than nominal or ordinal.

Note

This function was previously published as `kripp.alpha()` in the **irr** package and has been integrated here without logical changes, but with some adaptations in the result structure.

Author(s)

Jim Lemon

References

Krippendorff, K. (1980). Content analysis: An introduction to its methodology. Beverly Hills, CA: Sage.

Examples

```
# the "C" data from Krippendorff
nmm<-matrix(c(1,1,NA,1,2,2,3,2,3,3,3,3,3,3,3,2,2,2,2,1,2,3,4,4,4,4,4,
1,1,2,1,2,2,2,2,NA,5,5,5,NA,NA,1,1,NA,NA,3,NA),nrow=4)
# first assume the default nominal classification
KrippAlpha(nmm)
# now use the same data with the other three methods
KrippAlpha(nmm,"ordinal")
KrippAlpha(nmm,"interval")
KrippAlpha(nmm,"ratio")
```

Label *Label Attribute of an Object*

Description

Set and retrieve the `label` attribute of `x`. This can be helpful for documenting the specific meaning of a variable.

Usage

```
Label(x, default = NULL, ...)

## Default S3 method:
Label(x, ...)

## S3 method for class 'data.frame'
Label(x, ...)

Label(x, ...) <- value

## Default S3 replacement method:
Label(x, ...) <- value

## S3 replacement method for class 'data.frame'
Label(x, self = TRUE, ...) <- value
```

Arguments

x	any object
default	any default
value	any object
self	any object
...	the dots are passed to the specific function.

Details

The label should consist of a single text (length of 1). The text may contain any line feeds. It can be deleted by setting the label to NULL.

Value

Label returns the label attribute of x, if any; otherwise, NULL.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

A more elaborated version can be found in package **Hmisc** label().

Examples

```
# add a descriptive label to a variable
Label(d.diamonds$colour) <- "The rating scale applied to diamonds ranges from colorless
to yellow, as any other color is extremely rare."

# technically just appending the text as attribute to the variable
attributes(d.diamonds$colour)
```

```
# label is supported while describing data
Desc(d.diamonds$colour)

# The label can be deleted by setting it to NULL
Label(d.diamonds$colour) <- NULL
```

Lambda *Goodman Kruskal Lambda*

Description

Calculate symmetric and asymmetric Goodman Kruskal lambda and their confidence intervals.

Usage

```
Lambda(x, y = NULL, direction = c("symmetric", "row", "column"), conf.level = NA, ...)
```

Arguments

x	a numeric vector, a matrix or a table.
y	NULL (default) or a vector with compatible dimensions to x. If y is provided, table(x, y, ...) is calculated.
direction	direction of the calculation. Can be one out of "symmetric" (default), "row", "column" (abbreviations are allowed). If direction is set to "row" then Lambda(R|C) (column dependent) will be reported.
conf.level	confidence level for the returned confidence interval, restricted to lie between 0 and 1.
...	further arguments are passed to the function table, allowing i.e. to set useNA.

Details

Asymmetric lambda is interpreted as the probable improvement in predicting the column variable Y given knowledge of the row variable X.
The nondirectional lambda is the average of the two asymmetric lambdas, Lambda(C|R) and Lambda(R|C).
Lambda (asymmetric and symmetric) has a scale ranging from 0 to 1.
For x either a matrix or data.frame, or two vectors x and y are expected. In latter case table(x, y) is calculated. The function handles NAs the same way the table function does, so tables are by default calculated with omitted NAs. Use PairApply to calculate pairwise lambdas. Use complete.cases, if only the complete cases of a data.frame are to be used.

Value

a single numeric value if no confidence intervals are requested,
and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net> based on code from Antti Arppe <antti.arppe@helsinki.fi>, Nanina Anderegg (confidence interval symmetric lambda)

References

Agresti, A. (2002) *Categorical Data Analysis*. John Wiley & Sons

Goodman, L. A., and W. H. Kruskal. 1979. *Measures of Association for Cross Classifications*. New York: Springer-Verlag (contains articles appearing in J. Amer. Statist. Assoc. in 1954, 1959, 1963, 1972).

See Also

GoodmanKruskalGamma, SomersDelta, StuartTauC, GoodmanKruskalTauA, KendallTauB, cor

Examples

```
# example from Goodman Kruskal (1954)

m <- as.table(cbind(c(1768,946,115), c(807,1387,438), c(189,746,288), c(47,53,16)))
dimnames(m) <- list(paste("A", 1:3), paste("B", 1:4))
m

# direction default is "symmetric"
Lambda(m)
Lambda(m, conf.level=0.95)

Lambda(m, direction="row")
Lambda(m, direction="column")
```

Large *Kth Smallest/Largest Values*

Description

This function returns the kth smallest, resp. largest values from a vector x.

Usage

```
Large(x, k = 1, unique = FALSE, na.rm = FALSE)
```

Arguments

x	a numeric vector
k	an integer >0 defining how many extreme values should be returned. Default is k = 1. If k > length(x), all values will be returned.
unique	logical, defining if unique values should be considered or not. If this is set to TRUE a list with the extreme value and its frequency is returned. Default is FALSE.
na.rm	logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Details

There are several points discussed about that out there. This implementation uses the function sort(partial), which is'nt the fastest, but a fairly fast one.

Value

either a vector with the k most extreme values, if unique is set to FALSE or a list, containing the k most extreme values and their respective frequency.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`max`, `HighLow`

Examples

```
x <- sample(1:10, 1000, rep=TRUE)
Large(x, 3)
Large(x, k=3, unique=TRUE)

# works fine up to 1 M
x <- runif(1000000)
Small(x, 3, unique=TRUE)
Small(x, 3, unique=FALSE)
```

Lc *Lorenz Curve*

Description

Computes the (empirical) ordinary and generalized Lorenz curve of a vector x

Usage

```
Lc(x, n = rep(1, length(x)), na.rm = FALSE, plot = FALSE)

## S3 method for class 'Lc'
plot(x, general = FALSE, lwd = 2, type = "l", xlab = "p", ylab = "L(p)",
        main = "Lorenz curve", las = 1, ...)
```

Arguments

x	a vector containing non-negative elements.
n	a vector of frequencies, must be same length as x.
plot	logical. If TRUE the empirical Lorenz curve will be plotted.
na.rm	logical. Should missing values be removed? Defaults to FALSE.
general	logical. If TRUE the empirical Lorenz curve will be plotted.
lwd	Linewidth of the curve
type	type of the plot, default is line ("l").
xlab, ylab	Label of the x-, resp. y-axis.
main	main title of the plot.
las	las of the axis.
...	further argument to be passed to methods.

Details

Lc(x) computes the empirical ordinary Lorenz curve of x as well as the generalized Lorenz curve (= ordinary Lorenz curve * mean(x)). The result can be interpreted like this: p*100 percent have L(p)*100 percent of x.

If n is changed to anything but the default x is interpreted as a vector of class means and n as a vector of class frequencies: in this case Lc will compute the minimal Lorenz curve (= no inequality within each group).

Value

A list of class "Lc" with the following components:

p	vector of percentages
L	vector with values of the ordinary Lorenz curve
L.general	vector with values of the generalized Lorenz curve

Note

These functions were previously published as Lc() in the **ineq** package and have been integrated here without logical changes.

Author(s)

Achim Zeileis <Achim.Zeileis@R-project.org>, minor changes Andri Signorell <andri@signorell.net>

References

Arnold, B. C. (1987) *Majorization and the Lorenz Order: A Brief Introduction*, Springer

Cowell, F. A. (2000) Measurement of Inequality in Atkinson, A. B. / Bourguignon, F. (Eds): *Handbook of Income Distribution*. Amsterdam.

Cowell, F. A. (1995) *Measuring Inequality* Harvester Wheatshef: Prentice Hall.

See Also

The original location Lc(),
inequality measures Gini(), Atkinson()

Examples

```
data(d.pizza)
priceCarpenter <- d.pizza$price[d.pizza$driver=="Carpenter"]
priceMiller <- d.pizza$price[d.pizza$driver=="Miller"]

# compute the Lorenz curves
Lc.p <- Lc(priceCarpenter, na.rm=TRUE)
Lc.u <- Lc(priceMiller, na.rm=TRUE)
plot(Lc.p)
lines(Lc.u, col=2)

# the picture becomes even clearer with generalized Lorenz curves
plot(Lc.p, general=TRUE)
lines(Lc.u, general=TRUE, col=2)

# inequality measures emphasize these results, e.g. Atkinson's measure
Atkinson(priceCarpenter, na.rm=TRUE)
Atkinson(priceMiller, na.rm=TRUE)


# income distribution of the USA in 1968 (in 10 classes)
# x vector of class means, n vector of class frequencies
x <- c(541, 1463, 2445, 3438, 4437, 5401, 6392, 8304, 11904, 22261)
n <- c(482, 825, 722, 690, 661, 760, 745, 2140, 1911, 1024)

# compute minimal Lorenz curve (= no inequality in each group)
Lc.min <- Lc(x, n=n)
plot(Lc.min)


# input of frequency tables with midpoints of classes
fl <- c(2.5,7.5,15,35,75,150)   # midpoints
n  <- c(25,13,10,5,5,2)         # frequencies

plot( Lc(fl, n),                # Lorenz-Curve
      panel.first=grid(10, 10),
      main="Lorenzcurve Farmers",
      xlab="Percent farmers (cumulative)",
      ylab="Percent of area (%)"
)

Gini(fl, n)

# find specific function values using appprox
x <- c(1,1,4)
lx <- Lc(x)
plot(lx)

# get interpolated function value at p = 0.55
y0 <- approx(x=lx$p, y=lx$L, xout=0.55)
abline(v=0.55, h=y0$y, lty="dotted")

# and for the inverse question
y0 <- approx(x=lx$L, y=lx$p, xout=0.6)
abline(h=0.6, v=y0$y, col="red")
```

```
text(x=0.1, y=0.65, label=expression(L^{-1}*(0.6) == 0.8), col="red")
text(x=0.65, y=0.2, label=expression(L(0.55) == 0.275))
```

`LeveneTest` *Levene's Test for Homogeneity of Variance*

Description

Computes Levene's test for homogeneity of variance across groups.

Usage

```
LeveneTest(y, ...)

## S3 method for class 'formula'
LeveneTest(y, data, ...)
## S3 method for class 'lm'
LeveneTest(y, ...)
## Default S3 method:
LeveneTest(y, group, center=median, ...)
```

Arguments

`y`	response variable for the default method, or a `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`	factor defining groups.
`center`	The name of a function to compute the center of each group; `mean` gives the original Levene's test; the default, `median`, provides a more robust test (Brown-Forsythe-Test).
`data`	a data frame for evaluating the `formula`.
`...`	arguments to be passed down, e.g., `data` for the `formula` and `lm` methods; can also be used to pass arguments to the function given by `center` (e.g., `center=mean` and `trim=0.1` specify the 10% trimmed mean).

Value

returns an object meant to be printed showing the results of the test.

Note

This function was previously published as leveneTest() in the library(car) and has been integrated here without logical changes.

Author(s)

John Fox `<jfox@mcmaster.ca>`; original generic version contributed by Derek Ogle
adapted from a response posted by Brian Ripley to the r-help email list.

References

Fox, J. (2008) *Applied Regression Analysis and Generalized Linear Models*, Second Edition. Sage.

Fox, J. and Weisberg, S. (2011) *An R Companion to Applied Regression*, Second Edition, Sage.

See Also

`fligner.test` for a rank-based (nonparametric) k-sample test for homogeneity of variances; `mood.test` for another rank-based two-sample test for a difference in scale parameters; `var.test` and `bartlett.test` for parametric tests for the homogeneity in variance.

`ansari_test` in package coin for exact and approximate *conditional* p-values for the Ansari-Bradley test, as well as different methods for handling ties.

Examples

```
## example from ansari.test:
## Hollander & Wolfe (1973, p. 86f):
## Serum iron determination using Hyland control sera
ramsay <- c(111, 107, 100, 99, 102, 106, 109, 108, 104, 99,
            101, 96, 97, 102, 107, 113, 116, 113, 110, 98)
jung.parekh <- c(107, 108, 106, 98, 105, 103, 110, 105, 104,
            100, 96, 108, 103, 104, 114, 114, 113, 108, 106, 99)

LeveneTest( c(ramsay, jung.parekh),
  factor(c(rep("ramsay",length(ramsay)), rep("jung.parekh",length(jung.parekh)))))

LeveneTest( c(rnorm(10), rnorm(10, 0, 2)), factor(rep(c("A","B"),each=10)) )

## Not run:
# original example from package car

with(Moore, LeveneTest(conformity, fcategory))
with(Moore, LeveneTest(conformity, interaction(fcategory, partner.status)))

LeveneTest(conformity ~ fcategory * partner.status, data = Moore)
LeveneTest(conformity ~ fcategory * partner.status, data = Moore, center = mean)
LeveneTest(conformity ~ fcategory * partner.status, data = Moore, center = mean, trim = 0.1)

LeveneTest(lm(conformity ~ fcategory*partner.status, data = Moore))

## End(Not run)
```

`LillieTest` *Lilliefors (Kolmogorov-Smirnov) test for normality*

Description

Performs the Lilliefors (Kolmogorov-Smirnov) test for the composite hypothesis of normality, see e.g. Thode (2002, Sec. 5.1.1).

Usage

```
LillieTest(x)
```

Arguments

x	a numeric vector of data values, the number of which must be greater than 4. Missing values are allowed.

Details

The Lilliefors (Kolmogorov-Smirnov) test is an EDF omnibus test for the composite hypothesis of normality. The test statistic is the maximal absolute difference between empirical and hypothetical cumulative distribution function. It may be computed as $D = \max\{D^+, D^-\}$ with

$$D^+ = \max_{i=1,\ldots,n} \{i/n - p_{(i)}\}, D^- = \max_{i=1,\ldots,n} \{p_{(i)} - (i-1)/n\},$$

where $p_{(i)} = \Phi([x_{(i)} - \overline{x}]/s)$. Here, Φ is the cumulative distribution function of the standard normal distribution, and $\overline{x}$ and s are mean and standard deviation of the data values. The p-value is computed from the Dallal-Wilkinson (1986) formula, which is claimed to be only reliable when the p-value is smaller than 0.1. If the Dallal-Wilkinson p-value turns out to be greater than 0.1, then the p-value is computed from the distribution of the modified statistic $Z = D(\sqrt{n} - 0.01 + 0.85/\sqrt{n})$, see Stephens (1974), the actual p-value formula being obtained by a simulation and approximation process.

Value

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

statistic	the value of the Lilliefors (Kolomogorv-Smirnov) statistic.
p.value	the p-value for the test.
method	the character string "Lilliefors (Kolmogorov-Smirnov) normality test".
data.name	a character string giving the name(s) of the data.

Note

The Lilliefors (Kolomorov-Smirnov) test is the most famous EDF omnibus test for normality. Compared to the Anderson-Darling test and the Cramer-von Mises test it is known to perform worse. Although the test statistic obtained from `LillieTest(x)` is the same as that obtained from `ks.test(x, "pnorm", mean(x), sd(x))`, it is not correct to use the p-value from the latter for the composite hypothesis of normality (mean and variance unknown), since the distribution of the test statistic is different when the parameters are estimated.

The function call `LillieTest(x)` essentially produces the same result as the S-PLUS function call `ks.gof(x)` with the distinction that the p-value is not set to 0.5 when the Dallal-Wilkinson approximation yields a p-value greater than 0.1. (Actually, the alternative p-value approximation is provided for the complete range of test statistic values, but is only used when the Dallal-Wilkinson approximation fails.)

Author(s)

Juergen Gross <gross@statistik.uni-dortmund.de>

References

Dallal, G.E. and Wilkinson, L. (1986) An analytic approximation to the distribution of Lilliefors' test for normality. *The American Statistician*, 40, 294–296.

Stephens, M.A. (1974) EDF statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association*, 69, 730–737.

Thode Jr., H.C. (2002) *Testing for Normality* Marcel Dekker, New York.

See Also

`shapiro.test` for performing the Shapiro-Wilk test for normality. `AndersonDarlingTest`, `CramerVonMisesTest`, `PearsonTest`, `ShapiroFranciaTest` for performing further tests for normality. `qqnorm` for producing a normal quantile-quantile plot.

Examples

```
LillieTest(rnorm(100, mean = 5, sd = 3))
LillieTest(runif(100, min = 2, max = 4))
```

LinScale *Perform a linear scaling of x*

Description

This will scale the numeric vector x linearly from an old scale between `low` and `high` to a new one between `newlow` and `newhigh`.

Usage

```
LinScale(x, low = NULL, high = NULL, newlow = 0, newhigh = 1)
```

Arguments

`x`	a numeric matrix(like object).
`low`	numeric. The minimum value of the scale, defaults to min(x). This is calculated columnwise by default; defined `low` or `high` arguments will be recycled if necessary.
`high`	numeric. The maximum value of the scale, defaults to max(x). This is calculated columnwise by default; when a maxval is entered, it will be recycled.
`newlow`	numeric. The minimum value of the new scale, defaults to 0, resulting in a 0-1 scale for x. `newlow` is recycled if necessary.
`newhigh`	numeric. The maximum value of the scale, defaults to 1. `newhigh` is recycled if necessary.

Details

Hmm, hardly worth coding...

Value

The centered and scaled matrix. The numeric centering and scalings used (if any) are returned as attributes "scaled:center" and "scaled:scale"

Author(s)

Andri Signorell <andri@signorell.net>

See Also

scale, RobScale, sweep

Examples

```
# transform the temperature from Celsius to Fahrenheit
LinScale(d.pizza[1:20, "temperature"], 0, 100, -17.8, 37.8 )

# and the price from Dollar to Euro
LinScale(d.pizza[1:20, "price"], 0, 1, 0, 0.76)

# together
LinScale(d.pizza[1:20, c("temperature", "price")],
  0, c(100, 1), c(-17.8, 0), c(37.8, 0.76) )


## Not run:

par(mfrow=c(3,1), mar=c(0,5,0,3), oma=c(5,0,5,0))
plot(LinScale(d.frm[,1]), ylim=c(-2,2), xaxt="n", ylab="LinScale")
plot(RobScale(d.frm[,1]), ylim=c(-2,2), xaxt="n", ylab="RobScale")
plot(scale(d.frm[,1]), ylim=c(-2,2), ylab="scale")
title("Compare scales", outer = TRUE)


## End(Not run)
```

LOCF *Last Observation Carried Forward*

Description

Replace NAs by the last observed value (aka "Last Observation Carried Forward").

Usage

```
LOCF(x)
```

Arguments

x	x is a vector containing NAs.

Value

a vector with the same dimension as x.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

See also the package **Hmisc** for less coarse imputation functions.

Examples

```
d.frm <- data.frame(
  tag=rep(c("mo", "di", "mi", "do", "fr", "sa", "so"), 4)
, val=rep(c(runif(5), rep(NA,2)), 4) )

d.frm$locf <- LOCF( d.frm$val )
d.frm
```

Logit *Generalized Logit and Inverse Logit function*

Description

Compute generalized logit and generalized inverse logit functions.

Usage

```
Logit(x, min = 0, max = 1)
LogitInv(x, min = 0, max = 1)
```

Arguments

x	value(s) to be transformed
min	Lower end of logit interval
max	Upper end of logit interval

Details

The generalized logit function takes values on [min, max] and transforms them to span [-Inf,Inf] it is defined as:

$$y = log(\frac{p}{(1-p)})$$

where

$$p = \frac{(x - min)}{(max - min)}$$

The generized inverse logit function provides the inverse transformation:

$$x = p'(max - min) + min$$

where

$$p' = \frac{exp(y)}{(1 + exp(y))}$$

Value

Transformed value(s).

Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

logit

Examples

```
x <- seq(0,10, by=0.25)
xt <- Logit(x, min=0, max=10)
cbind(x,xt)

y <- LogitInv(xt, min=0, max=10)
cbind(x,xt,y)
```

LogLin *Log Linear Hybrid, Generalized Log*

Description

Computes the log linear hybrid transformation, resp. generalized log, with the goal to stabilize the variance.

Usage

```
LogLin(x, a)
LogGen(x, a)
```

Arguments

x	a numeric vector, matrix or data frame.
a	cutoff for the linear part of the transformation

Details

The log linear hybrid transformation function is linear for x <= a and logarithmic for x > a. It is continuously differentiable. The generalized log and log-linear hybrid transformations were introduced in then context of gen-expression microarray data by Rocke and Durbin (2003).

The function LogLin is currently defined as:

```
function (x, a) {\cr
  x[x<=a] <- x[x<=a] / a + log(a) - 1\cr
  x[x>a] <- log(x[x>a])\cr
  return(x)\cr
}
```

and LogGen as:

```
function (x, a) {\cr
 return(log((x + sqrt(x^2 + a^2)) / 2))
}
```

Value

a numeric vector of the same dimensions as x containing the transformed results.

Author(s)

Andri Signorell <andri@signorell.net>

References

Rocke DM, Durbin B (2003): Approximate variance-stabilizing transformations for gene-expression microarray data, *Bioinformatics*. 22;19(8):966-72.

See Also

log

Examples

```
x <- seq(-10, 50, 0.1 )

plot( LogLin(x, a=5) ~ x, type="l" )
grid()
lines( LogLin(x, a=2) ~ x, col="brown" )
lines( LogLin(x, a=0.5) ~ x, col="steelblue" )

lines( LogGen(x, a=1) ~ x, col="orange" )
```

LogSt *Started Logarithmic Transformation and It's Inverse*

Description

Transforms the data by a log10 transformation, modifying small and zero observations such that the transformation yields finite values.

Usage

```
LogSt(x, calib = x, threshold = NULL, mult = 1)

LogStInv(x, threshold = NULL)
```

Arguments

x	a vector or matrix of data, which is to be transformed
calib	a vector or matrix of data used to calibrate the transformation(s), i.e., to determine the constant c needed
threshold	constant c that determines the transformation. The inverse function will look for an attribute named "threshold" if the argument is set to NULL.
mult	a tuning constant affecting the transformation of small values, see Details

Details

Small values are determined by the threshold c. If not given by the argument threshold, then it is determined by the quartiles q1 and q3 of the non-zero data as those smaller than c = q1=(q3=q1)mult. The rationale is that for lognormal data, this constant identifies 2 percent of the data as small. Beyond this limit, the transformation continues linear with the derivative of the log curve at this point. See code for the formula.
Another possible value for the threshold c was: median(x) / (median(x)/quantile(x, 0.25))^2.9)

The function chooses log10 rather than natural logs because they can be backtransformed relatively easily in the mind.

Value

the transformed data. The value c needed for the transformation is returned as `attr(.,"threshold")`.

Note

The names of the function alludes to Tukey's idea of "started logs".

Author(s)

Werner A. Stahel, ETH Zurich

See Also

LogLin

Examples

```
dd <- c(seq(0,1,0.1), 5 * 10^rnorm(100, 0, 0.2))
dd <- sort(dd)
r.dl <- LogSt(dd)
plot(dd, r.dl, type="l")
abline(v=attr(r.dl, "threshold"), lty=2)

x <- rchisq(df=3, n=100)
# should give 0 (or at least something small):
LogStInv(LogSt(x)) - x
```

LsFct *List Functions of a Package*

Description

List all the functions of a package.

Usage

```
LsFct(package)
```

Arguments

package	the name of the package

Details

This is just a wrapper for `ls` with the appropriate arguments (as I always forgot how to do the trick..).

Author(s)

Andri Signorell <andri@signorell.net>

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

`ls`

Examples

```
LsFct("DescTools")
```

Mbind *Bind k nxm-matrices with the same dimension*

Description

`ftable` is nice to produce flat tables. But it does accept nothing but a n-dim table (resp. array) as argument.
So `Mbind` binds two (or n) r x c matrices to one 3-dimensional n x r x c table(array), which can be passed to ftable to produce flat tables.

Usage

```
Mbind(...)
```

Arguments

`...`	a list of 2 or more matrices of the same n x m Dimension.

Value

a 3dim array of the same class as the input matrices. If there are several classes, the matrices will be coerced following the usual R-rules.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`CatTable`, `matrix`

Examples

```
data(d.pizza)

m1 <- do.call("rbind", lapply( d.pizza[,c("delivery_min","temperature")],
  tapply, d.pizza$area, mean, na.rm=TRUE))
names(dimnames(m1)) <- c("vname","area")

m2 <- do.call("rbind", lapply( d.pizza[,c("delivery_min","temperature")],
  tapply, d.pizza$area, median, na.rm=TRUE))
names(dimnames(m2)) <- c("vname","area")

m3 <- do.call("rbind", lapply( d.pizza[,c("delivery_min","temperature")],
  tapply, d.pizza$area, sd, na.rm=TRUE))
names(dimnames(m3)) <- c("vname","area")

m <- Mbind(mean=m1, median=m3, sd=m2)
m
class(m)

ftab <- round(ftable(m, col.vars=c("area")),2)
ftab
```

```
# two different classes
Mbind( alpha=matrix(letters[1:4], nrow=2), num=matrix(1:4, nrow=2))

# matrices with different dimensions are not allowed, following would raise an error:
# Mbind( matrix(letters[1:4], nrow=2), matrix(1:9, nrow=3))
```

MeanAD *Mean Absolute Deviation From a Center Point*

Description

Calculates the mean absolute deviation from a center point, typically the sample mean or the median.

Usage

```
MeanAD(x, FUN = mean, na.rm = FALSE)
```

Arguments

x	a vector containing the observations.
FUN	the name of a function to be used as center. Can as well be a self defined function. Default is mean.
na.rm	a logical value indicating whether or not missing values should be removed. Defaults to FALSE.

Details

The MeanAD function calculates the mean absolute deviation from the mean value (or from another supplied center point) of x, after having removed NA values (if requested).
It exists primarily to simplify the discussion of descriptive statistics during an introductory stats class.

Value

Numeric value.

Author(s)

Andri Signorell <andri@signorell.net> following an idea of Daniel Navarro (aad in the **lsr** package)

See Also

mad

Examples

```
x <- runif(100)
MeanAD(x)

speed <- c(58, 88, 40, 60, 72, 66, 80, 48, NA)
MeanAD(speed)
MeanAD(speed, na.rm=TRUE)
```

MeanCI *Confidence Interval for the Mean*

Description

Calculates the confidence interval for the mean either the classical way or with the bootstrap approach.

Usage

```
MeanCI(x, trim = 0, type = c("classic", "norm", "basic", "stud", "perc", "bca"),
       conf.level = 0.95, na.rm = FALSE, R = 999)
```

Arguments

x	a (non-empty) numeric vector of data values.
trim	a (non-empty) numeric vector of data values.
type	A vector of character strings representing the type of intervals required. The value should be any subset of the values "norm", "basic", "stud", "perc", "bca". See boot.ci.
conf.level	confidence level of the interval.
na.rm	logical. Should missing values be removed? Defaults to FALSE.
R	The number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case R would be a vector of integers where each component gives the number of resamples from each of the rows of weights. See boot.

Details

Interfaces for data.frames are widely deprecated nowadays and so we abstained to implement one. Use do.call, rbind and lapply for getting a matrix with estimates and confidence intervals for more than 1 column. (See examples!)

Value

a numeric vector with 3 elements:

mean	mean
lwr.ci	lower bound of the confidence interval
upr.ci	upper bound of the confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

See Also

MeanDiffCI, MedianCI, VarCI

Examples

```
x <- d.pizza$price[1:20]

MeanCI(x, na.rm=TRUE)
MeanCI(x, conf.level=0.99, na.rm=TRUE)

# the different types of bootstrap confints
MeanCI(x, type="norm", na.rm=TRUE)
MeanCI(x, type="norm", trim=0.1, na.rm=TRUE)
MeanCI(x, type="basic", trim=0.1, na.rm=TRUE)
# MeanCI(x, type="stud", trim=0.1, na.rm=TRUE), this needs some more information
MeanCI(x, type="perc", trim=0.1, na.rm=TRUE)
MeanCI(x, type="bca", trim=0.1, na.rm=TRUE)


# Getting the MeanCI for more than 1 column
round( do.call("rbind", lapply(d.pizza[,1:4],  MeanCI, na.rm=TRUE)), 3)
```

MeanDiffCI *Confidence Intervals for Difference of Means*

Description

Calculates the confidence interval for the difference of two means either the classical way or with the bootstrap approach.

Usage

```
MeanDiffCI(x, ...)

## Default S3 method:
MeanDiffCI(x, y, type = c("classic", "norm", "basic", "stud", "perc", "bca"),
           conf.level = 0.95, na.rm = FALSE, R = 999, ...)

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

Arguments

x	a (non-empty) numeric vector of data values.
y	a (non-empty) numeric vector of data values.

type	a vector of character strings representing the type of intervals required. The value should be any subset of the values "norm", "basic", "stud", "perc", "bca". See boot.ci.
conf.level	confidence level of the interval.
na.rm	logical. Should missing values be removed? Defaults to FALSE.
R	the number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case R would be a vector of integers where each component gives the number of resamples from each of the rows of weights. See boot.
formula	a formula of the form lhs ~ rhs where lhs is a numeric variable giving the data values and rhs a factor with two levels giving the corresponding groups.
data	an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
...	further argument to be passed to or from methods.

Details

This function collects code from two sources. The classical confidence interval is calculated by means of t.test. The bootstrap intervals are strongly based on the example in boot.

Value

a numeric vector with 3 elements:

meandiff	the difference: mean(x) - mean(y)
lwr.ci	lower bound of the confidence interval
upr.ci	upper bound of the confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

See Also

MeanCI, VarCI, MedianCI, boot.ci

Examples

```
x <- d.pizza$price[d.pizza$driver=="Carter"]
y <- d.pizza$price[d.pizza$driver=="Miller"]

MeanDiffCI(x, y, na.rm=TRUE)
MeanDiffCI(x, y, conf.level=0.99, na.rm=TRUE)

# the different types of bootstrap confints
MeanDiffCI(x, y, type="norm", na.rm=TRUE)
MeanDiffCI(x, y, type="basic", na.rm=TRUE)
```

```
# MeanDiffCI(x, y, type="stud", na.rm=TRUE)
MeanDiffCI(x, y, type="perc", na.rm=TRUE)
MeanDiffCI(x, y, type="bca", na.rm=TRUE)

# the formula interface
MeanDiffCI(price ~ driver, data=d.pizza, subset=driver %in% c("Carter","Miller"))
```

MeanSE *Standard error of mean*

Description

Calculates the standard error of mean. If x is a matrix or a data frame, a vector of the standard error of the columns is returned.

Usage

```
MeanSE(x, na.rm = FALSE)
```

Arguments

x	a (non-empty) numeric vector of data values.
na.rm	logical. Should missing values be removed? Defaults to FALSE.

Details

MeanSE calculates the standard error of the mean defined as sd(x)/sqrt(length(x)).

Value

For a data frame or for a matrix, a named vector with the appropriate method being applied column by column.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

MeanCI

Examples

```
data(d.pizza)

MeanSE(d.pizza$price, na.rm=TRUE)

# evaluate data.frame
sapply(d.pizza[,1:4], MeanSE, na.rm=TRUE)
```

`median.factor` *Median for Ordered Factors*

Description

Calculate the median for ordered factors. This is not implemented in standard R, as it's not well defined (it is not clear what to do if the median sits between two levels in factors of even length). This function returns the high median and prints a warning if the low median would be different (which is supposed to be a rare event).

Usage

```
## S3 method for class 'factor'
median(x, na.rm = FALSE)
```

Arguments

`x`	an ordered factor containing the values whose median is to be computed.
`na.rm`	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

There's a vivid discussion between experts going on whether this should be defined or not. We'll wait for definitive results and enjoy the function's comfort so far...

Value

a level of the ordered factor.

Author(s)

Andri Signorell <andri@signorell.net>, based on code from Hong Ooi

See Also

https://stat.ethz.ch/pipermail/r-help/2003-November/042684.html

http://www.rqna.net/qna/nuiukm-idiomatic-method-of-finding-the-median-of-an-ordinal-in-r.html

Examples

```
median(d.pizza$quality, na.rm=TRUE)
```

MedianCI *Confidence Interval for the Median*

Description

Calculates the confidence interval for the median.

Usage

```
MedianCI(x, conf.level = 0.95, na.rm = FALSE,
         type = c("pseudo", "exact", "boot"), R = 1000)
```

Arguments

x	a (non-empty) numeric vector of data values.
conf.level	confidence level of the interval
na.rm	logical. Should missing values be removed? Defaults to FALSE.
type	defining the type of interval that should be calculated. Default is "pseudo". See Details.
R	The number of bootstrap replicates. Usually this will be a single positive integer. See boot.ci for details.

Details

The the confidence interval for the "pseudo median"" is extracted from wilcox.test (conf.int = TRUE). (Would you have expected it there?)
The exact type is the way SAS is said to calculate the confidence interval. This is implemented in SignTest and is extracted from there. The boot confidence interval type is calculated by means of boot.ci with type "basic".
Use sapply, resp.apply, to get the confidence intervals from a data.frame or from a matrix.

Value

a numeric vector with 3 elements:

median	median
lwr.ci	lower bound of the confidence interval
upr.ci	upper bound of the confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

See Also

wilcox.test, MeanCI, median

Examples

```
data(d.pizza)

MedianCI(d.pizza$price, na.rm=TRUE)
MedianCI(d.pizza$price, conf.level=0.99, na.rm=TRUE)

t(round(sapply(d.pizza[,c("delivery_min","temperature","price")], MedianCI, na.rm=TRUE), 3))


MedianCI(d.pizza$price, na.rm=TRUE, type="exact")
MedianCI(d.pizza$price, na.rm=TRUE, type="boot")
```

MHChisqTest *Mantel-Haenszel Chi-Square Test*

Description

The Mantel-Haenszel chi-square statistic tests the alternative hypothesis that there is a linear association between the row variable and the column variable. Both variables must lie on an ordinal scale.

Usage

```
MHChisqTest(x, srow = 1:nrow(x), scol = 1:ncol(x))
```

Arguments

x	a frequency table or a matrix.
srow	scores for the row variable, defaults to 1:nrow.
scol	scores for the columnn variable, defaults to 1:ncol.

Details

The statistic is computed as Q_MH = (n-1)r^2 , where r^2 is the Pearson correlation between the row variable and the column variable. The Mantel-Haenszel chi-square statistic use the scores specified by srow and scol. Under the null hypothesis of no association, Q_MH has an asymptotic chi-square distribution with one degree of freedom.

Value

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

statistic	the value the Mantel-Haenszel 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.
data.name	a character string giving the name(s) of the data.

Author(s)

Andri Signorell <andri@signorell.net>

References

Agresti, A. (2002) *Categorical Data Analysis*. John Wiley & Sons, pp 86 ff.

See Also

`chisq.test`, for calculating correlation of a table: `corr`

Examples

```
## A r x c table  Agresti (2002, p. 57) Job Satisfaction
Job <- matrix(c(1,2,1,0, 3,3,6,1, 10,10,14,9, 6,7,12,11), 4, 4,
              dimnames = list(income = c("< 15k", "15-25k", "25-40k", "> 40k"),
                              satisfaction = c("VeryD", "LittleD", "ModerateS", "VeryS"))
       )

MHChisqTest(Job, srow=c(7.5,20,32.5,60))
```

Midx *Find the Midpoints of a Numeric Vector*

Description

Calculate the midpoints of a sequence of numbers.

Usage

```
Midx(x, first = NULL)
```

Arguments

`x`	the numeric vector
`first`	number which is to be inserted in the very beginning of the vector. Default is NULL, meaning nothing will be inserted.

Value

numeric vector with the calculated midpoins

Author(s)

Andri Signorell <andri@signorell.net>

Examples

```
x <- c(1,3,6,7)
Midx(x)
Midx(x, 0)
```

Mode *Mode*

Description

Calculates the mode, the most frequent value, of a variable x. This makes mostly sense for qualitative data.

Usage

```
Mode(x, na.rm = FALSE)
```

Arguments

x	a (non-empty) numeric vector of data values.
na.rm	logical. Should missing values be removed? Defaults to FALSE.

Value

Returns the most frequent value. If there are more than one, all of them are returned in a vector.

Note

Consider using `density(x)$x[which.max(density(x)$y)]` for quantitative data or alternatively use `hist()`.
Another interesting idea:

```
peak <- optimize(function(x, model) predict(model, data.frame(x = x)),
                 c(min(x), max(x)),
                 maximum = TRUE,
                 model = y.loess)

points(peak$maximum, peak$objective, pch=FILLED.CIRCLE <- 19)
```

Author(s)

Andri Signorell <andri@signorell.net>

See Also

mean, median

Examples

```
data(d.pizza)
Mode(d.pizza$driver)

# use sapply for evaluating data.frames (resp. apply for matrices)
sapply(d.pizza[,c("driver","temperature","date")], Mode, na.rm=TRUE)
```

MosesTest *Moses Test of Extreme Reactions*

Description

Perform Moses test of extreme reactions, which can be used to determine the difference in range between two samples. The exact one-tailed probability is calculated.

Usage

```
MosesTest(x, ...)

## Default S3 method:
MosesTest(x, y, extreme = NULL, ...)

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

Arguments

x	numeric vector of data values. x will be treated as control group. Non-finite (e.g. infinite or missing) values will be omitted.
y	numeric vector of data values. y will be treated as experiment group. Non-finite (e.g. infinite or missing) values will be omitted.
formula	a formula of the form `lhs ~ rhs` where `lhs` gives the data values and rhs the corresponding groups.
data	an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to `getOption("na.action")`.
extreme	integer, defines the number of extreme values to be dropped from the control group before calculating the span. Default (NULL) is the integer part of 0.05 * length(x) or 1, whichever is greater. If extreme is too large, it will be cut down to floor(length(x)-2)/2.
...	further arguments to be passed to or from methods.

Details

For two independent samples from a continuous field, this tests whether extreme values are equally likely in both populations or if they are more likely to occur in the population from which the sample with the larger range was drawn.

Note that the ranks are calculated in decreasing mode.

Value

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

statistic	the value of the Moses Test statistic.
p.value	the p-value for the test.
method	the character string "Moses Test of Extreme Reactions".
data.name	a character string giving the name(s) of the data.

Author(s)

Andri Signorell <andri@signorell.net>

References

Moses, L.E. (1952) A Two-Sample Test, *Psychometrika*, 17, 239-247.

http://publib.boulder.ibm.com/infocenter/spssstat/v20r0m0/index.jsp?topic=%2Fcom.ibm.spss.statistics.help%2Falg_nonparametric_independent_moses.htm

See Also

wilcox.test, ks.test

Examples

```
x <- c(0.80, 0.83, 1.89, 1.04, 1.45, 1.38, 1.91, 1.64, 0.73, 1.46)
y <- c(1.15, 0.88, 0.90, 0.74, 1.21)

MosesTest(x, y)


set.seed(1479)
x <- sample(1:20, 10, replace=TRUE)
y <- sample(5:25, 6, replace=TRUE)

MosesTest(x, y)
```

MoveAvg *Moving Average*

Description

Compute a simple moving average (running mean).

Usage

```
MoveAvg(x, order, align = c("center", "left", "right"))
```

Arguments

x	univariate time series.
order	order of moving average.
align	specifies whether result should be centered (default), left-aligned or right-aligned.

Details

The implementation is using the function `filter` to calculate the moving average.

Value

Returns a numeric vector of the same size as x.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

There's a faster implementation of running mean in the package **caTools** `runmean()` and a slower one in **forecast** `ma()`.

Examples

```
MoveAvg(AirPassengers, order=5)
```

Mround *Round to Multiple*

Description

Returns a number rounded to the desired multiple.

Usage

```
Mround(x, multiple)
```

Arguments

x	numeric. The value to round.
multiple	numeric. The multiple to which the number is to be rounded.

Details

Mround rounds up, away from zero, if the remainder of dividing number by multiple is greater than or equal to half the value of multiple.

Value

the rounded value,

Author(s)

Andri Signorell <andri@signorell.net>

See Also

round

Examples

```
Mround(10, 3)     # Rounds 10 to a nearest multiple of 3 (9)
Mround(-10, -3)   # Rounds -10 to a nearest multiple of -3 (-9)

Mround(1.3, 0.2)  # Rounds 1.3 to a nearest multiple of 0.2 (1.4)
Mround(5, -2)     # Returns an error, because -2 and 5 have different signs

Mround(c(1.92, 45.38, 0.831125), 0.05)
```

MultinomCI *Confidence Intervals for Multinomial Proportions*

Description

Calculate simultaneous confidence intervals for multinomial proportions either according to the method of Sison and Glaz or according to Goodman's method.

Usage

```
MultinomCI(x, conf.level = 0.95, method = c("sisonglaz", "cplus1", "goodman"))
```

Arguments

x	A vector of positive integers representing the number of occurrences of each class. The total number of samples equals the sum of such elements.
conf.level	confidence level, defaults to 0.95.
method	character string specifing which method to use; can be one out of "sisonglaz", "cplus1", "goodman". Method can be abbreviated. See details. Defaults to "sisonglaz".

Details

Given a vector of observations with the number of samples falling in each class of a multinomial distribution, builds the simultaneous confidence intervals for the multinomial probabilities according to the method proposed by Sison and Glaz (1995). The R code has been translated from the SAS code written by May and Johnson (2000).

Value

A matrix with 3 elements columns for estimate, lower confidence intervall and upper for the upper one. The rows correspond to the dimension of x.

Author(s)

Pablo J. Villacorta Iglesias <pjvi@decsai.ugr.es> Department of Computer Science and Artificial Intelligence, University of Granada (Spain) (Sison-Glaz)

Andri Signorell <andri@signorell.net> (Goodman)

References

Sison, C.P and Glaz, J. (1995): Simultaneous confidence intervals and sample size determination for multinomial proportions. *Journal of the American Statistical Association*, 90:366-369.

http://tx.liberal.ntu.edu.tw/~purplewoo/Literature/!Methodology/!Distribution_SampleSize/SimultConfidIntervJASA.pdf

Glaz, J., Sison, C.P. (1999): Simultaneous confidence intervals for multinomial proportions. *Journal of Statistical Planning and Inference* 82:251-262.

May, W.L., Johnson, W.D.(2000): Constructing two-sided simultaneous confidence intervals for multinomial proportions for small counts in a large number of cells. *Journal of Statistical Software* 5(6) . Paper and code available at http://www.jstatsoft.org/v05/i06.

Examples

```
# Multinomial distribution with 3 classes, from which 79 samples
# were drawn: 23 of them belong to the first class, 12 to the
# second class and 44 to the third class. Punctual estimations
# of the probabilities from this sample would be 23/79, 12/79
# and 44/79 but we want to build 95% simultaneous confidence intervals
# for the true probabilities

MultinomCI(c(23,12,44), conf.level=0.95)


x <- c(35,74,22,69)

MultinomCI(x, method="goodman")
MultinomCI(x, method="sisonglaz")
MultinomCI(x, method="cplus1")

# compare to
BinomCI(x, n=sum(x))
```

Ndec *Count Decimal Places of a Number*

Description

Returns the number of decimal places in the vector x. x must be a character and contain formatted numbers.

Usage

```
Ndec(x)
```

Arguments

x	is a character vector containing formatted numbers

Details

The function is currently defined as:

```
Ndec <- function(x) {
  stopifnot(class(x)=="character")
  res <- rep(0, length(x))
  x <- gsub(pattern="[eE].+$", rep="", x=x)
  res[grep("\.",x)] <- nchar( sub("^.+[.]","",x) )[grep("\.",x)]
  return(res)
}
```

Value

an integer value.

Note

format.info
[1] ... Breite
[2] ... Anzahl Nachkommastellen
[3] ... Exponential ja/nein

Author(s)

Andri Signorell <andri@signorell.net>

See Also

format.info, Frac

Examples

```
x <- c("0.0000", "0", "159.283", "1.45e+10", "1.4599E+10" )
Ndec(x)
```

`OddsRatio` *Odds Ratio Estimation and Confidence Intervals*

Description

Calculates odds ratio by unconditional maximum likelihood estimation (wald), conditional maximum likelihood estimation (mle) and median-unbiased estimation (midp). Confidence intervals are calculated using normal approximation (wald) and exact methods (midp, mle).

Usage

```
OddsRatio(x, y = NULL, conf.level = NA, method = c("wald", "mle", "midp"),
          interval = c(0, 1000), ...)
```

Arguments

`x`	a numeric vector or a 2x2 numeric matrix, resp. table.
`y`	NULL (default) or a vector with compatible dimensions to `x`. If y is provided, `table(x, y, ...)` is calculated.
`method`	method for calculating odds ratio and confidence interval. Can be one out of "wald", "mle", "midp". Default is "wald" (not because it is the best, but because it is the most commonly used.)
`conf.level`	confidence level. Default is NA, meaning no confidence intervals will be reported.
`interval`	interval for the `uniroot` that finds the odds ratio median-unbiased estimate and mid-p exact confidence interval.
`...`	further arguments are passed to the function `table`, allowing i.e. to set useNA. This refers only to the vector interface.

Details

If a 2x2 table is provided the following table structure is preferred:

```
                  disease=0   disease=1
exposed=0 (ref)     n00         n01
exposed=1           n10         n11
```

however, for odds ratios from 2x2 tables, the following table is equivalent:

```
                  disease=1   disease=0
exposed=1           n11         n10
exposed=0           n01         n00
```

If the table to be provided to this function is not in the preferred form, just use the function `Rev()` to "reverse" the table rows, -columns, or both.

If a data.frame is provided the odds ratios are calculated pairwise and returned as numeric square matrix with the dimension of ncol(data.frame).

Value

If conf.level is not NA then the result will be a vector with 3 elements for estimate, lower confidence intervall and upper for the upper one. Else the odds ratio will be reported as a single value.

Author(s)

Andri Signorell <andri@signorell.net>, strongly based on code from Tomas Aragon, <aragon@berkeley.edu>

References

Kenneth J. Rothman and Sander Greenland (1998): *Modern Epidemiology*, Lippincott-Raven Publishers

Kenneth J. Rothman (2002): *Epidemiology: An Introduction*, Oxford University Press

Nicolas P. Jewell (2004): *Statistics for Epidemiology*, 1st Edition, 2004, Chapman & Hall, pp. 73-81

See Also

RelRisk

Examples

```
# Case-control study assessing whether exposure to tap water
#   is associated with cryptosporidiosis among AIDS patients

tab <- matrix(c(2, 29, 35, 64, 12, 6), 3, 2, byrow=TRUE)
dimnames(tab) <- list("Tap water exposure" = c("Lowest", "Intermediate", "Highest"),
                      "Outcome" = c("Case", "Control"))
tab <- Rev(tab, direction="column")

OddsRatio(tab[1:2,])
OddsRatio(tab[c(1,3),])

OddsRatio(tab[1:2,], method="mle")
OddsRatio(tab[1:2,], method="midp")
OddsRatio(tab[1:2,], method="wald", conf.level=0.95)
```

Outlier *Outlier*

Description

Return outliers following the Tukey's boxplot definition.

Usage

```
Outlier(x, na.rm = FALSE, method = c("boxplot"))
```

Arguments

x	a (non-empty) numeric vector of data values.
na.rm	logical. Should missing values be removed? Defaults to FALSE.
method	the method to be used. So far only Tukey's boxplot rule is implemented.

Details

Outlier detection is a tricky problem and should be handled with care. We implement only Tukey's boxplot rule as a rough idea of spotting extreme values.

Value

the values of x lying outside the whiskers in a boxplot

Author(s)

Andri Signorell <andri@signorell.net>

See Also

boxplot

Examples

```
Outlier(d.pizza$temperature, na.rm=TRUE)

# find the corresponding rows
d.pizza[which(d.pizza$temperature %in% Outlier(d.pizza$temperature, na.rm=TRUE)),]


# outliers for the drivers
tapply(d.pizza$temperature, d.pizza$driver, Outlier, na.rm=TRUE)

# see also
boxplot(temperature ~ driver, d.pizza)$out
```

PageTest *Exact Page Test for Ordered Alternatives*

Description

Performs a Page test for ordered alternatives using an exact algorithm by Stefan Wellek (1989) with unreplicated blocked data.

Usage

```
PageTest(y, ...)

## Default S3 method:
PageTest(y, groups, blocks, ...)

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

Arguments

y	either a numeric vector of data values, or a data matrix.
groups	a vector giving the group for the corresponding elements of y if this is a vector; ignored if y is a matrix. If not a factor object, it is coerced to one.
blocks	a vector giving the block for the corresponding elements of y if this is a vector; ignored if y is a matrix. If not a factor object, it is coerced to one.
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 matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
...	further arguments to be passed to or from methods.

Details

PageTest can be used for analyzing unreplicated complete block designs (i.e., there is exactly one observation in y for each combination of levels of groups and blocks) where the normality assumption may be violated.

The null hypothesis is that apart from an effect of blocks, the location parameter of y is the same in each of the groups.
The implemented alternative is, that the location parameter will be monotonly greater along the groups,
$H_A : \theta_1 \leq \theta_2 \leq \theta_3$ ... (where at least one inequality is strict).
If the other direction is required, the order of the groups has to be reversed.

The Page test for ordered alternatives is slightly more powerful than the Friedman analysis of variance by ranks.

If y is a matrix, groups and blocks are obtained from the column and row indices, respectively. NA's are not allowed in groups or blocks; if y contains NA's, corresponding blocks are removed.

For small values of k (methods) or N (data objects), 'PageTest' will calculate the exact p-values. For 'k, N > 15, Inf', a normal approximation is returned. Only one of these values will be returned.

Value

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

statistic	the L-statistic with names attribute "L".
p.value	the p-value of the test.
method	the character string "Page test for ordered alternatives".
data.name	a character string giving the names of the data.

Note

Special thanks to Prof. S. Wellek for porting old GAUSS code to R.

Author(s)

Stefan Wellek <stefan.wellek@zi-mannheim.de> (exact p-values), Andri Signorell <andri@signorell.net> (interface) (strongly based on R-Core code)

References

Page, E. (1963): Ordered hypotheses for multiple treatments: A significance test for linear ranks. *Journal of the American Statistical Association*, 58, 216-230.

Siegel, S. & Castellan, N. J. Jr. (1988): *Nonparametric statistics for the behavioral sciences*. Boston, MA: McGraw-Hill.

Wellek, S. (1989): Computing exact p-values in Page's nonparametric test against trend. *Biometrie und Informatik in Medizin und Biologie 20*, 163-170

See Also

friedman.test

Examples

```
 # Craig's data from Siegel & Castellan, p 186
 soa.mat <- matrix(c(.797,.873,.888,.923,.942,.956,
  .794,.772,.908,.982,.946,.913,
  .838,.801,.853,.951,.883,.837,
  .815,.801,.747,.859,.887,.902), nrow=4, byrow=TRUE)
 PageTest(soa.mat)


# Duller, pg. 236
pers <- matrix(c(
1, 72, 72, 71.5, 69, 70, 69.5, 68, 68, 67, 68,
2, 83, 81, 81, 82, 82.5, 81, 79, 80.5, 80, 81,
3, 95, 92, 91.5, 89, 89, 90.5, 89, 89, 88, 88,
4, 71, 72, 71, 70.5, 70, 71, 71, 70, 69.5, 69,
5, 79, 79, 78.5, 77, 77.5, 78, 77.5, 76, 76.5, 76,
6, 80, 78.5, 78, 77, 77.5, 77, 76, 76, 75.5, 75.5
), nrow=6, byrow=TRUE)

colnames(pers) <- c("person", paste("week",1:10))

# Alternative: week10 < week9 < week8 ...
PageTest(pers[, 11:2])


# Sachs, pg. 464
```

```
pers <- matrix(c(
  3,2,1,4,
  4,2,3,1,
  4,1,2,3,
  4,2,3,1,
  3,2,1,4,
  4,1,2,3,
  4,3,2,1,
  3,1,2,4,
  3,1,4,2),
  nrow=9, byrow=TRUE, dimnames=list(1:9, LETTERS[1:4]))

# Alternative: B < C < D < A
PageTest(pers[, c("B","C","D","A")])


# long shape and formula interface
plng <- data.frame(expand.grid(1:9, c("B","C","D","A")),
                   as.vector(pers[, c("B","C","D","A")]))
colnames(plng) <- c("block","group","x")

PageTest(plng$x, plng$group, plng$block)

PageTest(x ~ group | block, data = plng)



score <- matrix(c(
  3,4,6,9,
  4,3,7,8,
  3,4,4,6,
  5,6,8,9,
  4,4,9,9,
  6,7,11,10
  ), nrow=6, byrow=TRUE)

PageTest(score)
```

PairApply *Pairwise Calculations*

Description

Implements a logic to run pairwise calculations on the columns of a data.frame or a matrix.

Usage

```
PairApply(x, FUN = NULL, ..., symmetric = TRUE)
```

Arguments

x	a list, a data.frame or a matrix with columns to be processed pairwise.

FUN	a function to be calculated. It is assumed, that the first 2 arguments denominate x and y.
...	the dots are passed to FUN.
symmetric	logical. Does the function yield the same result for FUN(x, y) and FUN(y, x)? If TRUE just the lower triangular matrix is calculated and transposed.

Details

This code is based on the logic of `cor()` and extended for asymmetric functions.

Value

a matrix with the results of FUN.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`outer`, `GetPairs`

Examples

```
PairApply(d.diamonds[,c("colour","clarity","cut","polish")], FUN = CramerV)

# user defined functions are ok as well
PairApply(d.diamonds[,c("clarity","cut","polish","symmetry")],
  FUN = function(x,y) wilcox.test(as.numeric(x), as.numeric(y))$p.value)

# asymetric measure
PairApply(d.diamonds[,c("colour", "clarity", "cut", "polish")],
  FUN = Lambda, direction = "row", symmetric = FALSE)

# ... compare to:
Lambda(x=d.diamonds$colour, y=d.diamonds$clarity, direction="row")
Lambda(x=d.diamonds$colour, y=d.diamonds$clarity, direction="column")


# the data.frame
dfrm <- d.diamonds[, c("colour","clarity","cut","polish")]
PairApply(dfrm, FUN = CramerV)

# the same as matrix (columnwise)
m <- as.matrix(dfrm)
PairApply(m, FUN = CramerV)

# ... and the list interface
lst <- as.list(dfrm)
PairApply(lst, FUN = CramerV)
```

PalDescTools *Some Custom Palettes*

Description

Some more custom palettes.

Usage

```
PalDescTools(pal, n = 100)
```

Arguments

pal	name or number of the palette. One of RedToBlack (1), RedBlackGreen (2), SteeblueWhite (3), RedWhiteGreen (4), RedWhiteBlue1 (5), RedWhiteBlue2 (6), Helsana (7), Tibco (8)
n	integer, number of colors for the palette.

Value

a vector of colors

Author(s)

Andri Signorell <andri@signorell.net>

See Also

colorRampPalette

Examples

```
Canvas(c(0,1))
ColorLegend(x=0, y=1, width=0.1, col=PalDescTools(1, n=50))
ColorLegend(x=0.15, y=1, width=0.1, col=PalDescTools(2, n=50))
ColorLegend(x=0.3, y=1, width=0.1, col=PalDescTools(3, n=50))
ColorLegend(x=0.45, y=1, width=0.1, col=PalDescTools(4, n=50))
ColorLegend(x=0.6, y=1, width=0.1, col=PalDescTools(5, n=50))
ColorLegend(x=0.75, y=1, width=0.1, col=PalDescTools(6, n=50))
ColorLegend(x=0.9, y=1, width=0.1, col=PalDescTools(7))
ColorLegend(x=1.05, y=1, width=0.1, col=PalDescTools(8))

text(1:8, y=1.05, x=seq(0,1.05,.15)+.05)
title(main="DescTools palettes")
```

PalTibco *Some More Color Palettes*

Description

Defines some more color palettes.

Usage

```
PalTibco()
PalHelsana()
PalRedToBlack(n = 100)
PalRedBlackGreen(n = 100)
PalSteeblueWhite(n = 100)
PalRedWhiteGreen(n = 100)

hblue
hred
```

Arguments

n	the number of colors (= 1) to be in the palette.

Details

hblue and hred are 2 constants, pointing to the red and blue from the palette PalHelsana.

Value

A vector of color values in hexform.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

SetAlpha, HexToCol, ColToHex

Examples

```
par(mfrow=c(4,2), mar=c(1,1,2,1))
barplot(1:9, col=PalTibco(), axes=FALSE, main="PalTibco" )

barplot(1:7, col=PalHelsana(), axes=FALSE, main="PalHelsana" )
barplot(1:7, col=SetAlpha(PalHelsana()[c("ecru","hellgruen","hellblau")], 0.6),
        axes=FALSE, main="PalHelsana (Alpha)" )

barplot(1:10, col=PalRedToBlack(10), axes=FALSE, main="PalRedToBlack" )
barplot(1:10, col=PalRedBlackGreen(10), axes=FALSE, main="PalRedGreenGreen" )
barplot(1:10, col=PalSteeblueWhite(10), axes=FALSE, main="PalSteeblueWhite" )
```

```
barplot(1:10, col=PalRedWhiteGreen(10), axes=FALSE, main="PalRedWhiteGreen" )
```

ParseFormula *Parse a Formula and Create a Model Frame*

Description

Create a model frame for a formula object, by handling the left hand side the same way the right hand side is handled in model.frame. Especially variables separated by + are interpreted as separate variables.

Usage

```
ParseFormula(formula, data = parent.frame(), subset = TRUE, drop = TRUE)
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description for the variables to be described.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.
subset	an optional vector specifying a subset of observations to be used.
drop	if drop is TRUE, unused factor levels are dropped from the result when creating interaction terms. The default is to drop all unused factor levels.

Details

This is used by Desc.formula for describing data by groups while remaining flexible for using I(...) constructions, functions or interaction terms.

Value

a list of 3 elements

formula	the formula which had to be parsed
lhs	a list of 3 elements: mf: data.frame, the model.frame of the left hand side of the formula mf.eval: data.frame, the evaluated model.frame of the left hand side of the formula vars: the names of the evaluated model.frame
rhs	a list of 3 elements: mf: data.frame, the model.frame of the right hand side of the formula mf.eval: data.frame, the evaluated model.frame of the right hand side of the formula vars: the names of the evaluated model.frame

Author(s)

Andri Signorell <andri@signorell.net>

See Also

The functions used to handle formulas: model.frame, terms, formula
Used in: Desc.formula

Examples

```
set.seed(17)
piz <- d.pizza[sample(nrow(d.pizza),10), c("temperature","price","driver","weekday")]

f1 <- formula(. ~ driver)
f2 <- formula(temperature ~ .)
f3 <- formula(temperature + price ~ .)
f4 <- formula(temperature ~ . - driver)
f5 <- formula(temperature + price ~ driver)
f6 <- formula(temperature + price ~ driver * weekday)
f7 <- formula(I(temperature^2) + sqrt(price) ~ driver + weekday)
f8 <- formula(temperature + price ~ 1)
f9 <- formula(temperature + price ~ driver * weekday - price)

ParseFormula(f1, data=piz)
ParseFormula(f2, data=piz)
ParseFormula(f3, data=piz)
ParseFormula(f4, data=piz)
ParseFormula(f5, data=piz)
ParseFormula(f6, data=piz)
ParseFormula(f7, data=piz)
ParseFormula(f8, data=piz)
```

Partial *Find the Correlations for a Set x of Variables With Set y Removed*

Description

A straightforward application of matrix algebra to remove the effect of the variables in the y set from the x set. Input may be either a data matrix or a correlation matrix. Variables in x and y are specified by location.

Usage

```
Partial(m, x, y)
```

Arguments

m	a data or correlation matrix.
x	the variable numbers associated with the X set.
y	the variable numbers associated with the Y set.

Details

It is sometimes convenient to partial the effect of a number of variables (e.g., sex, age, education) out of the correlations of another set of variables. This could be done laboriously by finding the residuals of various multiple correlations, and then correlating these residuals. The matrix algebra alternative is to do it directly.

Value

The matrix of partial correlations.

Author(s)

William Revelle

References

Revelle, W. *An introduction to psychometric theory with applications in R* Springer.
(working draft available at http://personality-project.org/r/book/

Examples

```
# example from Bortz, J. (1993) Statistik fuer Sozialwissenschaftler, Springer, pp. 413

abstr <- c(9,11,13,13,14,9,10,11,10,8,13,7,9,13,14)
coord <- c(8,12,14,13,14,8,9,12,8,9,14,7,10,12,12)
age <- c(6,8,9,9,10,7,8,9,8,7,10,6,10,10,9)

# calculate the correlation of abstr and coord, after without the effect of the age
Partial(cbind(abstr, coord, age), 1:2, 3)

# by correlation matrix m
m <- cor(cbind(abstr, coord, age))
Partial(m, 1:2, 3)

# ... which would be the same as:
lm1 <- lm(abstr ~ age)
lm2 <- lm(coord ~ age)

cor(resid(lm1), resid(lm2))
```

PasswordDlg *Password Dialog*

Description

A tcltk dialog for entering passwords with ****.

Usage

```
PasswordDlg()
```

Value

the entered password characters

Author(s)

Markus Naepflin <markus@naepfl.in>

See Also

ImportDlg

Examples

```
## Not run:
PasswordDlg()

## End(Not run)
```

PearsonTest *Pearson chi-square test for normality*

Description

Performs the Pearson chi-square test for the composite hypothesis of normality.

Usage

```
PearsonTest(x, n.classes = ceiling(2 * (n^(2/5))), adjust = TRUE)
```

Arguments

x	a numeric vector of data values. Missing values are allowed.
n.classes	The number of classes. The default is due to Moore (1986).
adjust	logical; if TRUE (default), the p-value is computed from a chi-square distribution with n.classes-3 degrees of freedom, otherwise from a chi-square distribution with n.classes-1 degrees of freedom.

Details

The Pearson test statistic is $P = \sum(C_i - E_i)^2/E_i$, where C_i is the number of counted and E_i is the number of expected observations (under the hypothesis) in class i. The classes are build is such a way that they are equiprobable under the hypothesis of normality. The p-value is computed from a chi-square distribution with n.classes-3 degrees of freedom if adjust is TRUE and from a chi-square distribution with n.classes-1 degrees of freedom otherwise. In both cases this is not (!) the correct p-value, lying somewhere between the two, see also Moore (1986).

Value

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

`statistic`	the value of the Pearson chi-square statistic.
`p.value`	the p-value for the test.
`method`	the character string "Pearson chi-square normality test".
`data.name`	a character string giving the name(s) of the data.
`n.classes`	the number of classes used for the test.
`df`	the degress of freedom of the chi-square distribution used to compute the p-value.

Note

The Pearson chi-square test is usually not recommended for testing the composite hypothesis of normality due to its inferior power properties compared to other tests. It is common practice to compute the p-value from the chi-square distribution with `n.classes` - 3 degrees of freedom, in order to adjust for the additional estimation of two parameters. (For the simple hypothesis of normality (mean and variance known) the test statistic is asymptotically chi-square distributed with `n.classes` - 1 degrees of freedom.) This is, however, not correct as long as the parameters are estimated by `mean(x)` and `var(x)` (or `sd(x)`), as it is usually done, see Moore (1986) for details. Since the true p-value is somewhere between the two, it is suggested to run `PearsonTest` twice, with `adjust = TRUE` (default) and with `adjust = FALSE`. It is also suggested to slightly change the default number of classes, in order to see the effect on the p-value. Eventually, it is suggested not to rely upon the result of the test.

The function call `PearsonTest(x)` essentially produces the same result as the S-PLUS function call `chisq.gof((x-mean(x))/sqrt(var(x)), n.param.est=2)`.

Author(s)

Juergen Gross <gross@statistik.uni-dortmund.de>

References

Moore, D.S., (1986) Tests of the chi-squared type. In: D'Agostino, R.B. and Stephens, M.A., eds.: *Goodness-of-Fit Techniques*. Marcel Dekker, New York.

Thode Jr., H.C., (2002) *Testing for Normality*. Marcel Dekker, New York. Sec. 5.2

See Also

`shapiro.test` for performing the Shapiro-Wilk test for normality. `AndersonDarlingTest`, `CramerVonMisesTest`, `LillieTest`, `ShapiroFranciaTest` for performing further tests for normality. `qqnorm` for producing a normal quantile-quantile plot.

Examples

```
PearsonTest(rnorm(100, mean = 5, sd = 3))
PearsonTest(runif(100, min = 2, max = 4))
```

PercTable *Percentage Table*

Description

Prints a 2-way contingency table along with percentages, marginal, and conditional distributions. All the frequencies are nested into one single table.

Usage

```
## Default S3 method:
PercTable(x, y = NULL, ...)

## S3 method for class 'table'
PercTable(tab, row.vars = NULL, col.vars = 2, digits = 3, big.mark = "", pfmt = FALSE,
          freq = TRUE, rfrq = "100", expected = FALSE, residuals = FALSE,
          stdres = FALSE, margins = NULL, ...)

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

Arguments

x, y	objects which can be interpreted as factors (including character strings). x and y will be tabulated via table(x, y). If x is a matrix, it will be coerced to a table via as.table(x).
tab	a r x c-contingency table
row.vars	a vector of row variables (see Details).
col.vars	a vector of column variables (see Details).
digits	an integer defining with how many digits the percentages will be printed.
big.mark	character. If not empty used as mark between every 3 decimals before the decimal point.
pfmt	logical. If set to TRUE the relative frequencies' format will be xx.xxx %, with digits respected.
freq	boolean. Should absolute frequencies be included? Defaults to TRUE.
rfrq	a string with 3 characters, each of them being 1 or 0. The first position means total percentages, the second means row percentages and the third column percentages. "011" produces a table output with row and column percentages.
expected	the expected counts under the null hypothesis.
residuals	the Pearson residuals, (observed - expected) / sqrt(expected).
stdres	standardized residuals, (observed - expected) / sqrt(V), where V is the residual cell variance (for the case where x is a matrix, n * p * (1 - p) otherwise).
margins	a vector, consisting out of 1 and/or 2. Defines the margin sums to be included. 1 stands for row margins, 2 for column margins, c(1,2) for both. Default is NULL (none).

formula	a formula of the form lhs ~ rhs where lhs will be tabled versus rhs (table(lhs, rhs)).
data	an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
...	the dots are passed from PercTable.default() to the PercTable.table().

Details

The absolute and relative frequencies are nested into one flat table by means of ftable. By means of row.vars, resp. col.vars, the structure of the table can be defined. row.vars can either be the names of the dimensions (included percentages are named "idx") or numbers (1:3, where 1 is the first dimension of the table, 2 the second and 3 the percentages).

Use Sort() if you want to have your table sorted by rows.

Value

Returns an object of class "ftable".

Author(s)

Andri Signorell <andri@signorell.net>

References

Agresti, Alan (2007) *Introduction to categorical data analysis*. NY: John Wiley and Sons, Section 2.4.5

See Also

Freq, table, ftable, prop.table, addmargins
There are similar functions in package **sfsmisc** printTable2 and package **vcd** table2d_summary, both lacking some of the flexibility we needed here.

Examples

```
data(d.pizza)
tab <- table(d.pizza$driver, d.pizza$area)

PercTable(tab=tab, col.vars=2)

PercTable(tab=tab, col.vars=2, margins=c(1,2))
PercTable(tab=tab, col.vars=2, margins=2)
PercTable(tab=tab, col.vars=2, margins=1)
PercTable(tab=tab, col.vars=2, margins=NULL)

PercTable(tab=tab, col.vars=2, rfrq="000")
```

```
# just the percentages without absolute values
PercTable(tab=tab, col.vars=2, rfrq="110", freq=FALSE)

# just the row percentages in percent format (pfmt = TRUE)
PercTable(tab, freq= FALSE, rfrq="010", pfmt=TRUE, digits=1)

# just the expected frequencies and the standard residuals
PercTable(tab=tab, rfrq="000", expected = TRUE, stdres = TRUE)


# rearrange output such that freq are inserted as columns instead of rows
PercTable(tab=tab, col.vars=c(3,2), rfrq="111")

# putting the cities in rows
PercTable(tab=tab, col.vars=c(3,1), rfrq="100", margins=c(1,2))

# formula interface with subset
PercTable(driver ~ area, data=d.pizza, subset=wine_delivered==0)

# sort the table by rows, order first column (Zurich), then third, then row.names (0)
PercTable(tab=Sort(tab, ord=c(1,3,0)))

# the vector interface
PercTable(x=d.pizza$driver, y=d.pizza$area)
PercTable(x=d.pizza$driver, y=d.pizza$area, margins=c(1,2), rfrq="000", useNA="ifany")

# one dimensional x falls back to the function Freq()
PercTable(x=d.pizza$driver)
```

`Permn` *Determine All Possible Permutations of a Set*

Description

Return the set of permutations for a given set of values. The values can be numeric values, characters or factors.

Usage

```
Permn(x)
```

Arguments

`x`	a vector of numeric values or characters. Characters need not be unique.

Details

This is an implementation of the Steinhaus-Johnson-Trotter permutation algorithm.

Value

a data.frame with all possible permutations of the values in x.

Author(s)

David Kahle <david.kahle@gmail.com>

References

Steinhaus, H. (1964): One hundred problems in elementary mathematics, *New York: Basic Books*, pp. 49-50

Johnson, Selmer M. (1963): Generation of permutations by adjacent transposition, *Mathematics of Computation* 17

Trotter, H. F. (August 1962): Algorithm 115: Perm, *Communications of the ACM* 5 (8)

Sedgewick, R. (1977): Permutation generation methods, *ACM Comput. Surv.* 9 (2)

See Also

combn, choose, factorial, GetAllSubsets

Examples

```
Permn(letters[2:5])
Permn(2:5)

Permn(c("a","b","c","a"))
```

PlotACF *Combined Plot of a Time Series and it's ACF and PACF*

Description

Combined plot of a time Series and it's autocorrelation and partial autocorrelation

Usage

```
PlotACF(series, lag.max = 10 * log10(length(series)), ...)
PlotGACF(series, lag.max = 10 * log10(length(series)), type = "cor", ylab = NULL, ...)
```

Arguments

series	univariate time series.
lag.max	integer. Defines the number of lags to be displayed. The default is 10 * log10(length(series)).
type	character string giving the type of acf to be computed. Allowed values are "cor" (the default), "cov" or "part" for autocorrelation, covariance or partial correlation.
ylab	a title for the y axis: see title.
...	the dots are passed to the plot command.

Details

PlotACF plots a combination of the time series and its autocorrelation and partial autocorrelation. PlotGACF is used as subfunction to produce the acf- and pacf-plots.

Author(s)

Markus Huerzeler (ETH Zurich), some minor modifications Andri Signorell <andri@signorell.net>

See Also

ts

Examples

```
PlotACF(AirPassengers)
```

PlotArea *Create an Area Plot*

Description

Produce a stacked area plot, or add polygons to an existing plot.

Usage

```
## Default S3 method:
PlotArea(x, y = NULL, prop = FALSE, add = FALSE, xlab = NULL,
         ylab = NULL, col = NULL, frame.plot = FALSE, ...)
## S3 method for class 'formula'
PlotArea(formula, data, subset, na.action = NULL, ...)
```

Arguments

x	numeric vector of x values, or if y=NULL a numeric vector of y values. Can also be a 1-dimensional table (x values in names, y values in array), matrix or 2-dimensional table (x values in row names and y values in columns), a data frame (x values in first column and y values in subsequent columns), or a time-series object of class ts/mts.
y	numeric vector of y values, or a matrix containing y values in columns.
prop	whether data should be plotted as proportions, so stacked areas equal 1.
add	whether polygons should be added to an existing plot.
xlab	label for x axis.
ylab	label for y axis.
col	fill color of polygon(s). The default is a vector of gray colors.
frame.plot	a logical indicating whether a box should be drawn around the plot.
formula	a formula, such as y ~ x or cbind(y1, y2) ~ x, specifying x and y values. A dot on the left-hand side, formula = . ~ x, means all variables except the one specified on the right-hand side.
data	a data frame (or list) from which the variables in formula should be taken.

subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NA values. The default is to ignore missing values in the given variables.
...	further arguments are passed to matplot and polygon.

Value

Matrix of cumulative sums that was used for plotting.

Author(s)

Arni Magnusson <arnima@hafro.is>

References

http://r.789695.n4.nabble.com/areaplot-td2255121.html

See Also

barplot, polygon

Examples

```
data(d.pizza)

# PlotArea with stapled areas
tab <- table( d.pizza$date, d.pizza$driver )
PlotArea(x=as.Date(rownames(tab)), y=tab, xaxt="n", xlab="Date", ylab="Pizzas delivered" )

# add x-axis and some text labels
xrng <- pretty(range(as.Date(rownames(tab))))
axis(side=1, at=xrng, labels=xrng)
text( x=min(d.pizza$date + .5, na.rm=TRUE), y=cumsum(tab[2,])-2.5, label=levels(d.pizza$driver),
  adj=c(0,0.5), col=TextContrastColor(gray.colors(7)))


# formula
PlotArea(Armed.Forces~Year, data=longley)
PlotArea(cbind(Armed.Forces,Unemployed)~Year, data=longley)

# add=TRUE
plot(1940:1970, 500*runif(31), ylim=c(0,500))
PlotArea(Armed.Forces~Year, data=longley, add=TRUE)

# matrix
PlotArea(WorldPhones)
PlotArea(WorldPhones, prop=TRUE, col=rainbow(10))

# table
PlotArea(table(d.pizza$weekday))
PlotArea(table(d.pizza$weekday, d.pizza$driver))

# ts/mts
PlotArea(austres)
PlotArea(Seatbelts[,c("drivers","front","rear")],
```

```
        ylab="Killed or seriously injured")
abline(v=1983+1/12, lty=3)
```

PlotBag *PlotBag, a bivariate boxplot*

Description

compute.PlotBag() computes an object describing a PlotBag of a bivariate data set. plot.PlotBag() plots a bagplot object. PlotBag() computes and plots a bagplot.

Usage

```
PlotBag(x, y, factor = 3, na.rm = FALSE, approx.limit = 300,
        show.outlier = TRUE, show.whiskers = TRUE,
        show.looppoints = TRUE, show.bagpoints = TRUE,
        show.loophull = TRUE, show.baghull = TRUE,
        create.plot = TRUE, add = FALSE, pch = 16, cex = 0.4,
        dkmethod = 2, precision = 1, verbose = FALSE,
        debug.plots = "no",   col.loophull = "#aaccff",
        col.looppoints = "#3355ff", col.baghull = "#7799ff",
        col.bagpoints = "#000088", transparency = FALSE, ...
)
PlotBagPairs(dm, trim = 0.0, main, numeric.only = TRUE,
        factor = 3, approx.limit = 300, pch = 16,
        cex = 0.8, precision = 1, col.loophull = "#aaccff",
        col.looppoints = "#3355ff", col.baghull = "#7799ff",
        col.bagpoints = "#000088", ...)

compute.bagplot(x, y, factor = 3, na.rm = FALSE, approx.limit = 300,
        dkmethod = 2, precision = 1, verbose = FALSE, debug.plots = "no" )

## S3 method for class 'bagplot'
plot(x, show.outlier = TRUE, show.whiskers = TRUE,
         show.looppoints = TRUE, show.bagpoints = TRUE,
         show.loophull = TRUE, show.baghull = TRUE, add = FALSE,
         pch = 16, cex = .4, verbose = FALSE, col.loophull = "#aaccff",
         col.looppoints = "#3355ff", col.baghull = "#7799ff",
         col.bagpoints = "#000088", transparency = FALSE,...)
```

Arguments

x	x values of a data set; in PlotBag: an object of class PlotBag computed by compute.PlotBag
y	y values of the data set
factor	factor defining the loop
na.rm	if TRUE 'NA' values are removed otherwise exchanged by median

approx.limit	if the number of data points exceeds approx.limit a sample is used to compute some of the quantities; default: 300
show.outlier	if TRUE outlier are shown
show.whiskers	if TRUE whiskers are shown
show.looppoints	if TRUE loop points are plottet
show.bagpoints	if TRUE bag points are plottet
show.loophull	if TRUE the loop is plotted
show.baghull	if TRUE the bag is plotted
create.plot	if FALSE no plot is created
add	if TRUE the bagplot is added to an existing plot
pch	sets the plotting character
cex	sets characters size
dkmethod	1 or 2, there are two method of approximating the bag, method 1 is very rough (only based on observations
precision	precision of approximation, default: 1
verbose	automatic commenting of calculations
debug.plots	if TRUE additional plots describing intermediate results are constructed
col.loophull	color of loop hull
col.looppoints	color of the points of the loop
col.baghull	color of bag hull
col.bagpoints	color of the points of the bag
transparency	see section details
dm	x
trim	x
main	x
numeric.only	x
...	additional graphical parameters

Details

A bagplot is a bivariate generalization of the well known boxplot. It has been proposed by Rousseeuw, Ruts, and Tukey. In the bivariate case the box of the boxplot changes to a convex polygon, the bag of bagplot. In the bag are 50 percent of all points. The fence separates points within the fence from points outside. It is computed by increasing the the bag. The loop is defined as the convex hull containing all points inside the fence. If all points are on a straight line you get a classical boxplot. PlotBag() plots bagplots that are very similar to the one described in Rousseeuw et al. Remarks: The two dimensional median is approximated. For large data sets the error will be very small. On the other hand it is not very wise to make a (graphical) summary of e.g. 10 bivariate data points.

In case you want to plot multiple (overlapping) bagplots, you may want plots that are semi-transparent. For this you can use the transparency flag. If transparency==TRUE the alpha layer is set to '99' (hex). This causes the bagplots to appear semi-transparent, but ONLY if the output device is PDF and opened using: pdf(file="filename.pdf", version="1.4"). For this reason, the default is transparency==FALSE. This feature as well as the arguments to specify different colors has been proposed by Wouter Meuleman.

Value

compute.bagplot returns an object of class bagplot that could be plotted by plot.bagplot(). An object of the bagplot class is a list with the following elements: center is a two dimensional vector with the coordinates of the center. hull.center is a two column matrix, the rows are the coordinates of the corners of the center region. hull.bag and hull.loop contain the coordinates of the hull of the bag and the hull of the loop. pxy.bag shows you the coordinates of the points of the bag. pxy.outer is the two column matrix of the points that are within the fence. pxy.outlier represent the outliers. The vector hdepths shows the depths of data points. is.one.dim is TRUE if the data set is (nearly) one dimensional. The dimensionality is decided by analysing the result of prcomp which is stored in the element prdata. xy shows you the data that are used for the bagplot. In the case of very large data sets subsets of the data are used for constructing the bagplot. A data set is very large if there are more data points than approx.limit. xydata are the input data structured in a two column matrix.

Note

Version of bagplot: 10/2012

Author(s)

Peter Wolf

References

P. J. Rousseeuw, I. Ruts, J. W. Tukey (1999): The bagplot: a bivariate boxplot, *The American Statistician*, vol. 53, no. 4, 382–387

See Also

boxplot

Examples

```
# example: 100 random points and one outlier
dat <- cbind(rnorm(100) + 100, rnorm(100) + 300)
dat <- rbind(dat, c(105,295))
PlotBag(dat,factor=2.5,create.plot=TRUE,approx.limit=300,
   show.outlier=TRUE,show.looppoints=TRUE,
   show.bagpoints=TRUE,dkmethod=2,
   show.whiskers=TRUE,show.loophull=TRUE,
   show.baghull=TRUE,verbose=FALSE)
# example of Rousseeuw et al., see R-package rpart
cardata <- structure(as.integer( c(2560,2345,1845,2260,2440,
 2285, 2275, 2350, 2295, 1900, 2390, 2075, 2330, 3320, 2885,
 3310, 2695, 2170, 2710, 2775, 2840, 2485, 2670, 2640, 2655,
 3065, 2750, 2920, 2780, 2745, 3110, 2920, 2645, 2575, 2935,
 2920, 2985, 3265, 2880, 2975, 3450, 3145, 3190, 3610, 2885,
 3480, 3200, 2765, 3220, 3480, 3325, 3855, 3850, 3195, 3735,
 3665, 3735, 3415, 3185, 3690, 97, 114, 81, 91, 113, 97, 97,
 98, 109, 73, 97, 89, 109, 305, 153, 302, 133, 97, 125, 146,
 107, 109, 121, 151, 133, 181, 141, 132, 133, 122, 181, 146,
 151, 116, 135, 122, 141, 163, 151, 153, 202, 180, 182, 232,
 143, 180, 180, 151, 189, 180, 231, 305, 302, 151, 202, 182,
 181, 143, 146, 146)), .Dim = as.integer(c(60, 2)),
 .Dimnames = list(NULL, c("Weight", "Disp.")))
```

```
PlotBag(cardata,factor=3,show.baghull=TRUE,
  show.loophull=TRUE,precision=1, dkmethod=2)
title("car data Chambers/Hastie 1992")
# points of y=x*x
PlotBag(x=1:30,y=(1:30)^2,verbose=FALSE,dkmethod=2)
# one dimensional subspace
PlotBag(x=1:100,y=1:100)
```

PlotBubble *Draw a Bubble Plot*

Description

Draw a bubble plot, defined by a pair of coordinates x, y to place the bubbles, an area definition configuring the dimension and a color vector setting the color of the bubbles. The legitimation to define a new function instead of just using `plot(symbols(...))` is the automated calculation of the axis limits, ensuring all elements will be fully visible.

Usage

```
PlotBubble(x, y, area, col, border = NA, na.rm = FALSE, inches = FALSE, ...)
```

Arguments

x, y	the x and y co-ordinates for the centres of the bubbles. They can be specified in any way which is accepted by xy.coords.
area	a vector giving the area of the bubbles.
col	colors for the bubbles, passed to symbol.
border	the border color fot the bubbles. Set NA if there should be no border at all. This is the default.
na.rm	logical, should NAs be omitted? Defaults to FALSE.
inches	TRUE, FALSE or a positive number. See 'Details'.
...	the dots are passed to the plot function.

Details

Argument inches controls the sizes of the symbols. If TRUE (the default), the symbols are scaled so that the largest dimension of any symbol is one inch. If a positive number is given the symbols are scaled to make largest dimension this size in inches (so TRUE and 1 are equivalent). If inches is FALSE, the units are taken to be those of the appropriate axes. This behaviour is the same as in symbols.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

symbols, sunflowerplot

Examples

```
PlotBubble( x=d.pizza$delivery_min, y=d.pizza$temperature, area=d.pizza$price,
  xlab="delivery time", ylab="temperature",
  col=SetAlpha(as.numeric(d.pizza$area)+2, .5), border="darkgrey",
  na.rm=TRUE, main="Price-Bubbles", panel.first=grid())


PlotBubble(d.world$x, d.world$y, area=d.world$pop/90, col=SetAlpha("steelblue",0.4),
  border="darkblue", xlab="", ylab="", panel.first=grid(),
  main="World population")
text(d.world$x, d.world$y, labels=d.world$country, cex=0.7, adj=0.5)
```

PlotCandlestick *Plot Candlestick Chart*

Description

Plot a candlestick chart. This is used primarily to describe price movements of a security, derivative, or currency over time. Candlestick charts are a visual aid for decision making in stock, foreign exchange, commodity, and option trading.

Usage

```
PlotCandlestick(x, y, xlim = NULL, ylim = NULL,
                col = c("springgreen4","firebrick"),
                border = NA, args.grid = NULL, ...)
```

Arguments

x	a numeric vector for the x-values. Usually a date.
y	the y-values in a matrix (or a data.frame that can be coerced to a matrix) with 4 columns, whereas the first column contains the open price, the second the high, the third the lowest and the 4th the close price of daily stock prices.
xlim	the x limits (x1, x2) of the plot. The default value, NULL, indicates that the range of the finite values to be plotted should be used.
ylim	the y limits of the plot.
col	color for the body. To better highlight price movements, modern candlestick charts often replace the black or white of the candlestick body with colors such as red for a lower closing and blue or green for a higher closing.
border	the border color of the rectangles. Default is NA, meaning no border will be plotted.
args.grid	the arguments of a potential grid. Default is NULL, which will have a grid plotted. If arguments are provided, they have to be organized as list with the names of the arguments. (For example: ..., args.grid = list(col="red"))
...	the dots are passed to plot() command

Details

Candlesticks are usually composed of the body (black or white), and an upper and a lower shadow (wick): the area between the open and the close is called the real body, price excursions above and below the real body are called shadows. The wick illustrates the highest and lowest traded prices of a security during the time interval represented. The body illustrates the opening and closing trades. If the security closed higher than it opened, the body is white or unfilled, with the opening price at the bottom of the body and the closing price at the top. If the security closed lower than it opened, the body is black, with the opening price at the top and the closing price at the bottom. A candlestick need not have either a body or a wick.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

PlotBubble, stars

Examples

```
nov <- rbind(
 "2013-05-28"= c(70.99,71.82,70.49,71.49),
 "2013-05-29"= c(71.13,71.90,70.81,71.57),
 "2013-05-30"= c(71.25,71.53,70.90,71.01),
 "2013-05-31"= c(70.86,70.92,70.30,70.30),
 "2013-06-03"= c(70.56,70.89,70.05,70.74),
 "2013-06-04"= c(70.37,71.11,69.67,69.90),
 "2013-06-05"= c(69.76,69.76,68.92,68.99),
 "2013-06-06"= c(69.13,70.02,68.56,70.02),
 "2013-06-07"= c(70.45,70.52,69.51,70.20),
 "2013-06-10"= c(70.53,70.75,70.05,70.20),
 "2013-06-11"= c(69.36,69.66,69.01,69.17),
 "2013-06-12"= c(69.65,70.03,68.85,69.21),
 "2013-06-13"= c(69.21,70.18,69.13,70.10),
 "2013-06-14"= c(70.17,70.48,69.30,69.58),
 "2013-06-17"= c(70.14,70.96,69.98,70.44),
 "2013-06-18"= c(70.55,71.97,70.55,71.49),
 "2013-06-19"= c(71.33,72.00,70.89,70.97),
 "2013-06-20"= c(70.04,70.06,68.40,68.55),
 "2013-06-21"= c(69.15,69.27,67.68,68.21)
)
colnames(nov) <- c("open","high","low","close")

PlotCandlestick(x=as.Date(rownames(nov)), y=nov, border=NA, las=1, ylab="")
```

PlotCirc *Plot Circular Plot*

Description

This visualising scheme represents the unidirectional relationship between the rows and the columns of a contingency table.

Usage

```
PlotCirc(tab, acol = rainbow(sum(dim(tab))), aborder = "darkgrey",
         rcol = SetAlpha(acol[1:nrow(tab)], 0.5), rborder = "darkgrey",
         gap = 5, main = "", labels = NULL, cex.lab = 1.0)
```

Arguments

tab	a table to be visualised.
acol	the colors for the peripheral annuli.
aborder	the border colors for the peripheral annuli.
rcol	the colors for the ribbons.
rborder	the border colors for the ribbons.
gap	the gap between the entities in degrees.
main	the main title, defaults to "".
labels	the labels. Defaults to the column names and rownames of the table.
cex.lab	the character extension for the labels.

Details

The visual scheme of representing relationships can be applied to a table, given the observation that a table cell is a relationship (with a value) between a row and column. By representing the row and columns as segments along the circle, the information in the corresponding cell can be encoded as a link between the segments. In general, the cell represents a unidirectional relationship (e.g. row->column) - in this relationship the role of the segments is not interchangeable (e.g. (row,col) and (col,row) are different cells). To identify the role of the segment, as a row or column, the ribbon is made to terminate at the row segment but slightly away from the column segment. In this way, for a given ribbon, it is easy to identify which segment is the row and which is the column.

Author(s)

Andri Signorell <andri@signorell.net>

References

The idea is taken from: http://circos.ca/presentations/articles/vis_tables1/

See Also

PlotPolar

Examples

```
tab <- matrix(c(2,5,8,3,10,12,5,7,15), nrow=3, byrow=FALSE)
dimnames(tab) <- list(c("A","B","C"), c("D","E","F"))
tab


PlotCirc( tab,
  acol = c("dodgerblue","seagreen2","limegreen","olivedrab2","goldenrod2","tomato2"),
  rcol = SetAlpha(c("red","orange","olivedrab1"), 0.5)
)
```

```
tab <- table(d.pizza$weekday, d.pizza$operator)
par(mfrow=c(1,2))
PlotCirc(tab, main="weekday ~ operator")
PlotCirc(t(tab), main="operator ~ weekday")
```

PlotCorr *Plot a Correlation Matrix*

Description

This function produces a graphical display of a correlation matrix. The cells of the matrix can be shaded or colored to show the correlation value.

Usage

```
PlotCorr(x, cols = colorRampPalette(c("red", "white", "blue"), space = "rgb")(20),
         breaks = seq(-1, 1, length = length(cols) + 1),
         border = NA, lwd = 1,
         args.colorlegend = NULL, xaxt = par("xaxt"), yaxt = par("yaxt"),
         cex.axis = 0.8, las = 2, mar = c(3, 8, 8, 8), ...)
```

Arguments

x	x is a correlation matrix to be visualized.
cols	the colors for shading the matrix
breaks	a set of breakpoints for the colours: must give one more breakpoint than colour. These are passed to `image()` function. If breaks is specified then the algorithm used follows `cut`, so intervals are closed on the right and open on the left except for the lowest interval.
border	color for borders. The default is `NA`, which means borders are omitted.
lwd	line width for borders. Default is 1.
args.colorlegend	list of arguments for the `ColorLegend`. Use NA if no color legend should be painted.
xaxt	parameter to define, whether to draw an x-axis, defaults to `"n"`.
yaxt	parameter to define, whether to draw an y-axis, defaults to `"n"`.
cex.axis	character extension for the axis labels.
las	the style of axis labels.
mar	sets the margins, defaults to mar = c(3, 8, 8, 8) as we need a bit more room on the right.
...	the dots are passed to the function `image`, which produces the plot.

Value

no values returned.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

image, ColorLegend, corrgram()

Examples

```
m <- cor(d.pizza[,WhichNumerics(d.pizza)], use="pairwise.complete.obs")

PlotCorr(m, cols=colorRampPalette(c("red", "black", "green"), space = "rgb")(20))
PlotCorr(m, cols=colorRampPalette(c("red", "black", "green"), space = "rgb")(20),
         args.colorlegend=NA)

m <- PairApply(d.diamonds[,WhichFactors(d.diamonds)], CramerV)
PlotCorr(m, cols = colorRampPalette(c("white", "steelblue"), space = "rgb")(20),
         breaks=seq(0, 1, length=21), border="black",
       args.colorlegend = list(labels=sprintf("%.1f", seq(1, 0, length = 11)), frame=TRUE)
)
title(main="Cramer's V", line=2)
text(x=rep(1:ncol(m),ncol(m)), y=rep(1:ncol(m),each=ncol(m)),
     label=sprintf("%0.2f", m[,ncol(m):1]), cex=0.8, xpd=TRUE)

# Spearman correlation on ordinal factors
csp <- cor(data.frame(lapply(d.diamonds[,c("carat", "clarity", "cut", "polish",
                      "symmetry", "price")], as.numeric)), method="spearman")
PlotCorr(csp)

# some more colors
PlotCorr(cor(mtcars), col=PalDescTools("RedWhiteBlue1", 100), border="grey",
         args.colorlegend=list(labels=FormatFix(seq(1,-1,-.25), 2), frame="grey"))
```

PlotDesc *Display descriptive plots*

Description

Specific descriptive plots depending on the class of x.

Usage

```
PlotDesc(x, ..., wrd = NULL)

## Default S3 method:
PlotDesc(x, ...)

## S3 method for class 'integer'
PlotDesc(x, main = deparse(substitute(x)),
         ord = c("val_asc","val_desc","frq_asc","frq_desc"),
         maxrows = 10, ... , wrd = NULL)
## S3 method for class 'numeric'
```

```
PlotDesc(x, main = deparse(substitute(x)), ...,
         wrd = NULL)
## S3 method for class 'factor'
PlotDesc(x, main = deparse(substitute(x)),
         ord = c("desc", "level", "name", "asc", "none"),
         maxrows = 10, lablen = 25, type = c("bar","dot"),
         col = hblue, ..., wrd = NULL)
## S3 method for class 'table'
PlotDesc(x, col0 = hred, col1=hblue,
         horiz = TRUE,  main="", ..., wrd = NULL)
## S3 method for class 'ordered'
PlotDesc(x, ..., wrd = NULL)
## S3 method for class 'data.frame'
PlotDesc(x, ..., wrd = NULL)
## S3 method for class 'logical'
PlotDesc(x, xlab = "", col0 = hblue, col1 = hred, ...,
         wrd = NULL)
## S3 method for class 'Date'
PlotDesc(x, breaks = "month", ..., wrd = NULL)

PlotDescNumFact(formula, data, main=deparse(formula), notch=FALSE,
                add_ni = TRUE, ..., wrd = NULL)
PlotDescNumNum(form1, form2, data, ..., wrd = NULL)
```

Arguments

x	the vector to be plotted.
main	the main title of the plot.
ord	the row order of a frequency table to be chosen. Is used for factors and integers.
maxrows	the maximum number of rows to be displayed for a factor.
lablen	the maximum numbe of characters for a factor level to be displayed, before the level is truncated and ... are added.
type	the type of plot to be used for describing factors. Can be "bar" for a horizontal barchart or "dot" for a dotchart.
col0, col1	two colors to be chosen for doing mosaicplots.
col	color of the points used in a dotchart, typically for integers. Defaults to lightblue.
xlab	the xlab for the logical plot.
breaks	vector of limits to bin a date.
horiz	logical, indicating if the two mosaicplots should be arranged horizontally (default is TRUE).
formula	a formula, such as y ~ grp, where y is a numeric vector of data values to be split into groups according to the grouping variable grp (usually a factor).
data	a data.frame (or list) from which the variables in formula should be taken.
notch	if notch is TRUE, a notch is drawn in each side of the boxes. If the notches of two plots do not overlap this is 'strong evidence' that the two medians differ.
add_ni	logical. Indicates if the group length should be displayed in the boxplot.
form1, form2	the formula used for calculating the smoother.

wrd — the pointer to a word instance. Can be a new one, created by GetNewWrd() or an existing one, created by GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").

...

Details

See the detailed description for informations about specific plots.

Value

no value returned

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Desc

Examples

```
PlotDesc(x=na.omit(d.pizza$delivery_min))    # numeric
PlotDesc(x=na.omit(d.pizza$week))                          # integer
PlotDesc(x=na.omit(d.pizza$driver))                        # factor
PlotDesc(x=na.omit(d.pizza$quality))                       # ordered factor
PlotDesc(x=na.omit(d.pizza$wrongpizza))      # logical
PlotDesc(x=na.omit(d.pizza$date))            # Date

d.frm <- d.pizza[,c("price","operator")]
d.frm <- d.frm[complete.cases(d.frm),]
PlotDescNumFact(temperature ~ driver, data=d.pizza)       # numeric ~ factor

PlotDesc(table(d.pizza$driver, d.pizza$operator))         # factor ~ factor
```

PlotDotCI — *Plot a Dotchart with Confidence Intervals*

Description

Plot a dotchart with confidence intervals as segments. The reason for creating a new function for the job is the automated calculation of the axis limits, ensuring all error bars will be fully visible.

Usage

```
PlotDotCI(x, xlim = NULL,
          pch = 21, pch.cex = 1, pch.col = "black", pch.bg = "grey50",
          lcol = "grey40", lwd = 2,
          args.legend = NULL, code = 3,
          mar = c(7.1, 4.1, 4.1, 2.1), ...)
```

Arguments

x	matrix with 3 columns, the first is used as x-values, the second is the left bound and the 3rd the right bound of the segment.
xlim	the x limits of the plot.
pch	a vector of plotting characters or symbols.
pch.cex	magnification to be used for plotting characters relative to the current setting of cex.
pch.col	the colors for points. If using 21 etc. this is the margins color of the point character.
pch.bg	the colors for points. If using 21 etc. this is the fill color of the point character.
lcol	the colors for lines.
lwd	the widths for the segments.
args.legend	list of additional arguments to be passed to the legend function. Use args.legend = NA if no legend should be added.
code	integer value. Determines the kind of arrows to be drawn. code = 1 means that a lower arrowhead will be drawn, code = 2 will produce one on the right side and code = 3 on both sides. (See the function arrows).
mar	numeric vector with margins defined.
...	further arguments are passed to the function dotchart.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

PlotDotCIp

Examples

```
xci <- do.call(rbind, tapply( d.pizza$delivery_min, d.pizza$driver,
  MeanCI, conf.level=0.99, na.rm=TRUE))

PlotDotCI(xci, main="delivery_min ~ driver",
  args.legend=list(y=-1.5, legend=c("estimate", "99%-CI")))
```

PlotDotCIp *Plot a Dotchart with Binomial Confidence Intervals*

Description

Plot a dotchart with binomial confidence intervals ("wilson") as segments.

Usage

```
PlotDotCIp(x, n, xlim = c(0, 1),
           ord = c("rel", "abs", "names"), decreasing = FALSE, ...)
```

Arguments

x	number of successes, or a vector of length 2 giving the numbers of successes and failures, respectively.
n	number of trials; ignored if x has length 2.
xlim	the x limits of the plot.
ord	how should the result be ordered? Default is relative frequency "rel". The argument can be abbreviated.
decreasing	logical, sort order decreasing or ascending?
...	further arguments are passed to the function PlotDotCI().

Author(s)

Andri Signorell <andri@signorell.net>

See Also

PlotDotCI

Examples

```
tab <- table(d.pizza$driver, d.pizza$wine_delivered)

PlotDotCIp(x=tab[,2], n=apply(tab,1,sum), ord="abs", dec=TRUE)
```

PlotFaces *Chernoff Faces*

Description

Plot Chernoff faces. The rows of a data matrix represent cases and the columns the variables.

Usage

```
PlotFaces(xy, which.row, fill = FALSE, nrow, ncol,
          scale = TRUE, byrow = FALSE, main, labels)
```

Arguments

xy	xy data matrix, rows represent individuals and columns attributes.
which.row	defines a permutation of the rows of the input matrix.
fill	logic. If set to TRUE, only the first nc attributes of the faces are transformed, nc is the number of columns of x.
nrow	number of columns of faces on graphics device
ncol	number of rows of faces
scale	logic. If set to TRUE, attributes will be normalized.
byrow	if(byrow==TRUE), x will be transposed.
main	title.
labels	character strings to use as names for the faces.

Details

The features paramters of this implementation are:

- 1 height of face
- 2 width of face
- 3 shape of face
- 4 height of mouth
- 5 width of mouth
- 6 curve of smile
- 7 height of eyes
- 8 width of eyes
- 9 height of hair
- 10 width of hair
- 11 styling of hair
- 12 height of nose
- 13 width of nose
- 14 width of ears
- 15 height of ears

For details look at the literate program of `faces`

Value

a plot of faces is created on the graphics device, no numerical results

Note

version 12/2003

Author(s)

H. P. Wolf

References

Chernoff, H. (1973) The use of faces to represent statistiscal assoziation, *JASA*, 68, pp 361–368.

The smooth curves are computed by an algorithm found in:
Ralston, A. and Rabinowitz, P. (1985) *A first course in numerical analysis*, McGraw-Hill, pp 76ff.
http://www.wiwi.uni-bielefeld.de/~wolf/: S/R - functions : faces

Examples

```
PlotFaces(rbind(1:3,5:3,3:5,5:7))

data(longley)
PlotFaces(longley[1:9,])

set.seed(17)
PlotFaces(matrix(sample(1:1000,128,),16,8),main="random faces")
```

PlotFdist *Frequency Distribution Plot*

Description

This function is designed to give a univariate graphic representation of a numeric vector's frequency distribution. It combines a histogram, a density curve, a boxplot and a plot of the empirical cumulative distribution function (ecdf) in one single plot, resulting in a dense and informative picture of the facts. Still the function remains flexible as all possible arguments can be passed to the single components (hist, boxplot etc.) by list (see examples).

Usage

```
PlotFdist(x, main = deparse(substitute(x)), xlab = "", xlim = NULL,
        do.hist = !(all(IsWhole(x, na.rm = TRUE)) & length(unique(na.omit(x))) < 13),
          args.hist = NULL, args.rug = NA, args.dens = NULL,
          args.boxplot = NULL, args.ecdf = NULL, heights = NULL,
          pdist = c(0,0), na.rm = FALSE)
```

Arguments

x	the numerical variable, whose distribution is to be plotted.
main	main title of the plot.
xlab	label of the x-axis, defaults to "". (The name of the variable is typically placed in the main title and would be redundant.)
xlim	range of the x-axis, defaults to a pretty `range(x, na.rm = TRUE)`.
do.hist	defines, whether a histogram or a plot with `type = "h"` should be used. Default is `TRUE` (meaning a histogram will be plotted), unless x is an integer with less than 13 unique values!
args.hist	list of additional arguments to be passed to the histogram `hist()`, ignored if `do.hist = FALSE`. The defaults chosen when setting `args.hist = NULL` are more or less the same as in `hist`.
args.rug	list of additional arguments to be passed to the function `rug()`. Use `args.rug = NA` if no rug should be added. This is the default. Use `args.rug = NULL` to add rug with reasonable default values.
args.dens	list of additional arguments to be passed to `density()`. Use `args.dens = NA` if no density curve should be drawn. The defaults are taken from `density`.
args.boxplot	list of additional arguments to be passed to the boxplot `boxplot()`. The defaults are pretty much the same as in `boxplot`.
args.ecdf	list of additional arguments to be passed to `ecdf()`. Use `args.ecdf = NA` if no empirical cumulation function should be included in the plot. The defaults are taken from `plot.ecdf`.
heights	heights of the plotparts, defaults to `c(2,0.5,1.4)` for the histogram, the boxplot and the empirical cumulative distribution function, resp. to `c(2,1.5)` for a histogram and a boxplot only.
pdist	distances of the plotparts, defaults to `c(0, 0)`, say there will be no distance between the histogram, the boxplot and the ecdf-plot. This can be changed for instance in case that the xaxis is to be added to the histogram.

na.rm	logical, should NAs be omitted? Histogram and boxplot could do without this option, but the density-function refuses to plot with missings. Defaults to FALSE.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

hist, boxplot, ecdf, density, rug, layout

Examples

```
# create a new window and do the plot
PlotFdist(x=d.pizza$delivery_min, na.rm=TRUE)

# define additional arguments for hist and dens
PlotFdist(d.pizza$delivery_min, args.hist=list(breaks=50),
  args.dens=list(col="olivedrab4"), na.rm=TRUE )

# do a "h"-plot instead of a histogram for integers
PlotFdist(d.pizza$weekday, na.rm=TRUE)

# special arguments for ecdf
PlotFdist(x=faithful$eruptions, args.ecdf=list(verticals=FALSE,
  do.points=TRUE, cex=1.2, pch=16, lwd=1), args.rug=NULL)

# no density curve, no ecdf but add rug instead, make boxplot a bit higher
PlotFdist(x=d.pizza$delivery_min, na.rm=TRUE, args.dens=NA, args.ecdf=NA,
  args.hist=list(xaxt="s"),  # display x-axis on the histogram
  args.rug=TRUE, heights=c(3, 2.5), pdist=2.5, main="Delivery time")

# alpha channel on rug is cool, but take its time for being drawn...
PlotFdist(x=d.pizza$temperature, args.rug=list(col=SetAlpha("black", 0.1)), na.rm=TRUE)
```

PlotHorizBar *Plot Horizontal Bars*

Description

A more flexible implementation of plotting horizontal bars with definable start and end points.

Usage

```
PlotHorizBar(from, to, grp = 1, col = "lightgrey", border = "black", ...)
```

Arguments

from	a numeric vector specifying the start of the bars.
to	a numeric vector specifying the end of the bars.
grp	a grouping factor, determining on which line the bar will be printed. The groups start with y=1 and grow.

col	a vector with the colors of the bars. Default is "lightgrey". This will be recyled if necessary.
border	the border color of the bars. Default ist "black". Set this to NA if no border is to be printed.
...	the dots are passed to plot.new.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

barplot

Examples

```
PlotHorizBar( 1:5, 3:8)
```

PlotMarDens *Scatterplot with Marginal Densities*

Description

Draw a scatter plot with marginal densities on the x- and y-axis. Groups can be defined by grp.

Usage

```
PlotMarDens(x, y, grp = 1, xlim = NULL, ylim = NULL,
            col = rainbow(nlevels(factor(grp))),
            mardens = c("all","x","y"), pch = 1, pch.cex = 1,
            main = "", na.rm = FALSE, args.legend = NULL,
            args.dens = NULL, ...)
```

Arguments

x	numeric vector of x values.
y	numeric vector of y values (of same length as x).
grp	grouping variable(s), typically factor(s), all of the same length as x.
xlim	the x limits of the plot.
ylim	the y limits of the plot.
col	the colors for lines and points. Uses rainbow() colors by default.
mardens	which marginal densities to plot. Can be set to either just x or y, or both ("all", latter being the default).
pch	a vector of plotting characters or symbols.
pch.cex	magnification to be used for plotting characters relative to the current setting of cex.
main	a main title for the plot, see also title.

na.rm	logical, should NAs be omitted? Defaults to FALSE.
args.legend	list of additional arguments for the legend. args.legend set to NA prevents a legend from being drawn.
args.dens	list of additional arguments to be passed to density. Use args.dens = NA if no density curve should be drawn. The defaults are taken from density.
...	further arguments are passed to the function plot().

Author(s)

Andri Signorell <andri@signorell.net>

See Also

plot, points, density, layout

Examples

```
# best seen with: x11(7.5, 4.7)

# just one variable with marginal densities
PlotMarDens( y=d.pizza$temperature, x=d.pizza$delivery_min, grp=1
             , xlab="delivery_min", ylab="temperature", col=SetAlpha("brown", 0.4)
             , pch=15, lwd=3
             , panel.first= grid(), args.legend=NA
             , main="Temp ~ delivery"
)

# use a group variable
PlotMarDens( y=d.pizza$temperature, x=d.pizza$delivery_min, grp=d.pizza$area
  , xlab="delivery_min", ylab="temperature", col=c("brown","orange","lightsteelblue")
  , panel.first=list( grid() )
  , main = "temperature ~ delivery_min | area"
)
```

PlotMatrix *Scatterplot Matrix*

Description

Plots a scatterplot matrix, for which the variables shown horizontally do not necessarily coincide with those shown vertically. If desired, the matrix is divided into several blocks such that it fills more than 1 plot page.

Usage

```
PlotMatrix(x, y = NULL, data = NULL, panel = l.panel,
  nrows = 0, ncols = nrows, save = TRUE,
  robrange. = FALSE, range. = NULL, pch = NULL, col = 1,
  reference = 0, ltyref = 3, log = "", xaxs = "r", yaxs = "r",
  xaxmar = NULL, yaxmar = NULL, vnames = NULL,
  main = "", cex.points = NA, cex.lab = 0.7, cex.text = 1.3, cex.title = 1,
  bty = "o", oma = NULL, ...)
```

Arguments

x	data for columns (x axis), or formula defining column variables. If it is a formula containing a left hand side, the left side variables will be used last.
y	data or formula for rows (y axis). Defaults to x
data	data.frame containing the variables in case x or y is a formula
panel	a function that generates the marks of the individual panels, see Details. Defaults essentially to points or text depending on the argument pch
nrows	number of rows of panels on a page
ncols	number of columns of panels on a page
save	if y is not provided and save==TRUE, the first row and the last column are suppressed.
robrange.	if TRUE, robust plot ranges will be used
range.	plot ranges, given as a matrix with 2 rows (min, max) and colnames identifying the variables.
pch	plotting character. A vector of integers, characters or strings can also be given for the default panel function
col	color(s) to be used for plotting the observations
reference	coordinates for reference lines to be shown in the panels. A named vector can be used to define a value for each or any variable.
ltyref	line type for reference lines
log	specifies logarithmic scale of axes. "x" asks for log scale on horizontal axis, "y", on vertical axis, "xy", on both axes.
xaxs, yaxs	styles for x and y axis, see par
xaxmar, yaxmar	in which margin should the x- [y-] axis be labelled?
vnames	labels for the variables
main	main title for the plot (to be repeated on each plot page)
cex.points	character expansion for showing the observations
cex.lab, cex.text	character expansion for variable labels in the margin and in the "diagonal", respectively, relative to cex
cex.title	character expansion for the main title
bty	box type for each panel, see par
oma	width of outer margins, ee par
...	further arguments passed to the panel function

Details

If x or y is a data.frame, it is converted to a numerical matrix.

The panel function can be user written. It needs $>= 6$ arguments, which are given:

- the values of the horizontal variable,
- the values of the vertical variable,
- the index of the variable shown horizontally, among the y variables,
- the index of the variable shown vertically, among the x variables,

- argument pch, and
- argument col

Since large scatterplot matrices lead to tiny panels, PlotMatrix splits the matrix into blocks of at most nrows rows and ncols columns. If these numbers are missing, they default to nrows=5 and ncols=6 for landscape pages, and to nrows=8 and ncols=5 for portrait pages.

Value

none

Author(s)

Werner A. Stahel, ETH Zurich

See Also

pairs

Examples

```
PlotMatrix(iris[,1:4], main="Iris", pch=as.numeric(iris[,"Species"]))
```

PlotMonth *Plot Monthly or Seasonal Effects Of a Univariate Time Series*

Description

Plot monthly or seasonal effects of a univariate time series

Usage

```
PlotMonth(x, type = "l", labels, xlab = "", ylab = deparse(substitute(x)), ...)
```

Arguments

x	univariate time series
type	todo
labels	todo
xlab	a title for the x axis: see title.
ylab	a title for the y axis: see title.
...	the dots are passed to the plot command.

Details

todo

Author(s)

Markus Huerzeler, ETH Zurich

See Also

ts

Examples

```
PlotMonth(AirPassengers)
```

PlotMultiDens *Plot Multiple Density Curves*

Description

Multiple density curves are plotted on the same plot. The function plots the density curves in the defined colors and linetypes, after having calculated the globally appropriate xlim- and ylim-values. A legend can directly be included.

Usage

```
PlotMultiDens(x, ...)

## Default S3 method:
PlotMultiDens(x, xlim = NULL, ylim = NULL,
              col = rainbow(length(x)),
              lty = "solid", lwd = 1, xlab = "x", ylab = "density",
              args.dens = NULL, args.legend = NULL,
              na.rm = FALSE, flipxy=FALSE, ...)

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

Arguments

x	a list of vectors whose densities are to be plotted. Use split to separate a vector by groups. (See examples)
xlim, ylim	xlim, ylim of the plot.
col	colors of the lines, defaults to rainbow(1:length(x)).
lty	line type of the lines.
lwd	line widths for the lines.
xlab, ylab	a title for the x, resp. y axis. Defaults to "x" and "density".
args.dens	list of additional arguments to be passed to the density function. If set to NULL the defaults will be used. Those are n = 4096 (2^12) and kernel = "epanechnikov".
args.legend	list of additional arguments to be passed to the legend function. Use args.legend = NA if no legend should be added.
na.rm	should NAs be omitted? Defaults to FALSE.
flipxy	logical, should x- and y-axis be flipped? Defaults to FALSE.

formula	a formula of the form lhs ~ rhs where lhs gives the data values and rhs the corresponding groups.
data	an optional matrix or data frame (or similar: see model.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.
...	the dots are passed to plot(...).

Details

All style arguments, density arguments and data list elements will be recycled if necessary.
The flipxy parameter flips x an y-values, so as to plot density curves on the x-axis.

Value

data.frame with 3 columns, containing the bw, n and kernel parameters used for the list elements.
The number of rows correspond to the length of the list x.

Note

Consider using:

```
library(lattice)
densityplot( ~ delivery_min | driver, data=d.pizza)
```

as alternative when not all curves should be plotted in the same plot.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

PlotViolin, density

Examples

```
x <- rnorm(1000,0,1)
y <- rnorm(1000,0,2)
z <- rnorm(1000,2,1.5)

# the input of the following function MUST be a numeric list
PlotMultiDens(list(x=x,y=y,z=z))


PlotMultiDens( x=split(d.pizza$delivery_min, d.pizza$driver), na.rm=TRUE
  , main="delivery time ~ driver", xlab="delivery time [min]", ylab="density"
  , lwd=1:7, lty=1:7
  , panel.first=grid())
# this example demonstrates the definition of different line types and -colors
# an is NOT thought as recommendation for good plotting practice... :-)
```

```
# the formula interface
PlotMultiDens(delivery_min ~ driver, data=d.pizza)

# recyling of the density parameters
res <- PlotMultiDens(x=split(d.pizza$temperature, d.pizza$driver),
             args.dens = list(bw=c(5,2), kernel=c("rect","epanechnikov")), na.rm=TRUE)
res

# compare bandwidths
PlotMultiDens(x=split(d.pizza$temperature, d.pizza$driver)[1],
                     args.dens = list(bw=c(1:5)), na.rm=TRUE,
                     args.legend=NA, main="Compare bw")
legend(x="topright", legend=gettextf("bw = %s", 1:5), fill=rainbow(5))
```

`PlotPolar` *Plot Values on a Circular Grid*

Description

`PlotPolar` creates a polar coordinate plot of the radius r in function of the angle theta. 0 degrees is drawn at the 3 o'clock position and angular values increase in a counterclockwise direction.

Usage

```
PlotPolar(r, theta = NULL, type = "p", rlim = NULL, main = "", lwd = par("lwd"),
          lty = par("lty"), col = par("col"), pch = par("pch"), fill = NA,
          cex = par("cex"), mar = c(2, 2, 5, 2), add = FALSE, ...)
```

Arguments

`r`	a vector of radial data.
`theta`	a vector of angular data specified in radians.
`type`	one out of `c("p","l","h")`, the plot type, defined following the definition in plot type. `"p"` means points, `"l"` will connect the points with lines and `"h"` is used to plot radial lines from the center to the points. Default is `"p"`.
`rlim`	the r limits (r1, r2) of the plot
`main`	a main title for the plot, see also `title`.
`lwd`	a vector of line widths, see `par`.
`lty`	a vector of line types, see `par`.
`col`	The colors for lines and points. Multiple colors can be specified so that each point can be given its own color. If there are fewer colors than points they are recycled in the standard fashion. Lines will all be plotted in the first colour specified.
`pch`	a vector of plotting characters or symbols: see `points`.
`fill`	fill color, defaults to NA (none).

cex	a numerical vector giving the amount by which plotting characters and symbols should be scaled relative to the default. This works as a multiple of par("cex"). NULL and NA are equivalent to 1.0.
mar	A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot.
add	defines whether points should be added to an existing plot.
...	further arguments are passed to the plot command.

Details

The function is rather flexible and can produce quite a lot of of different plots. So is it also possible to create spider webs or radar plots.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

PolarGrid

Examples

```
testlen <- c(sin(seq(0, 1.98*pi, length=100))+2+rnorm(100)/10)
testpos <- seq(0, 1.98*pi, length=100)

PlotPolar(testlen, testpos, type="l", main="Test Polygon", col="blue")
PolarGrid(ntheta=9, col="grey", lty="solid", lblradians=TRUE)

# start at 12 o'clock and plot clockwise
PlotPolar(testlen, -(testpos - pi/2), type="p", main="Test Polygon",
          col="green", pch=16)

PolarGrid(ntheta = rev(seq(0, 2*pi, by=2*pi/9) + pi/2),
          alabels=FormatFix(seq(0, 2*pi, by=2*pi/9),2)[-10], col="grey",
          lty="solid", lblradians=TRUE)


# just because of it's beauty
t <- seq(0,2*pi,0.01)
PlotPolar( r=sin(2*t)*cos(2*t), theta=t, type="l", lty="dashed", col="red" )
PolarGrid()


# use some filled polygons
ions <- c(3.2,5,1,3.1,2.1,5)
ion.names <- c("Na","Ca","Mg","Cl","HCO3","SO4")

PlotPolar(r = ions, type="l", fill="yellow")

# the same, but let's have a grid first
PlotPolar(r = ions, type="l", lwd=2, col="blue", main="Ions",
          panel.first=PolarGrid(nr=seq(0, 6, 1)) )
```

```
# leave the radial grid out
PlotPolar(r = ions, type="l", fill="yellow")
PolarGrid(nr = NA, ntheta = length(ions), alabels = ion.names,
          col = "grey", lty = "solid" )

# display radial lines
PlotPolar(r = ions, type="h", col="blue", lwd=3)
# add some points
PlotPolar(r = ions, type="p", pch=16, add=TRUE, col="red", cex=1.5)

# spiderweb (not really recommended...)
posmat <- matrix(sample(2:9,30,TRUE),nrow=3)
PlotPolar(posmat, type="l", main="Spiderweb plot", col=2:4, lwd=1:3)
PolarGrid(nr=NA, ntheta=ncol(posmat), alabels=paste("X", 1:ncol(posmat), sep=""),
          col="grey", lty="solid" )

# example from: The grammar of graphics (L. Wilkinson)
data("UKgas")
m <- matrix(UKgas, ncol=4, byrow=TRUE)
cols <- c(SetAlpha(rep("green", 10), seq(0,1,0.1)),
          SetAlpha(rep("blue", 10), seq(0,1,0.1)),
          SetAlpha(rep("orange", 10), seq(0,1,0.1)))

PlotPolar(r=m, type="l", col=cols, lwd=2 )
PolarGrid(ntheta=4, alabels=c("Winter","Spring","Summer","Autumn"), lty="solid")
legend(x="topright", legend=c(1960,1970,1980), fill=c("green","blue","orange"))


# radarplot (same here, consider alternatives...)
data(mtcars)
d.car <- scale(mtcars[1:6,1:7], center=FALSE)

# let's have a palette with transparent colors (alpha = 32)
cols <- SetAlpha(colorRampPalette(c("red","yellow","blue"), space = "rgb")(6), 0.25)
PlotPolar(d.car, type="l", fill=cols, main="Cars in radar")
PolarGrid(nr=NA, ntheta=ncol(d.car), alabels=colnames(d.car), lty="solid", col="black")
```

`PlotPyramid` *Draw a Back To Back Pyramid Plot*

Description

Pyramid plots are a common way to display the distribution of age groups.

Usage

```
PlotPyramid(lx, rx = NA, ylab = "", ylab.x = 0,
            col = c("red", "blue"), border = par("fg"),
            main = "", lxlab = "", rxlab = "",
            xlim = NULL, gapwidth = NULL,
            xaxt = TRUE, args.grid = NULL, cex.axis = par("cex.axis"),
```

```
cex.lab = par("cex.axis"), cex.names = par("cex.axis"), adj = 0.5, ...)
```

Arguments

lx	either a vector or matrix of values describing the bars which make up the plot. If lx is a vector, it will be used to construct the left barplot. If lx is a matrix the first column will be plotted to the left side and the second to the right side. Other columsn are ignored.
rx	a vector with the values used to build the right barplot. lx and rx should be of equal length.
ylab	a vector of names to be plotted either in the middle or at the left side of the plot. If this argument is omitted, then the names are taken from the names attribute of lx if this is a vector.
ylab.x	the x-position of the y-labels.
col	the color(s) of the bars. If there are more than one the colors will be recycled.
border	the border color of the bars. Set this to NA if no border is to be plotted.
main	overall title for the plot.
lxlab	a label for the left x axis.
rxlab	a label for the right x axis.
xlim	limits for the x axis. The first value will determine the limit on the left, the second the one on the right.
gapwidth	the width of a gap in the middle of the plot. If set to 0, no gap will be plotted. Default is NULL which will make the gap as wide, as it is necessary to plot the longest ylab.
xaxt	a character which specifies the x axis type. Specifying "n" suppresses plotting of the axis.
args.grid	list of additional arguments for the grid. Set this argument to NA if no grid should be drawn.
cex.axis	expansion factor for numeric axis labels.
cex.lab	expansion factor for numeric variable labels.
cex.names	expansion factor for y labels (names).
adj	one or two values in [0, 1] which specify the x (and optionally y) adjustment of the labels.
...	the dots are passed to the barplot function.

Details

Pyramid plots are a common way to display the distribution of age groups in a human population. The percentages of people within a given age category are arranged in a barplot, typically back to back. Such displays can be used to distinguish males vs. females, differences between two different countries or the distribution of age at different timepoints. The plot type can also be used to display other types of opposed bar charts with suitable modification of the arguments.

Value

A numeric vector giving the coordinates of all the bar midpoints drawn, useful for adding to the graph.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

barplot

Examples

```
d.sda <- data.frame(
  kt_x =  c("ZH","BL","ZG","SG","LU","AR","SO","GL","SZ",
            "NW","TG","UR","AI","OW","GR","BE","SH","AG",
            "BS","FR","GE","JU","NE","TI","VD","VS"),
  apo_n = c(18,16,13,11,9,12,11,8,9,8,11,9,7,9,24,19,
            19,20,43,27,41,31,37,62,38,39),
  sda_n = c(235,209,200,169,166,164,162,146,128,127,
            125,121,121,110,48,34,33,0,0,0,0,0,0,0,0,0)
)

PlotPyramid(lx=d.sda[,c("apo_n","sda_n")], ylab=d.sda$kt_x,
            col=c("lightslategray", "orange2"), border = NA, ylab.x=0,
            xlim=c(-110,250),
            gapwidth = NULL, cex.lab = 0.8, cex.axis=0.8, xaxt = TRUE,
            lxlab="Drugstores", rxlab="General practitioners",
            main="Density of general practitioners and drugstores in CH (2010)",
            space=0.5, args.grid=list(lty=1))


par(mfrow=c(1,3))

m.pop<-c(3.2,3.5,3.6,3.6,3.5,3.5,3.9,3.7,3.9,3.5,
         3.2,2.8,2.2,1.8,1.5,1.3,0.7,0.4)
f.pop<-c(3.2,3.4,3.5,3.5,3.5,3.7,4,3.8,3.9,3.6,3.2,
         2.5,2,1.7,1.5,1.3,1,0.8)
age <- c("0-4","5-9","10-14","15-19","20-24","25-29",
         "30-34","35-39","40-44","45-49","50-54",
         "55-59","60-64","65-69","70-74","75-79","80-44","85+")

PlotPyramid(m.pop, f.pop,
            ylab = age, space = 0, col = c("cornflowerblue", "indianred"),
            main="Age distribution at baseline of HELP study",
            lxlab="male", rxlab="female" )

PlotPyramid(m.pop, f.pop,
            ylab = age, space = 0, col = c("cornflowerblue", "indianred"),
            xlim=c(-5,5),
            main="Age distribution at baseline of HELP study",
            lxlab="male", rxlab="female", gapwidth=0, ylab.x=-5 )


PlotPyramid(c(1,3,5,2,0.5), c(2,4,6,1,0),
            ylab = LETTERS[1:5], space = 0.3, col = rep(rainbow(5), each=2),
            xlim=c(-10,10), args.grid=NA, cex.names=1.5, adj=1,
            lxlab="Group A", rxlab="Group B", gapwidth=0, ylab.x=-8, xaxt="n")
```

PlotQQ *QQ-Plot for Optional Distributions*

Description

Create a QQ-plot for a variable which is not normally distributed. The assumed underlying distribution can freely be defined.

Usage

```
PlotQQ(x, qdist, ..., main = NULL, xlab = NULL, ylab = NULL)
```

Arguments

x	the data sample
qdist	the quantile function of the assumed distribution as text or as function.
...	optional arguments to be passed to the quantile function
main	the main title for the plot. This will be "Q-Q-Plot" by default
xlab	the xlab for the plot
ylab	the ylab for the plot

Details

The function generates a sequence of points between 0 and 1 and transforms those into quantiles by means of the defined assumed distribution.

Note

The code is based on the tip 10.22 "Creating Ohrer Quantile-Quantile Plots" from R Cookbook.

Author(s)

Andri Signorell <andri@signorell.net>

References

Teetor, P. (2011) *R Cookbook*. O'Reilly, pp. 254-255.

See Also

qqnorm, qqline, qqplot

Examples

```
y <- rexp(100, 1/10)
PlotQQ(y, qexp, rate=1/10)
```

PlotRCol *Information plots*

Description

The function `PlotPar()` plots the typically used parameters and their values.
`PlotRCol()` plots the R-colors in a dense manner.

Usage

```
PlotRCol(ord = c("hsv", "default"), label = c("text", "hex", "dec"))
PlotPar()
PlotMar()
```

Arguments

`ord`	the order of the colors, can be either defined by hsv-value or by the R internal color-number
`label`	label for the colors, can be the colorname (text), the hex-code (#rrggbb) or the decimal RGB-number

Details

`PlotMar()` should plot the margins, but waits for its implementation...

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`par`, `colors`

Examples

```
PlotPar()

PlotRCol(ord="hsv")
PlotRCol(label="hex")
```

PlotTreemap *Create a Treemap*

Description

Creates a treemap where rectangular regions of different size, color, and groupings visualize the elements.

Usage

```
PlotTreemap(x, grp = NULL, labels = NULL, cex = 1, text.col = "black",
            col = rainbow(length(x)), labels.grp = NULL, cex.grp = 3,
            text.col.grp = "black", border.grp = "grey50",
            lwd.grp = 5, main = "")
```

Arguments

x	a vector storing the values to be used to calculate the areas of rectangles.
grp	a vector specifying the group (i.e. country, sector, etc.) to which each element belongs.
labels	a vector specifying the labels.
cex	the character extension for the area labels. Default is 1.
text.col	the text color of the area labels. Default is "black".
col	a vector storing the values to be used to calculate the color of rectangles.
labels.grp	a character vector specifying the labels for the groups.
cex.grp	the character extension for the group labels. Default is 3.
text.col.grp	the text color of the group labels. Default is "black".
border.grp	the border color for the group rectangles. Default is "grey50". Set this to NA if no special border is desired.
lwd.grp	the linewidth of the group borders. Default is 5.
main	a title for the plot.

Details

A treemap is a two-dimensional visualization for quickly analyzing large, hierarchical data sets. Treemaps are unique among visualizations because they provide users with the ability to see both a high level overview of data as well as fine-grained details. Users can find outliers, notice trends, and perform comparisons using treemaps. Each data element contained in a treemap is represented with a rectangle, or a cell. Treemap cell arrangement, size, and color are each mapped to an attribute of that element. Treemap cells can be grouped by common attributes. Within a group, larger cells are placed towards the bottom left, and smaller cells are placed at the top right.

Value

returns a list with groupwise organized midpoints in x and y for the rectangles within a group and for the groups themselves.

Author(s)

Andri Signorell <andri@signorell.net>, strongly based on code from Jeff Enos <jeff@kanecap.com>

See Also

PlotCirc, mosaicplot, barplot

Examples

```
set.seed(1789)
N <- 20
area <- rlnorm(N)

PlotTreemap(x=sort(area, decreasing=TRUE), labels=letters[1:20], col=PalRedToBlack(20))


grp <- sample(x=1:3, size=20, replace=TRUE, prob=c(0.2,0.3,0.5))

z <- Sort(data.frame(area=area, grp=grp), c("grp","area"), decreasing=c(FALSE,TRUE))
z$col <- SetAlpha(c("steelblue","green","yellow")[z$grp],
                  unlist(lapply(split(z$area, z$grp),
                  function(...) LinScale(..., newlow=0.1, newhigh=0.6))))

PlotTreemap(x=z$area, grp=z$grp, labels=letters[1:20], col=z$col)


b <- PlotTreemap(x=z$area, grp=z$grp, labels=letters[1:20], labels.grp=NA,
                 col=z$col, main="Treemap")

# the function returns the midpoints of the areas
# extract the group midpoints from b
mid <- do.call(rbind, lapply(lapply(b, "[", 1), data.frame))

# and draw some visible text
BoxedText( x=mid$grp.x, y=mid$grp.y, labels=LETTERS[1:3], cex=3, border=NA,
  col=SetAlpha("white",0.7) )
```

PlotVenn *Plot a Venn Diagram*

Description

This function produces Venn diagrams for up to 5 datasets.

Usage

```
PlotVenn(x, col = "transparent", plot = TRUE)
```

Arguments

x	the list with the sets to be analysed. Those can be factors or something coercable to a factor.
col	the colors for the sets on the plot.
plot	logical. Should a plot be produced or just the results be calculated.

Details

The function calculates the necessary frequencies and plots the venn diagram.

Value

a list with 2 elements, the first contains a table with the observed frequencies in the given sets. The second returns a data.frame with the xy coordinates for the labels in the venn diagram, the specific combination of factors and the frequency in that intersection area. The latter can be 0 as well.

Author(s)

Andri Signorell <andri@signorell.net>

References

Venn, J. (1880): On the Diagrammatic and Mechanical Representation of Propositions and Reasonings. *Dublin Philosophical Magazine and Journal of Science* 9 (59): 1-18.

Edwards, A.W.F. (2004): Cogwheels of the mind: the story of Venn diagrams. *JHU Press* ISBN 978-0-8018-7434-5.

Examples

```
element <- function() paste(sample(LETTERS, 5, replace=TRUE), collapse="")
group <- replicate(1000, element())

GroupA <- sample(group, 400, replace=FALSE)
GroupB <- sample(group, 750, replace=FALSE)
GroupC <- sample(group, 250, replace=FALSE)
GroupD <- sample(group, 300, replace=FALSE)

x <- list(GroupA, GroupB, GroupC, GroupD)
x

PlotVenn(x=x[1:2])
PlotVenn(x=x[1:3])
PlotVenn(x=x[1:4], col=SetAlpha(c("blue","red","yellow","green","lightblue"), 0.2))

r.venn <- PlotVenn(x=x[1:5], col=SetAlpha(c("blue","red","yellow","green","lightblue"), 0.2))
r.venn
```

PlotViolin *Plot Violins Instead of Boxplots*

Description

This function serves the same utility as side-by-side boxplots, only it provides more detail about the different distribution. It plots violins instead of boxplots. That is, instead of a box, it uses the density function to plot the density. For skewed distributions, the results look like "violins". Hence the name.

Usage

```
PlotViolin(x, ...)

## Default S3 method:
PlotViolin(x, ..., horizontal = FALSE, bw = "SJ", na.rm = FALSE,
           names = NULL, args.boxplot = NULL)

## S3 method for class 'formula'
PlotViolin(formula, data = NULL, ..., subset)
```

Arguments

x	Either a sequence of variable names, or a data frame, or a model formula
horizontal	logical indicating if the densityplots should be horizontal; default FALSE means vertical arrangement.
bw	the smoothing bandwidth (method) being used by density. bw can also be a character string giving a rule to choose the bandwidth. See bw.nrd. The default, has been switched from "nrd0" to "SJ", following the general recommendation in Venables & Ripley (2002). In case of a method, the average computed bandwidth is used.
na.rm	logical, should NAs be omitted? The density-function can't do with missings. Defaults to FALSE.
names	a vector of names for the groups.
formula	a formula, such as y ~ grp, where y is a numeric vector of data values to be split into groups according to the grouping variable grp (usually a factor).
data	a data.frame (or list) from which the variables in formula should be taken.
subset	an optional vector specifying a subset of observations to be used for plotting.
...	The dots are passed to polygon. Notably, you can set the color to red with col="red", and a border color with border="blue"
args.boxplot	list of arguments for a boxplot to be superposed to the densityplot. By default (NULL) a black boxplot will be drawn. Set this to NA to suppress the boxplot.

Value

If a boxplot was drawn then the function returns a list with the following components:

stats	a matrix, each column contains the extreme of the lower whisker, the lower hinge, the median, the upper hinge and the extreme of the upper whisker for one group/plot. If all the inputs have the same class attribute, so will this component.

n	a vector with the number of observations in each group.
conf	a matrix where each column contains the lower and upper extremes of the notch.
out	the values of any data points which lie beyond the extremes of the whiskers.
group	a vector of the same length as out whose elements indicate to which group the outlier belongs.
names	a vector of names for the groups.

Note

This function is based on violinplot (package **UsingR**). Some adaptions were made in the interface, such as to accept the same arguments as boxplot does. Moreover the function was extended by the option to have a boxplot superposed.

Author(s)

John Verzani, Andri Signorell <andri@signorell.net>

References

The code is based on the boxplot function from R/base.

See Also

boxplot, PlotMultiDens , density

Examples

```
# make a "violin"
x <- c(rnorm(100), rnorm(50,5))

PlotViolin(x, col = "brown")

par(mfrow=c(1,2))
f <- factor(rep(1:5, 30))
# make a quintet. Note also choice of bandwidth
PlotViolin(x ~ f, col = SetAlpha("steelblue",0.3), bw = "SJ", main="Vertical")

# and the same, but in horizontal arrangement
PlotViolin(x ~ f, col = SetAlpha("steelblue",0.3), bw = "SJ", horizontal = TRUE,
  las=1, main="Horizontal")


# example taken from boxplot
boxplot(count ~ spray, data = InsectSprays, col = "lightgray", main="Boxplot")
PlotViolin(count ~ spray, data = InsectSprays, col = "lightgray", main="Violinplot")


# groupwise densityplots defined the same way as in boxplot
boxplot(len ~ supp*dose, data = ToothGrowth,
        main = "Guinea Pigs' Tooth Growth",
        xlab = "Vitamin C dose mg", ylab = "tooth length",
        col=c("yellow", "orange"), lty=c(1,2)
)

b <- PlotViolin(len ~ supp*dose, data = ToothGrowth,
```

```
          main = "Guinea Pigs' Tooth Growth",
          xlab = "Vitamin C dose mg", ylab = "tooth length",
          col=c("yellow", "orange"), lty=c(1,2)
)
# use points, if the medians deserve special attention
points(x=1:6, y=b$stats[3,], pch=21, bg="white", col="black", cex=1.2)
```

`PlotWeb` *Plot a Web of Connected Points*

Description

This plot can be used to graphically display a correlation matrix by using the linewidth between the nodes in proportion to the correlation of two variables. It will place the elements homogenously around a circle and draw connecting lines between the points.

Usage

```
PlotWeb(m, col = c("red", "blue"), lty = par("lty"), args.legend=NULL,
        pch = 21, pt.cex = 2, pt.col = "black", pt.bg = "darkgrey", ...)
```

Arguments

`m`	a matrix with numeric values
`col`	the color for the connecting lines
`lty`	the line type for the connecting lines
`args.legend`	list of additional arguments to be passed to the `legend` function. Use `args.legend = NA` if no legend should be added.
`pch`	the plotting symbols appearing in the plot, as a non-negative numeric vector (see `points`, but unlike there negative values are omitted) or a vector of 1-character strings, or one multi-character string.
`pt.cex`	expansion factor(s) for the points.
`pt.col`	the foreground color for the points, corresponding to its argument `col`.
`pt.bg`	the background color for the points, corresponding to its argument `bg`.
`...`	dots are passed to plot.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`PlotCorr`

Examples

```
m <- cor(d.pizza[,WhichNumerics(d.pizza)[-c(1:2)]], use="pairwise.complete.obs")
PlotWeb(m=m, col=c("red","blue"), main="Pizza Correlation")
```

PoissonCI *Poisson Confidence Interval*

Description

Computes the confidence intervals of a poisson distributed variable's lambda. Several methods are implemented, see details.

Usage

```
PoissonCI(x, n = 1, conf.level = 0.95, method = c("exact", "score", "wald"))
```

Arguments

x	number of events.
n	time base for event count.
conf.level	confidence level, defaults to 0.95.
method	character string specifing which method to use; can be one out of "wald", "score". Method can be abbreviated. See details. Defaults to "score".

Details

The Wald interval uses the asymptotic normality of the test statistic.

Value

A vector with 3 elements for estimate, lower confidence intervall and upper for the upper one.

Author(s)

Andri Signorell <andri@signorell.net>

References

Agresti, A. and Coull, B.A. (1998) Approximate is better than "exact" for interval estimation of binomial proportions. *American Statistician*, **52**, pp. 119-126.

Garwood, F. (1936) Fiducial Limits for the Poisson distribution. *Biometrika* 28:437-442.

http://www.ine.pt/revstat/pdf/rs120203.pdf

See Also

poisson.test, BinomCI, MultinomCI

Examples

```
# the horse kick example
count <- 0:4
deaths <- c(144, 91, 32, 11, 2)

n <- sum(deaths)
x <- sum(count * deaths)

lambda <- x/n

PoissonCI(x=x, n=n, method = c("exact","score", "wald"))

exp <- dpois(0:4, lambda) * n

barplot(rbind(deaths, exp * n/sum(exp)), names=0:4, beside=TRUE,
  col=c(hred, hblue), main = "Deaths from Horse Kicks", xlab = "count")
legend("topright", legend=c("observed","expected"), fill=c(hred, hblue),
  bg="white")


## SMR, Welsh Nickel workers
PoissonCI(x=137, n=24.19893)
```

PolarGrid *Plot a Grid in Polar Coordinates*

Description

PolarGrid adds a polar grid to an existing plot. The number of radial gridlines are set by ntheta and the tangential lines by nr.

Usage

```
PolarGrid(nr = NULL, ntheta = NULL, col = "lightgray", lty = "dotted", lwd = par("lwd"),
          rlabels = NULL, alabels = NULL, lblradians = FALSE)
```

Arguments

nr	number of circles. When NULL, as per default, the grid aligns with the tick marks on the corresponding default axis (i.e., tickmarks as computed by axTicks). When NA, no circular grid lines are drawn.
ntheta	number of radial grid lines. Defaults to 12 uniformly distributed between 0 and 2*pi (each pi/3).
col	character or (integer) numeric; color of the grid lines.
lty	character or (integer) numeric; line type of the grid lines.
lwd	non-negative numeric giving line width of the grid lines.
rlabels	the radius labels. Use NA if no labels should be to be added.
alabels	the labels for the angles, they are printed on a circle outside the plot. Use NA for no angle labels.
lblradians	logic, defines if angle labels will be in degrees (default) or in radians.

Details

This can be made better....

Author(s)

Andri Signorell <andri@signorell.net>

See Also

PlotPolar

Examples

```
Canvas(xlim=c(-5,5), xpd=TRUE)
PolarGrid()

Canvas(xlim=c(-5,5), xpd=TRUE)
PolarGrid(nr=0:5, ntheta=6)

Canvas(xlim=c(-5,5), xpd=TRUE)
PolarGrid(ntheta=36, rlabels=NA, lblradians=TRUE)
```

PpPlot *Add Slides, Insert Texts and Plots to PowerPoint*

Description

A couple of functions to get R-stuff into MS-Powerpoint. PpAddSlide inserts a new slide into the active presentation.
PpPlot inserts the active plot into PowerPoint. The image is transferred by saving the picture to a file in R and inserting the file in PowerPoint. The format of the plot can be selected, as well as crop options and the size factor for inserting.
PpText inserts a new textbox with given text and box properties.

Usage

```
PpAddSlide(pos = NULL, pp = getOption("lastPP"))

PpPlot(type = "png", crop = c(0, 0, 0, 0), picscale = 100, x = 1, y = 1,
       height = NA, width = NA, res=200, dfact=1.6, pp = getOption("lastPP"))

PpText(txt, x = 1, y = 1, height = 50, width = 100, fontname = "Calibri", fontsize = 18,
       bold = FALSE, italic = FALSE, col = "black", bg = "white",
       hasFrame = TRUE, pp = getOption("lastPP"))
```

Arguments

pos	position of the new inserted slide.
type	the format for the picture file, default is "png".
crop	crop options for the picture, defined by a 4-elements-vector. The first element is the bottom side, the second the left and so on.
picscale	scale factor of the picture in percent, default ist 100.
x, y	left/upper xy-coordinate for the plot or for the textbox.
height	height in cm, this overrides the picscale if both are given.
width	width in cm, this overrides the picscale if both are given.
res	resolution for the png file, defaults to 200.
dfact	the size factor for the graphic.
txt	text to be placed in the textbox
fontname	used font for textbox
fontsize	used fontsize for textbox
bold	logic. Text is set bold if this is set to TRUE (default is FALSE).
italic	logic. Text is set italic if this is to TRUE (default is FALSE).
col	font color, defaults to "black".
bg	background color for textboxdefaults to "white".
hasFrame	logical. Defines if a textbox is to be framed. Default is TRUE.
pp	the pointer to a PowerPoint instance, can be a new one, created by GetNewPP() or the last created by getOption("lastPP") (default).

Details

See PowerPoint-objectmodel for further informations.

Value

Returns a pointer to the inserted picture.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetNewPP, WrdPlot

Examples

```
## Not run: # Windows-specific example

# let's have some graphic
plot(1,type="n", axes=FALSE, xlab="", ylab="", xlim=c(0,1), ylim=c(0,1))
rect(0,0,1,1,col="black")
segments(x0=0.5, y0=seq(0.632,0.67, length.out=100),
  y1=seq(0.5,0.6, length.out=100), x1=1, col=rev(rainbow(100)))
polygon(x=c(0.35,0.65,0.5), y=c(0.5,0.5,0.75), border="white",
  col="black", lwd=2)
```

```
segments(x0=0,y0=0.52, x1=0.43, y1=0.64, col="white", lwd=2)
x1 <- seq(0.549,0.578, length.out=50)
segments(x0=0.43, y0=0.64, x1=x1, y1=-tan(pi/3)* x1 + tan(pi/3) * 0.93,
  col=rgb(1,1,1,0.35))


# get a handle to a new PowerPoint instance
pp <- GetNewPP()
# insert plot with a specified height
PpPlot(pp=pp,  x=150, y=150, height=10, width=10)

PpText("Remember?\n", fontname="Arial", x=200, y=70, height=30, fontsize=14,
       bold=TRUE, pp=pp, bg="lemonchiffon", hasFrame=TRUE)

PpAddSlide(pp=pp)
# crop the picture
pic <- PpPlot(pp=pp, x=1, y=200, height=10, width=10, crop=c(9,9,0,0))
pic


# some more automatic procedure
pp <- GetNewPP()
PpText("Hello to my presentation", x=100, y=100, fontsize=32, bold=TRUE,
       width=300, hasFrame=FALSE, col="blue", pp=pp)

for(i in 1:4){
  barplot(1:4, col=i)
  PpAddSlide(pp=pp)
  PpPlot(height=15, width=21, x=50, y=50, pp=pp)
  PpText(gettextf("This is my barplot nr %s", i), x=100, y=10, width=300, pp=pp)
}

## End(Not run)
```

`Primes` *Find all Primes Less Than n*

Description

Find all prime numbers aka 'primes' less than n.

Uses an obvious sieve method and some care, working with logical and integers to be quite fast.

Usage

```
Primes(n)
```

Arguments

`n`	a (typically positive integer) number.

Details

As the function only uses `max(n)`, `n` can also be a *vector* of numbers.

Value

numeric vector of all prime numbers $\leq n$.

Note

This function was previously published in the package **sfsmisc** as `primes` and has been integrated here without logical changes.

Author(s)

Bill Venables ($\leq n$); Martin Maechler gained another 40% speed, working with logicals and integers.

See Also

`Factorize`

Examples

```
(p1 <- Primes(100))
system.time(p1k <- Primes(1000)) # still lightning ..

stopifnot(length(p1k) == 168)
```

PtInPoly *Point in Polygon*

Description

`PtInPoly` works out if 2D points lie within the boundaries of a defined polygon.

Note: Points that lie on the boundaries of the polygon or vertices are assumed to be within the polygon.

Usage

```
PtInPoly(pnts, poly.pnts)
```

Arguments

`pnts`	a 2-column matrix or dataframe defining locations of the points of interest
`poly.pnts`	a 2-column matrix or dataframe defining the locations of vertices of the polygon of interest

Details

The algorithm implements a sum of the angles made between the test point and each pair of points making up the polygon. The point is interior if the sum is 2pi, otherwise, the point is exterior if the sum is 0. This works for simple and complex polygons (with holes) given that the hole is defined with a path made up of edges into and out of the hole.

This sum of angles is not able to consistently assign points that fall on vertices or on the boundary of the polygon. The algorithm defined here assumes that points falling on a boundary or polygon vertex are part of the polygon.

Value

A 3-column dataframe where the first 2 columns are the original locations of the points. The third column (names pip) is a vector of binary values where 0 represents points not with the polygon and 1 within the polygon.

Author(s)

Jeremy VanDerWal <jjvanderwal@gmail.com>

Examples

```
#define the points and polygon
pnts <- expand.grid(x=seq(1,6,0.1), y=seq(1,6,0.1))
polypnts <- cbind(x=c(2,3,3.5,3.5,3,4,5,4,5,5,4,3,3,3,2,2,1,1,1,1,2),
                  y=c(1,2,2.5,2,2,1,2,3,4,5,4,5,4,3,3,4,5,4,3,2,2) )

#plot the polygon and all points to be checked
plot(rbind(polypnts, pnts))
polygon(polypnts, col='#99999990')

#create check which points fall within the polygon
out <- PtInPoly(pnts, polypnts)
head(out)

#identify points not in the polygon with an X
points(out[which(out$pip==0), 1:2], pch='X')
```

Ray *Compact Information About the Columns of a Data Frame*

Description

An alternative description of a data.frame.

Usage

```
Ray(x)
```

Arguments

x	a data.frame to be described.

Details

I always missed the index of the variables in `str`...

Value

a list with the results

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`str`, `summary`

Examples

```
Ray(d.pizza)
```

Recode *Recode a Factor*

Description

Combining or rearranging a factor can be tedious if it has many levels. `Recode` supports this step by accepting a direct definition of newlevels by oldlevels as argument and adding an `"elselevel"` option.

Usage

```
Recode(x, newlevels, elselevel = NA, use.empty = FALSE)
```

Arguments

`x`	the factor whose levels are to be altered.
`newlevels`	a list with the oldlevels combined by `c()` and named with the name of the new level. See examples.
`elselevel`	how should the levels, which are not matched by newlevels' list be named. Set this to `NULL`, if elselevels should be left unchanged. Defaults to `NA`.
`use.empty`	logical. Defines, if newlevels which don't match with x should be left in the levels' list. The default is FALSE, which drops empty levels.

Value

the factor having the new levels applied.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

factor, levels
There's another possible solution in the package car.

Examples

```
x <- factor(sample(letters, 20))

y <- Recode( x, newlevels=list(
  "good" = c("a","b","c"),          # the old levels "a","b","c" get the new level "good"
  "bad"  = c("d","e","f"),         # the old levels "d","e","f" get the new level "bad" etc.
  "ugly" = c("g","h","k")), elselevel="other")

data.frame(x, y)



x <- factor(letters[1:6])

z1 <- Recode(x, newlevels=list("AB"=c("a","b"), "CD"=c("c","d")), elselevel="none of these")
z2 <- Recode(x, newlevels=list("AB"=c("a","b"), "CD"=c("c","d")), elselevel=NA)
z3 <- Recode(x, newlevels=list("AB"=c("a","b"), "CD"=c("c","d")), elselevel=NULL)
z4 <- Recode(x, newlevels=list("AB"=c("a","b"), "GH"=c("g","h")), elselevel=NA, use.empty=TRUE)
z5 <- Recode(x, newlevels=list("AB"=c("a","b"), "GH"=c("g","h")), elselevel=NA, use.empty=FALSE)

data.frame(z1, z2, z3, z4, z5)

# empty level GH in z4...
table(z4, useNA="ifany")
# but not in z5
table(z5, useNA="ifany")
```

RelRisk *Relative Risk*

Description

Computes the relative risk and it's confidence intervals. Confidence intervals are calculated using normal approximation ("wald"), ("score") or by using oddsratio ("use.or")

Usage

```
RelRisk(x, y = NULL, conf.level = NA,
        method = c("score", "wald", "use.or"), delta = 0.5, ...)
```

Arguments

x	x is a 2x2-table, with
y	NULL (default) or a vector with compatible dimensions to x. If y is provided, table(x, y, ...) is calculated.

conf.level	confidence level. Default is NA, meaning no confidence intervals will be reported.
method	method for calculating odds ratio and confidence interval. Can be one out of "score", "wald", "use.or". Default is "score".
delta	small constant to be added to the numerator for calculating the log risk ratio (Wald method). Usual choice is 0.5 although there does not seem to be any theory behind this. (Dewey, M. 2006)
...	further arguments are passed to the function table, allowing i.e. to set useNA.

Details

This function expects the following table structure:

```
                    disease=0   disease=1
exposed=0 (ref)       n00         n01
exposed=1             n10         n11
```

If the table to be provided is not in the preferred form, use the function Rev() to "reverse" the table rows, -columns, or both.

Value

If conf.level is not NA then the result will be a vector with 3 elements for estimate, lower confidence intervall and upper for the upper one. Else the relative risk will be reported as a single value.

Author(s)

Andri Signorell <andri@signorell.net>, based on code of Yongyi Min and Michael Dewey

References

Rothman, K. J. and Greenland, S. (1998) *Modern Epidemiology*. Lippincott-Raven Publishers

Rothman, K. J. (2002) *Epidemiology: An Introduction*. Oxford University Press

Jewell, N. P. (2004) *Statistics for Epidemiology*. 1st Edition, 2004, Chapman & Hall, pp. 73-81

Selvin, S. (1998) *Modern Applied Biostatistical Methods Using S-Plus*. 1st Edition, Oxford University Press

See Also

OddsRatio

Examples

```
mm <- cbind(c(9,20),c(41,29))
mm

RelRisk(t(mm), conf.level=0.95)
RelRisk(t(mm), conf.level=0.95, method="wald")
RelRisk(t(mm), conf.level=0.95, method="use.or")
```

Rename *Change Names of a Named Object*

Description

Rename changes the names of a named object.

Usage

```
Rename(x, ..., gsub = FALSE, fixed = TRUE, warn = TRUE)
```

Arguments

x	Any named object
...	A sequence of named arguments, all of type character
gsub	a logical value; if TRUE, gsub is used to change the row and column labels of the resulting table. That is, instead of substituting whole names, substrings of the names of the object can changed.
fixed	a logical value, passed to gsub. If TRUE, substitutions are by fixed strings and not by regular expressions.
warn	a logical value; should a warning be issued if those names to change are not found?

Details

This function changes the names of x according to the remaining arguments. If gsub is FALSE, argument tags are the *old* names, the values are the new names. If gsub is TRUE, arguments are substrings of the names that are substituted by the argument values.

Value

The object x with new names defined by the ... arguments.

Note

This function was previously published in the package **memisc** as rename and has been integrated here without logical changes.

Author(s)

Martin Elff <melff@essex.ac.uk>

See Also

Recode for recoding of a factor (renaming or combining levels)

Examples

```
x <- c(a=1, b=2)
Rename(x, a="A", b="B")

str(Rename( iris,
            Sepal.Length="Sepal_Length",
            Sepal.Width ="Sepal_Width",
            Petal.Length="Petal_Length",
            Petal.Width ="Petal_Width"
            ))

str(Rename(iris, .="_", gsub=TRUE))
```

Rev *Reverse Elements of a Vector or the Rows/Columns of Matrices and Tables*

Description

Rev provides a reversed version of its argument. It wraps the base function rev and provides an additional method for data.frames, matrices and tables, such as to reverse the order of rows and columns.

Usage

```
Rev(x, ...)

## S3 method for class 'matrix'
Rev(x, direction = c("row", "column", "both"), ...)

## S3 method for class 'table'
Rev(x, direction = c("row", "column", "both"), ...)

## S3 method for class 'data.frame'
Rev(x, direction = c("row", "column", "both"), ...)
```

Arguments

x	a data.frame, a matrix or a table to be reversed.
direction	defines the dimensions in which the elements are to be reversed. This can be any value out of c("row", "column", "both"), default being "row". If it is set to "both" then rows and columns are reversed.
...	the dots are passed to the matrix, resp. table interface.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

rev, order, sort, seq

Examples

```
tab <- matrix( c( 1, 11, 111,
                  2, 22, 222,
                  3, 33, 333), byrow=TRUE, nrow=3)

Rev(tab, direction="row")
Rev(tab, direction="column")

# abbreviations are accepted
Rev(tab, direction="b")
```

RgbToCol *Find the Named R-Color Which Is Nearest to a Given RGB-Color*

Description

Convert a RGB-color to a named R-Color means looking for a color in the R-palette, which is nearest to the given RGB-color. The function uses the minimum of squared distance as proximity measure.
RgbToLong converts an RGB-color to a long integer in numeric format.

Usage

```
RgbToCol(col, method = "rgb", metric = "euclidean")
RgbToLong(col)
```

Arguments

col	the color in rgb code, say a matrix with the red, green and blue code in the rows.
method	character string specifying the color space to be used. Can be "rgb" (default) or "hsv".
metric	character string specifying the metric to be used for calculating distances between the colors. Available options are "euclidean" (default) and "manhattan". Euclidean distances are root sum-of-squares of differences, and manhattan distances are the sum of absolute differences.

Value

the name of the nearest found R color.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

ColToRgb and the other conversion functions

Examples

```
RgbToCol(matrix(c(162,42,42), nrow=3))

RgbToLong(matrix(c(162,42,42), nrow=3))
```

RobRange *Robust Range*

Description

Determines a robust range of the data on the basis of the trimmed mean and variance.

Usage

```
RobRange(x, trim = 0.2, fac = 3, na.rm = FALSE)
```

Arguments

x	a vector of data.
trim	the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint. Default is 0.2.
fac	factor used for expanding the range, see Details. Default is 3.
na.rm	a logical value indicating whether NA values should be stripped before the computation proceeds.

Details

The function determines the trimmed mean m and then the "upper trimmed mean" s of absolute deviations from m, multiplied by fac. The robust minimum is then defined as m-fac*s or min(x), whichever is larger, and similarly for the maximum.

Value

The robust range.

Author(s)

Werner Stahel, ETH Zurich

See Also

RobScale

Examples

```
x <- c(rnorm(20), rnorm(3, 5, 20))
RobRange(x)

# compared to:
range(x)
```

RobScale *Robust Scaling With Median and Mad*

Description

RobScale is a wrapper function for robust standardization, using median and mad instead of mean and sd.

Usage

```
RobScale(x, center = TRUE, scale = TRUE)
```

Arguments

x	a numeric matrix(like object).
center	a logical value defining whether x should be centered by the median. Centering is done by subtracting the column medians (omitting NAs) of x from their corresponding columns. If center is FALSE, no centering is done.
scale	a logical value defining whether x should be scaled by the mad. Scaling is done by dividing the (centered) columns of x by their mad. If scale is FALSE, no scaling is done.

Value

the centered, scaled matrix. The numeric centering and scalings used (if any) are returned as attributes "scaled:center" and "scaled:scale"

Author(s)

Andri Signorell <andri@signorell.net>

See Also

scale, sweep

Examples

```
x <- d.pizza$temperature
plot(x=seq_along(x), y=RobScale(x), xlim=c(0,100))
points(x=seq_along(x), y=scale(x), col="red" )
```

Rotate *Rotate a Geometric Structure*

Description

Rotate a geometric structure by an angle theta around a centerpoint xy.

Usage

```
Rotate(x, y, mx = 0, my = 0, theta = pi/3)
```

Arguments

x, y	a vector of xy-coordinates of the geometric structure, which has to be rotated
mx,my	xy-coordinates of the center of the rotation.
theta	angle of the rotation

Value

The function invisibly returns a list of the coordinates for the rotated shape(s).

Author(s)

Andri Signorell <andri@signorell.net>

See Also

polygon, DrawRegPolygon, DrawEllipse, DrawArc, DrawAnnulus

Examples

```
# let's have a triangle
Canvas(main="Rotation")
x <- DrawRegPolygon(nv=3)[[1]]

# and rotate
sapply( (0:3) * pi/6, function(theta) {
  xy <- Rotate( x=x$x, y=x$y, theta=theta )
  polygon(xy$x, xy$y, col=SetAlpha("blue", 0.2))
} )

abline(v=0,h=0)
```

RunsTest *Runs Test for Randomness*

Description

This function performs the runs test for randomness.

Usage

```
RunsTest(x, alternative = c("two.sided", "less", "greater"), na.rm = FALSE)
```

Arguments

x	a dichotomous vector of data values.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "less" or "greater".
na.rm	defines if NAs are omitted. Default is FALSE.

Details

The runs test for randomness, also known as the Wald-Wolfowitz test, is used to test the hypothesis that a series of numbers is random.

For a categorical variable, the number of runs correspond to the number of times the category changes, that is, where xi belongs to one category and xi+1 belongs to the other. The number of runs, is the number of sign changes plus one.

For a numeric variable x, a run is a set of sequential values that are either all above or below a specified cutpoint, typically the median.

Value

A list with the following components.

statistic	the value of the standardized runs statistic.
p.value	the p-value for the test.
data.name	a character string giving the names of the data.
alternative	a character string describing the alternative hypothesis.

Author(s)

Andri Signorell <andri@signorell.net>

References

Wackerly, D., Mendenhall, W. Scheaffer, R. L. (1986): *Mathematical Statistics with Applications*, 3rd Ed., Duxbury Press, CA.

Wald, A. and Wolfowitz, J. (1940): On a test whether two samples are from the same population, *Ann. Math Statist*. 11, 147-162.

See Also

Run Length Encoding rle

Examples

```
# x has to be dichotomous for the test being
x <- c("S","S", "T", "S", "T","T","T", "S", "T")
RunsTest( x < median(x) )


x <- c(13, 3, 14, 14, 1, 14, 3, 8, 14, 17, 9, 14, 13, 2, 16, 1, 3, 12, 13, 14)
RunsTest( x < median(x) )

plot( (x < median(x)) - 0.5, type="s", ylim=c(-1,1) )
abline(h=0)

set.seed(123)
x <- sample(0:1, size=100, replace=TRUE)
RunsTest(x)
# As you would expect of values from a random number generator, the test fails to reject
# the null hypothesis that the data are random.
```

SampleTwins *Sample Twins*

Description

Draw a twin sample out of a population for a given recordset, by matching some strata criteria.

Usage

```
SampleTwins(data, stratanames = NULL, twins,
            method = c("srswor", "srswr", "poisson", "systematic"),
            pik, description = FALSE)
```

Arguments

data	the data to draw the sample from
stratanames	the stratanames to use
twins	the twin sample
method	method to select units; the following methods are implemented: simple random sampling without replacement (srswor), simple random sampling with replacement (srswr), Poisson sampling (poisson), systematic sampling (systematic); if "method" is missing, the default method is "srswor". See Strata.
pik	vector of inclusion probabilities or auxiliary information used to compute them; this argument is only used for unequal probability sampling (Poisson and systematic). If an auxiliary information is provided, the function uses the inclusion-probabilities function for computing these probabilities. If the method is "srswr" and the sample size is larger than the population size, this vector is normalized to one.

description — a message is printed if its value is TRUE; the message gives the number of selected units and the number of the units in the population. By default, the value is FALSE.

Value

The function produces an object, which contains the following information:

id — the identifier of the selected units.

stratum — the unit stratum.

prob — the final unit inclusion probability.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Strata, sample

Examples

```
m <- rbind(matrix(rep("nc",165), 165, 1, byrow=TRUE),
           matrix(rep("sc", 70), 70, 1, byrow=TRUE))
m <- cbind.data.frame(m, c(rep(1, 100), rep(2,50), rep(3,15),
                           rep(1,30), rep(2,40)), 1000*runif(235))
names(m) <- c("state","region","income")

# this would be our sample to be reproduced by a twin sample
d.smp <- m[sample(nrow(m), size=10, replace=TRUE),]

# draw the sample
s <- SampleTwins(data = m, stratanames=c("state","region"), twins = d.smp, method="srswor")

d.twin <- m[s$id,]
d.twin
```

SelectVarDlg *Select Elements of a Set by Click*

Description

SelectVarDlg is a GUI utility, which brings up a dialog and lets the user select elements (either variables of a data.frame or levels of a factor) by point and click in a listbox. The list of selected items is written to the clipboard so that the code can afterwards easily be pasted in the source file.

Usage

```
SelectVarDlg(x, ...)

## Default S3 method:
SelectVarDlg(x, useIndex = FALSE, ...)

## S3 method for class 'factor'
SelectVarDlg(x, ...)

## S3 method for class 'data.frame'
SelectVarDlg(x, ...)
```

Arguments

x	the object containing the elements to be selected. x can be a data.frame, a factor or any other vector.
useIndex	defines, if the enquoted names (default) or the index values should be returned.
...	further arguments to be passed to the default function.

Details

When working with big data.frames with many variables it is often tedious to build subsets by typing the columnnames. Here is where the function comes in offering a "point and click" approach for selecting the interesting columns. When x is a data.frame the columnnames are listed, when x is a factor the levels are listed and in all other cases the list is filled wiht the unique elements of x.

Value

A comma separated list with the selected values enquoted is returned invisibly as well as written to clipboard for easy inserting the text in an editor afterwards.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

select.list

Examples

```
## Not run:
data(d.pizza)
SelectVarDlg(x = d.pizza, T)
SelectVarDlg(x = d.pizza$driver )

x <- replicate(10, paste( sample(LETTERS, 5, replace = TRUE), collapse="") )
SelectVarDlg(x)
```

```
## End(Not run)
```

SetAlpha *Add an Alpha Channel To a Color*

Description

Add transparency to a color defined by its name or number. The function first converts the color to RGB and then appends the alpha channel.

Usage

```
SetAlpha(col, alpha = 0.5)
```

Arguments

col	vector of two kind of R colors, i.e., either a color name (an element of `colors()`) or an integer i meaning `palette()[i]`.
alpha	the alpha value to be added. This can be any value from 0 (fully transparent) to 1 (opaque). NA is interpreted so as to delete potentially Alpha channel. Default is 0.5.

Details

All arguments are recyled as necessary.

Value

Vector with the same length as `col`, giving the rgb-values extended by the alpha channel as hex-number (#rrggbbaa).

Author(s)

Andri Signorell <andri@signorell.net>

See Also

ColToHex, col2rgb

Examples

```
SetAlpha("yellow", 0.2)
SetAlpha(2, 0.5)   # red

Canvas(3)
DrawCircle(x=c(-1,0,1), y=c(1,-1,1), radius=2, col=SetAlpha(2:4, 0.4))

x <- rnorm(15000)
par(mfrow=c(1,2))
plot(x, type="p", col="blue" )
plot(x, type="p", col=SetAlpha("blue", .2), main="Better insight with alpha channel" )
```

`ShapiroFranciaTest` *Shapiro-Francia test for normality*

Description

Performs the Shapiro-Francia test for the composite hypothesis of normality.

Usage

```
ShapiroFranciaTest(x)
```

Arguments

`x`	a numeric vector of data values, the number of which must be between 5 and 5000. Missing values are allowed.

Details

The test statistic of the Shapiro-Francia test is simply the squared correlation between the ordered sample values and the (approximated) expected ordered quantiles from the standard normal distribution. The p-value is computed from the formula given by Royston (1993).

Value

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

`statistic`	the value of the Shapiro-Francia statistic.
`p.value`	the p-value for the test.
`method`	the character string "Shapiro-Francia normality test".
`data.name`	a character string giving the name(s) of the data.

Note

The Shapiro-Francia test is known to perform well, see also the comments by Royston (1993). The expected ordered quantiles from the standard normal distribution are approximated by `qnorm(ppoints(x, a = 3/8))`, being slightly different from the approximation `qnorm(ppoints(x, a = 1/2))` used for the normal quantile-quantile plot by `qqnorm` for sample sizes greater than 10.

Author(s)

Juergen Gross <gross@statistik.uni-dortmund.de>

References

Royston, P. (1993): A pocket-calculator algorithm for the Shapiro-Francia test for non-normality: an application to medicine. *Statistics in Medicine*, 12, 181–184.

Thode Jr., H.C. (2002): *Testing for Normality*. Marcel Dekker, New York. (2002, Sec. 2.3.2)

See Also

shapiro.test for performing the Shapiro-Wilk test for normality. AndersonDarlingTest, CramerVonMisesTest, LillieTest, PearsonTest for performing further tests for normality. qqnorm for producing a normal quantile-quantile plot.

Examples

```
ShapiroFranciaTest(rnorm(100, mean = 5, sd = 3))
ShapiroFranciaTest(runif(100, min = 2, max = 4))
```

SiegelTukeyTest *Siegel-Tukey Test for equality in variability*

Description

Non-parametric Siegel-Tukey test for equality in variability. The null hypothesis is that the variability of x is equal between two groups. A rejection of the null hypothesis indicates that variability differs between the two groups. SiegelTukeyRank returns the ranks, calculated after Siegel Tukey logic.

Usage

```
SiegelTukeyTest(x, ...)

## Default S3 method:
SiegelTukeyTest(x, y, adjust.median = FALSE,
                alternative = c("two.sided", "less", "greater"),
                mu = 0, exact = NULL, correct = TRUE, conf.int = FALSE,
                conf.level = 0.95, ...)

## S3 method for class 'formula'
SiegelTukeyTest(formula, data, subset, na.action, ...)


SiegelTukeyRank(x, g)
```

Arguments

x, y	numeric vector of data values. Non-finite (e.g. infinite or missing) values will be omitted.
adjust.median	Should between-group differences in medians be leveled before performing the test? In certain cases, the Siegel-Tukey test is susceptible to median differences and may indicate significant differences in variability that, in reality, stem from differences in medians.
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.
mu	a number specifying an optional parameter used to form the null hypothesis. See Details.

exact	a logical indicating whether an exact p-value should be computed. This is passed directly to wilcox.test.
correct	a logical indicating whether to apply continuity correction in the normal approximation for the p-value.
conf.int	a logical indicating whether a confidence interval should be computed.
conf.level	confidence level of the interval.
formula	a formula of the form lhs ~ rhs where lhs gives the data values and rhs the corresponding groups.
data	an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
g	a vector or factor object giving the group for the corresponding elements of x.
...	further arguments to be passed to or from methods.

Details

The Siegel-Tukey test has relatively low power and may, under certain conditions, indicate significance due to differences in medians rather than differences in variabilities (consider using the argument adjust.median). Consider also using mood.test or ansari.test.

Value

A list of class htest, containing the following components:

statistic	Siegel-Tukey test (Wilcoxon test on tie-adjusted Siegel-Tukey ranks, after the median adjustment if specified).
p.value	the p-value for the test
null.value	is the value of the median specified by the null hypothesis. This equals the input argument mu.
alternative	a character string describing the alternative hypothesis.
method	the type of test applied
data.name	a character string giving the names of the data.

Author(s)

Daniel Malter, Tal Galili <tal.galili@gmail.com>, Andri Signorell <andri@signorell.net>
published on: http://www.r-statistics.com/2010/02/siegel-tukey-a-non-parametric-test-for-equality-

References

Siegel, S., Tukey, J. W. (1960): A nonparametric sum of ranks procedure for relative spread in unpaired samples. *Journal of the American Statistical Association*.

Sheskin, D. J. (2004): *Handbook of parametric and nonparametric statistical procedures* 3rd edition. Chapman and Hall/CRC. Boca Raton, FL.

See Also

mood.test, ansari.test, wilcox.test, LeveneTest

Examples

```
# Duller, S. 183
x <- c(12, 13, 29, 30)
y <- c(15, 17, 18, 24, 25, 26)
SiegelTukeyTest(x, y)
SiegelTukeyTest(x, y, alternative="greater")

# Duller, S. 323
old <- c(870,930,935,1045,1050,1052,1055)
new <- c(932,970,980,1001,1009,1030,1032,1040,1046)
SiegelTukeyTest(old, new, alternative = "greater")
# compare to the recommended alternatives
mood.test(old, new, alternative="greater")
ansari.test(old, new, alternative="greater")

# Bortz, S. 250
x <- c(26.3,26.5,26.8,27.0,27.0,27.2,27.3,27.3,27.4,27.5,27.6,27.8,27.9)
id <- c(2,2,2,1,2,2,1,2,2,1,1,1,2)-1
SiegelTukeyTest(x ~ id)


# Sachs, Angewandte Statistik, 12. Auflage, 2007, S. 314
A <- c(10.1,7.3,12.6,2.4,6.1,8.5,8.8,9.4,10.1,9.8)
B <- c(15.3,3.6,16.5,2.9,3.3,4.2,4.9,7.3,11.7,13.1)
SiegelTukeyTest(A, B)



### 1
x <- c(4,4,5,5,6,6)
y <- c(0,0,1,9,10,10)
SiegelTukeyTest(x, y)

### 2
# example for a non equal number of cases:
x <- c(4,4,5,5,6,6)
y <- c(0,0,1,9,10)
SiegelTukeyTest(x, y)

### 3
x <- c(33, 62, 84, 85, 88, 93, 97, 4, 16, 48, 51, 66, 98)
id <- c(0,0,0,0,0,0,0,1,1,1,1,1,1)
SiegelTukeyTest(x ~ id)

### 4
x <- c(177,200,227,230,232,268,272,297,47,105,126,142,158,172,197,220,225,230,262,270)
id <- c(rep(0,8),rep(1,12))
SiegelTukeyTest(x ~ id, adjust.median=TRUE)

### 5
x <- c(33,62,84,85,88,93,97)
```

```
y <- c(4,16,48,51,66,98)
SiegelTukeyTest(x, y)

### 6
x <- c(0,0,1,4,4,5,5,6,6,9,10,10)
id <- c(0,0,0,1,1,1,1,1,1,0,0,0)
SiegelTukeyTest(x ~ id)

### 7
x <- c(85,106,96, 105, 104, 108, 86)
id <- c(0,0,1,1,1,1,1)
SiegelTukeyTest(x ~ id)
```

SignTest *Sign Test*

Description

Performs one- and two-sample sign tests on vectors of data.

Usage

```
SignTest(x, ...)

## Default S3 method:
SignTest(x, y = NULL, alternative = c("two.sided", "less", "greater"),
         mu = 0, conf.level = 0.95, ... )

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

Arguments

x	numeric vector of data values. Non-finite (e.g. infinite or missing) values will be omitted.
y	an optional numeric vector of data values: as with x non-finite values will be omitted.
mu	a number specifying an optional parameter used to form the null hypothesis. See Details.
alternative	is a character string, one of "greater", "less", or "two.sided", or the initial letter of each, indicating the specification of the alternative hypothesis. For one-sample tests, alternative refers to the true median of the parent population in relation to the hypothesized value of the median.
conf.level	confidence level for the returned confidence interval, restricted to lie between zero and one.
formula	a formula of the form lhs ~ rhs where lhs gives the data values and rhs the corresponding groups.

data	an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
...	further arguments to be passed to or from methods.

Details

The formula interface is only applicable for the 2-sample test.

SignTest computes a "Dependent-samples Sign-Test" if both x and y are provided. If only x is provided, the "One-sample Sign-Test" will be computed.

For the one-sample sign-test, the null hypothesis is that the median of the population from which x is drawn is mu. For the two-sample dependent case, the null hypothesis is that the median for the differences of the populations from which x and y are drawn is mu. The alternative hypothesis indicates the direction of divergence of the population median for x from mu (i.e., "greater", "less", "two.sided".)

The confidence levels are exact.

Value

A list of class htest, containing the following components:

statistic	the S-statistic (the number of positive differences between the data and the hypothesized median), with names attribute "S".
parameter	the total number of valid differences.
p.value	the p-value for the test.
null.value	is the value of the median specified by the null hypothesis. This equals the input argument mu.
alternative	a character string describing the alternative hypothesis.
method	the type of test applied.
data.name	a character string giving the names of the data.
conf.int	a confidence interval for the median.
estimate	the sample median.

Author(s)

Andri Signorell <andri@signorell.net>

References

Gibbons, J.D. and Chakraborti, S. (1992): *Nonparametric Statistical Inference*. Marcel Dekker Inc., New York.

Kitchens, L. J. (2003): *Basic Statistics and Data Analysis*. Duxbury.

Conover, W. J. (1980): *Practical Nonparametric Statistics, 2nd ed*. Wiley, New York.

See Also

t.test, wilcox.test, ZTest, binom.test, SIGN.test in the package **BSDA** (reporting approximative confidence intervals).

Examples

```
x <- c(1.83,  0.50,  1.62,  2.48, 1.68, 1.88, 1.55, 3.06, 1.30)
y <- c(0.878, 0.647, 0.598, 2.05, 1.06, 1.29, 1.06, 3.14, 1.29)

SignTest(x, y)
wilcox.test(x, y, paired = TRUE)


d.light <- data.frame(
  black = c(25.85,28.84,32.05,25.74,20.89,41.05,25.01,24.96,27.47),
  white <- c(18.23,20.84,22.96,19.68,19.5,24.98,16.61,16.07,24.59),
  d <- c(7.62,8,9.09,6.06,1.39,16.07,8.4,8.89,2.88)
)

d <- d.light$d

SignTest(x=d, mu = 4)
wilcox.test(x=d, mu = 4, conf.int = TRUE)

SignTest(x=d, mu = 4, alternative="less")
wilcox.test(x=d, mu = 4, conf.int = TRUE, alternative="less")

SignTest(x=d, mu = 4, alternative="greater")
wilcox.test(x=d, mu = 4, conf.int = TRUE, alternative="greater")


# test die interfaces
x <- runif(10)
y <- runif(10)
g <- rep(1:2, each=10)
xx <- c(x, y)

SignTest(x ~ group, data=data.frame(x=xx, group=g ))
SignTest(xx ~ g)
SignTest(x, y)

SignTest(x - y)
```

Skew *Skewness and Kurtosis*

Description

Skew computes the skewness, Kurt the kurtosis of the values in x.

Usage

```
Skew(x, na.rm = FALSE, method = 3, conf.level = NA, type = "bca", R = 1000)

Kurt(x, na.rm = FALSE, method = 3, conf.level = NA, type = "bca", R = 1000)
```

Arguments

x	a numeric vector, matrix or data frame. An object which is not a vector, matrix or data frame is coerced (if possible) by as.vector.
na.rm	logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.
method	integer out of 1, 2 or 3. Default ist 3. See Details.
conf.level	confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.
type	The type of confidence interval required. The value should be any subset of the values "norm", "basic", "stud", "perc", "bca") or simply "all" which will compute all five types of intervals.
R	The number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case R would be a vector of integers where each component gives the number of resamples from each of the rows of weights.

Details

If x is a matrix or a data frame, a vector of the skewness, resp. kurtosis, of the columns is returned.

If na.rm is TRUE then missing values are removed before computation proceeds.

The type of skewness can either be:

```
type = 1:   g_1 = m_3 / m_2^(3/2)
type = 2:   G_1 = g_1 * sqrt(n(n-1)) / (n-2)
type = 3:   b_1 = m_3 / s^3 = g_1 ((n-1)/n)^(3/2)
```

The type of kurtosis can either be:

```
type = 1:  g_2 = m_4 / m_2^2 - 3
type = 2:  G_2 = ((n+1) g_2 + 6) * (n-1) / ((n-2)(n-3))
type = 3:  b_2 = m_4 / s^4 - 3 = (g_2 + 3) (1 - 1/n)^2 - 3
```

type = 1 is the typical definition used in many older textbooks.
type = 2 is used in SAS and SPSS.
type = 3 is used in MINITAB and BMDP.

Value

For a data frame or for a matrix, a named vector with the appropriate method being applied column by column.

Note

Cramer et al. (1997) mention the asymptotic standard error of the skewness

```
ASE.skew = sqrt( 6n(n-1)/((n-2)(n+1)(n+3)) ), resp.
ASE.kurt = sqrt( (n^2 - 1)/((n-3)(n+5)) )
```

for the kurtosis, to be used for calculating the confidence intervals.
Joanes and Gill advise against that, pointing out that the normal assumptions would virtually always be violated. They suggest using the bootstrap method.

Author(s)

Andri Signorell <andri@signorell.net>, David Meyer <david.meyer@r-project.org> (type = 3)

References

Cramer, D. (1997): *Basic Statistics for Social Research* Routledge.

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

See Also

`mean`, `sd`, similar code in `library(e1071)`

Examples

```
data(d.pizza)

Skew(d.pizza$price, na.rm=TRUE)
Kurt(d.pizza$price, na.rm=TRUE)

# use sapply to calculate skewness for a data.frame
sapply(d.pizza[,c("temperature","price","delivery_min")], Skew, na.rm=TRUE)

# or apply to do that columnwise with a matrix
apply(as.matrix(d.pizza[,c("temperature","price","delivery_min")]), 2, Skew, na.rm=TRUE)
```

SomersDelta *Somers' Delta*

Description

Calculate Somers' Delta statistic, a measure of association for ordinal factors in a two-way table. The function has interfaces for a table, a matrix, a data.frame and for single vectors.

Usage

```
SomersDelta(x, y = NULL, direction = c("row", "column"), conf.level = NA, ...)
```

Arguments

x	a numeric vector, matrix or data.frame.
y	NULL (default) or a vector with compatible dimensions to x. If y is provided, table(x, y, ...) is calculated.
direction	direction of the calculation. Can be "row" (default) or "column", where "row" calculates Somers' D (R | C) ("column dependent").
conf.level	confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.
...	further arguments are passed to the function table, allowing i.e. to set useNA. This refers only to the vector interface.

Details

Somers' D(C | R) and Somers' D(R | C) are asymmetric modifications of tau-b. C | R indicates that the row variable x is regarded as the independent variable and the column variable y is regarded as dependent. Similarly, R | C indicates that the column variable y is regarded as the independent variable and the row variable x is regarded as dependent.
Somers' D differs from tau-b in that it uses a correction only for pairs that are tied on the independent variable. Somers' D is appropriate only when both variables lie on an ordinal scale.
The range of Somers' D is [-1, 1] and is computed as
D(C | R) = (P-Q)/(n^2 - sum(rowSums(tab)^2)), where P equals twice the number of concordances and Q twice the number of discordances.

Value

a single numeric value if no confidence intervals are requested
and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

References

Agresti, A. (2002) *Categorical Data Analysis*. John Wiley & Sons, pp. 57–59.

Goodman, L. A., & Kruskal, W. H. (1954) Measures of association for cross classifications. *Journal of the American Statistical Association*, 49, 732-764.

Somers, R. H. (1962) A New Asymmetric Measure of Association for Ordinal Variables, *American Sociological Review*, 27, 799–811.

Goodman, L. A., & Kruskal, W. H. (1963) Measures of association for cross classifications III: Approximate sampling theory. *Journal of the American Statistical Association*, 58, 310–364.

http://support.sas.com/onlinedoc/913/getDoc/en/statug.hlp/freq_sect18.htm
http://support.sas.com/onlinedoc/913/getDoc/en/statug.hlp/freq_sect20.htm

See Also

There's an implementation of Somers's D in Frank Harrell's **Hmisc** somers2, which is quite fast for large sample sizes. However it is restricted to computing Somers' Dxy rank correlation between a variable x and a binary (0-1) variable y.

ConDisPairs yields concordant and discordant pairs
other association measures:
GoodmanKruskalTauA (tau-a), cor (method="kendall") for tau-b, StuartTauC (tau-c), GoodmanKruskalGamma
Lambda, UncertCoef, MutInf

Examples

```
# example in:
# http://support.sas.com/documentation/cdl/en/statugfreq/63124/PDF/default/statugfreq.pdf
# pp. S. 1821

tab <- as.table(rbind(c(26,26,23,18,9),c(6,7,9,14,23)))

# Somers' D C|R
SomersDelta(tab, direction="column", conf.level=0.95)
# Somers' D R|C
SomersDelta(tab, direction="row", conf.level=0.95)
```

Sort *Sort a Vector, a Matrix, a Table or a Data.frame*

Description

Sort a vector, a matrix, a table or a data.frame. The base sort function does not have an interface for classes other than vectors and coerces the whole world to a vector. This means you get a sorted vector as result while passing a matrix to `sort`.
`Sort` wraps the base sort function and adds an interface for sorting the rows of the named 2-dimensional data structures by the order of one or more of its columns.

Usage

```
Sort(x, ...)

## Default S3 method:
Sort(x, ...)
## S3 method for class 'matrix'
Sort(x, ord = NULL, decreasing = FALSE, na.last = TRUE, ...)
## S3 method for class 'table'
Sort(x, ord = NULL, decreasing = FALSE, na.last = TRUE, ...)
## S3 method for class 'data.frame'
Sort(x, ord = NULL, decreasing = FALSE,
                factorsAsCharacter = TRUE, na.last = TRUE, ...)
```

Arguments

`x`	a numeric, complex. character or logical vector, a factor, a table or a data.frame to be sorted.
`decreasing`	logical. Should the sort be increasing or decreasing?

`factorsAsCharacter`
: logical. Should factors be sorted by the alphabetic order of their labels or by the order or their levels. Default is TRUE (by labels).

`ord`
: vector of integers or columnames. Defines the columns in a table, in a matrix or in a data.frame to be sorted for.
0 means row.names, 1:n the columns and n+1 the marginal sum. See examples.

`na.last`
: for controlling the treatment of `NA`s. If `TRUE`, missing values in the data are put last; if `FALSE`, they are put first; if `NA`, they are removed (see `order`.)

`...`
: further arguments to be passed to or from methods.

Details

The sort order for factors is the order of their levels (which is particularly appropriate for ordered factors), and usually confusing for unordered factors, whose levels may be defined in the sequence in which they appear in the data (which normally is unordered).

Value

the sorted object.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`sort`, `order`

Examples

```
data(d.pizza, package="DescTools")
d.frm <- d.pizza[1:10, c("driver","temperature","delivery_min")]

Sort(d.frm[,1])
# Sort follows the levels by default
levels(d.frm[,1])

Sort(x=d.frm, ord="driver", decreasing=FALSE)
# set factorsAsCharacter = TRUE, if alphabetical order is required
Sort(x=d.frm, ord="driver", decreasing=FALSE, factorsAsCharacter=TRUE)

Sort(x=d.frm, ord=c("driver","delivery_min"), factorsAsCharacter = TRUE)
Sort(x=d.frm, ord=c("driver","delivery_min"), factorsAsCharacter = FALSE)

Sort(x=d.frm, ord=c("driver","delivery_min"), decreasing=c(FALSE, TRUE),
  factorsAsCharacter = FALSE)

# Sorting tables
tab <- table(d.pizza$driver, d.pizza$area)

Sort(x=tab, ord=c(0,2), decreasing=c(TRUE, FALSE))
Sort(x=tab, ord=2, decreasing=TRUE)

# partial matching ok:
```

```
Sort(tab, o=1, d=TRUE)
```

SpearmanRho *Spearman Rank Correlation*

Description

Calculate Spearman correlation coefficient and it's confidence interval.

Usage

```
SpearmanRho(x, y = NULL, use = c("everything", "all.obs", "complete.obs",
            "na.or.complete","pairwise.complete.obs"),
            conf.level = NA)
```

Arguments

x	a numeric vector, an ordered factor, matrix or data frame. An ordered factor will be coerced to numeric.
y	NULL (default) or a vector, an ordered factor, matrix or data frame with compatible dimensions to x. An ordered factor will be coerced to numeric.
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs".
conf.level	confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.

Details

The function calculates Spearman's rho statistic by means of `cor(..., method="spearman")`. The confidence intervals are calculated via z-Transformation.

Value

Either a single numeric value, if no confidence interval is required,
or a vector with 3 elements for estimate, lower and upper confidence intervall.

Author(s)

Andri Signorell <andri@signorell.net>

References

Conover W. J. (1999) *Practical Nonparametric Statistics (3rd edition)*. Wiley

See Also

cor

Examples

```
SpearmanRho(d.diamonds$clarity, d.diamonds$cut)

SpearmanRho(d.diamonds$clarity, d.diamonds$cut, conf.level = 0.95)
```

`split.formula` *Formula Interface for Split*

Description

Implementation of a simple formula interface for the `split` function.

Usage

```
## S3 method for class 'formula'
 split(x, f, drop = FALSE, data = NULL, ...)
```

Arguments

`x`	a formula of the form `y ~ x`.
`f`	a 'factor' in the sense that `as.factor(f)` defines the grouping, or a list of such factors in which case their interaction is used for the grouping.
`drop`	logical indicating if levels that do not occur should be dropped (if `f` is a `factor` or a list). Defaults to FALSE.
`data`	the data frame from which the formula should be evaluated.
`...`	other arguments to be passed to `split`.

Author(s)

Andri Signorell <andri@signorell>

See Also

`split`

Examples

```
split(extra ~ group, data = sleep)
```

SpreadOut *Spread out a vector of numbers to a minimum interval*

Description

Spread out a vector of numbers so that there is a minimum interval between any two numbers when in ascending or descending order.

Usage

```
SpreadOut(x, mindist)
```

Arguments

x	a numeric vector which may contain NAs.
mindist	the minimum interval between any two values when in ascending or descending order.

Details

SpreadOut starts at or near the middle of the vector and increases the intervals between the ordered values. NAs are preserved. SpreadOut first tries to spread groups of values with intervals less than mindist out neatly away from the mean of the group. If this doesn't entirely succeed, a second pass that forces values away from the middle is performed.

SpreadOut is currently used to avoid overplotting of axis tick labels where they may be close together.

Value

On success, the spread out values. If there are less than two valid values, the original vector is returned.

Note

This function is borrowed from the package **plotrix** (SpreadOut) and has been integrated here without logical changes.

Author(s)

Jim Lemon

Examples

```
SpreadOut(c(1, 3, 3, 3, 3, 5), 0.2)
SpreadOut(c(1, 2.5, 2.5, 3.5, 3.5, 5), 0.2)
SpreadOut(c(5, 2.5, 2.5, NA, 3.5, 1, 3.5, NA), 0.2)

# this will almost always invoke the brute force second pass
SpreadOut(rnorm(10), 0.5)
```

Str *Compactly Display the Structure of an Arbitrary R Object*

Description

Just a wrapper for `str` with the variables of a data.frame enumerated.

Usage

```
Str(x, ...)
```

Arguments

`x`	any R object about which you want to have some information.
`...`	all the dots are passed to `str`.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`str`

Examples

```
Str(d.pizza)
```

StrAbbr *String Abbreviation*

Description

Abbreviate a character vector. The function includes starting from the first character as many characters as there are needed to result in a vector of unique values.

Usage

```
StrAbbr(x, minchar = 1, method = c("left", "fix"))
```

Arguments

`x`	character vector to be abbreviated
`minchar`	integer, minimal number of characters for the abbreviations.
`method`	one out of `left` or `fix`. While `left` restricts the result to as many characters are needed to ensure uniqueness, does `fix` yield a vector with all the elements being as long, as the the longest needed substring for differentiating the terms.

Value

The abbreviated strings.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

abbreviate, StrTrunc, StrTrim

Examples

```
StrAbbr(x=levels(d.pizza$driver), minchar=2)
StrAbbr(x=levels(d.pizza$driver), minchar=2, method="left")
StrAbbr(x=levels(d.pizza$driver), minchar=2, method="fix")


x <- c("Aaron", "Aaramis", "Berta", "Bello", "Claudia", "Cardinale", "Doretta", "Emilia")
StrAbbr(x, minchar=2, method="left")
StrAbbr(x, minchar=2, method="fix")
```

Strata *Stratified sampling*

Description

Stratified sampling with equal/unequal probabilities.

Usage

```
Strata(data, stratanames = NULL, size,
       method = c("srswor", "srswr", "poisson", "systematic"),
       pik, description = FALSE)
```

Arguments

data	data frame or data matrix; its number of rows is N, the population size.
stratanames	vector of stratification variables.
size	vector of stratum sample sizes (in the order in which the strata are given in the input data set).
method	method to select units; the following methods are implemented: simple random sampling without replacement (srswor), simple random sampling with replacement (srswr), Poisson sampling (poisson), systematic sampling (systematic); if "method" is missing, the default method is "srswor".
pik	vector of inclusion probabilities or auxiliary information used to compute them; this argument is only used for unequal probability sampling (Poisson and systematic). If an auxiliary information is provided, the function uses the inclusion-probabilities function for computing these probabilities. If the method is "srswr" and the sample size is larger than the population size, this vector is normalized to one.

`description` a message is printed if its value is TRUE; the message gives the number of selected units and the number of the units in the population. By default, the value is FALSE.

Details

The data should be sorted in ascending order by the columns given in the stratanames argument before applying the function. Use, for example, `data[order(data$state, data$region), ]`.

Value

The function produces an object, which contains the following information:

`id`	the identifier of the selected units.
`stratum`	the unit stratum.
`prob`	the final unit inclusion probability.

Note

This function has been taken from the library **sampling** without logical changes.

Author(s)

Yves Tille <yves.tille@unine.ch>, Alina Matei <alina.matei@unine.ch>

See Also

`sample`

Examples

```
# Example from An and Watts (New SAS procedures for Analysis of Sample Survey Data)
# generates artificial data (a 235X3 matrix with 3 columns: state, region, income).
# the variable "state" has 2 categories ('nc' and 'sc').
# the variable "region" has 3 categories (1, 2 and 3).
# the sampling frame is stratified by region within state.
# the income variable is randomly generated

m <- rbind(matrix(rep("nc",165), 165, 1, byrow=TRUE),
           matrix(rep("sc", 70), 70, 1, byrow=TRUE))
m <- cbind.data.frame(m, c(rep(1, 100), rep(2,50), rep(3,15),
                      rep(1,30), rep(2,40)), 1000*runif(235))
names(m) <- c("state","region","income")

# computes the population stratum sizes
table(m$region, m$state)

# not run
#     nc  sc
#  1 100  30
#  2  50  40
#  3  15   0
# there are 5 cells with non-zero values
# one draws 5 samples (1 sample in each stratum)
# the sample stratum sizes are 10,5,10,4,6, respectively
```

```
# the method is 'srswor' (equal probability, without replacement)

s <- Strata(m, c("region","state"), size=c(10,5,10,4,6), method="srswor")

# extracts the observed data
data.frame(income=m[s$id, "income"], s)

# see the result using a contigency table
table(s$region, s$state)


# The same data as in Example 1
# the method is 'systematic' (unequal probability, without replacement)
# the selection probabilities are computed using the variable 'income'
s <- Strata(m,c("region","state"), size=c(10,5,10,4,6),
            method="systematic", pik=m$income)

# extracts the observed data
data.frame(income=m[s$id, "income"], s)

# see the result using a contigency table
table(s$region, s$state)
```

`StrCap` *Capitalize the First Letter of a String*

Description

Capitalize the first letter of each element of the string vector.

Usage

```
StrCap(x)
```

Arguments

`x`	String to be capitalized.

Value

Returns a vector of charaters with the first letter capitalized

Author(s)

Charles Dupont <charles.dupont@vanderbilt.edu>

Examples

```
StrCap(c("Hello", "bob", "daN"))
```

StrChop *Split a String in a Number of Pieces With Fixed Length*

Description

Split a string in a number of pieces with fixed length

Usage

```
StrChop(x, len)
```

Arguments

x	the string to be cut in pieces.
len	a vector with the lengths of the pieces.

Details

If length is going over the end of the string the last part will be returned.

Value

a vector with the parts of the string.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

FixToTab

Examples

```
x <- paste(letters, collapse="")
StrChop(x=x, len = c(3,5,2))
```

StrCountW *Count Words in a String*

Description

Count the number of words that appear within a character string.

Usage

```
StrCountW(x)
```

Arguments

x	a vector of strings to be parsed.

Details

This is just a wrapper for a fine regexpr. It uses the expression \b\W+\b to separate the words. The code \W is equivalent to [^[:alnum:]_]) wheras [:alnum:] contains [:alpha:] and [:digit:]. So everything that is not an alphanumeric character, a digit or a _ (underscore) is used as separator for the words to be counted.

Value

an integer defining the number of word in the string

Author(s)

Andri Signorell <andri@signorell.net>, based on code from Adam Bradley <hisself@adambradley.net>

References

http://stackoverflow.com/questions/8920145/count-the-number-of-words-in-a-string-in-r

See Also

nchar

Examples

```
StrCountW("This is a true story!")

StrCountW("Just_one_word")
StrCountW("Not-just.one/word")

StrCountW("And what about numbers 8899 or special characters $$$/*?")
StrCountW("   Starting'n ending with some whitespace  ")

StrCountW(c("This is a", "text in more", "than one line."))
```

StrDist *Compute Distances Between Strings*

Description

StrDist computes distances between strings following to Levenshtein or Hamming method.

Usage

```
StrDist(x, y, method = "levenshtein", mismatch = 1, gap = 1)
```

Arguments

x	character vector, first string.
y	character vector, second string.
method	character, name of the distance method. This must be "levenshtein" or "hamming". Default is the classical Levenshtein distance.
mismatch	numeric, distance value for a mismatch between symbols.
gap	numeric, distance value for inserting a gap.

Details

The function computes the Hamming and the Levenshtein (edit) distance of two given strings (sequences). The Hamming distance between two vectors is the number mismatches between corresponding entries.

In case of the Hamming distance the two strings must have the same length.

In case of the Levenshtein (edit) distance a scoring and a trace-back matrix are computed and are saved as attributes "ScoringMatrix" and "TraceBackMatrix". The numbers in the trace-back matrix reflect insertion of a gap in string y (1), match/missmatch (2), and insertion of a gap in string x (3).

Value

StrDist returns an object of class "dist"; cf. dist.

Note

For distances between strings and for string alignments see also Bioconductor package **Biostrings**

Author(s)

Matthias Kohl <Matthias.Kohl@stamats.de>

References

R. Merkl and S. Waack (2009) *Bioinformatik Interaktiv*. Wiley.

See Also

adist, dist

Examples

```
x <- "GACGGATTATG"
y <- "GATCGGAATAG"
## Levenshtein distance
d <- StrDist(x, y)
d
attr(d, "ScoringMatrix")
attr(d, "TraceBackMatrix")

## Hamming distance
StrDist(x, y, method="hamming")
```

StrIsNumeric *Does a String Contain Only Numeric Data*

Description

Check whether a string does only contain numeric data.

Usage

```
StrIsNumeric(x)
```

Arguments

x	a character vector

Value

a logical vector with the same dimension as x

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Other string functions, e.g. StrTrunc

Examples

```
x <- c("123", "-3.141", "foobar123")
StrIsNumeric(x)
```

StrPad *StrPad a String With Justification*

Description

StrPad will fill a string x with some characters (str) to fit a given length.

Usage

```
StrPad(x, len, str = " ", adj = c("left", "right", "center"))
```

Arguments

x	string to be padded.
len	resulting length of padded string.
str	string to pad with. Will be repeated as often as necessary. Default is " ".
adj	adjustement of the old string, one of "left", "right", "center". If set to "left" the old string will be adjusted on the left and the new characters will be filled in on the right side.

Details

If a string x has more characters than len, it will be chopped on the length of len.

Value

the string

Author(s)

Christian W. Hoffmann <c-w.hoffmann@sunrise.ch>
some minor adjustments Andri Signorell <andri@signorell.net>

Examples

```
StrPad("My string", 25, "XoX", "center")
 # [1] "XoXXoXXoMy stringXXoXXoXX"
```

StrRev *Reverse a String*

Description

Returns a string in reverse order.

Usage

```
StrRev(x)
```

Arguments

x	a string to be processed.

Value

string

Author(s)

Andri Signorell <andri@signorell.net>

See Also

String functions: nchar, match, grep, regexpr, substr, sub, gsub, StrTrunc, StrDist

Examples

```
StrRev("home")
StrRev("Anna")
```

StrTrim *Trim a string*

Description

The function removes all spaces, tabs and newlines from the beginning and end of the supplied string. If these whitespace characters occur in the middle of the string, they are preserved.
Trim with method "left" deletes only leading whitespaces, "right" only trailing. Designed for users who were socialized by SQL...

Usage

```
StrTrim(x, pattern = " \t\n", method = "both")
```

Arguments

x	the string to be trimmed.
pattern	the pattern of the whitespaces to be deleted, defaults to space, tab and newline: " \t\n".
method	one out of "both", "left", "right". Determines on which side the string should be trimmed. Default is "both".

Details

The functions are defined depending on method as
both: gsub(pattern=gettextf("^[%s]+|[%s]+$", pattern, pattern), replacement="", x=x)
left: gsub(pattern=gettextf("^[%s]+",pattern), replacement="", x=x)
right: gsub(pattern=gettextf("[%s]+$",pattern), replacement="", x=x)

Value

the string x without whitespaces

Author(s)

Andri Signorell <andri@signorell.net>

See Also

String functions: nchar, match, grep, regexpr, substr, sub, gsub, StrTrunc, StrDist

Examples

```
StrTrim("  Hello world! ")

StrTrim("  Hello world! ", method="left")
StrTrim("  Hello world! ", method="right")

# user defined pattern
StrTrim(" ..Hello ... world! ", pattern=" \\.")
```

StrTrunc *Truncate Strings and Add Ellipses If a String is Truncated.*

Description

Truncates one or more strings to a specified length, adding an ellipsis (...) to those strings that have been truncated. Use formatC to justify the strings if needed.

Usage

```
StrTrunc(x, maxlen = 20)
```

Arguments

x	a vector of strings.
maxlen	the maximum length of the returned strings.

Value

The string(s) passed as 'x' now with a maximum length of 'maxlen' + 3 (for the ellipsis).

Author(s)

Andri Signorell, based on code of Jim Lemon

See Also

String functions: nchar, match, grep, regexpr, substr, sub, gsub, StrTrim, StrDist

truncString() in the package **prettyR**

Examples

```
set.seed(1789)
x <- sapply(seq(10),  function(x) paste(sample(letters, sample(20,1)),collapse=""))
x

StrTrunc(x, maxlen=10)

# right justification
formatC(StrTrunc(x, maxlen=10), width = 10, flag=" ")
```

StrVal *Extract All Numbers From a String*

Description

Extract all numbers from a string, using a regular expression.

Usage

```
StrVal(x)
```

Arguments

x	a character vector

Value

depending on the results the function will return either a character vector, in the case every element of x contained only one number, or a list of character vectors containing the found numbers.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

other string functions, e.g. StrTrunc

Examples

```
# a simple vector with only one number per element
StrVal(x=c("week 1", "week 3", "week 4", "week 5"))

# several numbers per element
StrVal(x=c("This is 1. place: 45.2", "none", "12.1 but -2.7 follow, 10.2e23 "))
```

StuartTauC *Stuart Tau C*

Description

Calculate Stuart tau-c statistic, a measure of association for ordinal factors in a two-way table. The function has interfaces for a table, a matrix, a data.frame and for single vectors.

Usage

```
StuartTauC(x, y = NULL, conf.level = NA, ...)
```

Arguments

x	a numeric vector, matrix or data.frame.
y	NULL (default) or a vector with compatible dimensions to x. If y is provided, table(x, y, ...) is calculated.
conf.level	confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.
...	further arguments are passed to the function table, allowing i.e. to set useNA. This refers only to the vector interface.

Details

Stuart's tau-c makes an adjustment for table size in addition to a correction for ties. Tau-c is appropriate only when both variables lie on an ordinal scale.
The range of tau-c is [-1, 1]. Stuart's tau-c is estimated by
tau-c = m (P-Q) / (n^2 (m-1)),
where P equals twice the number of concordances and Q twice the number of discordances, n is the total amount of observations and m = min(R,C).
See http://support.sas.com/documentation/cdl/en/statugfreq/63124/PDF/default/statugfreq.pdf, pp. 1739 for the estimation of the asymptotic variance.

Value

a single numeric value if no confidence intervals are requested,
and otherwise a numeric vector with 3 elements for the estimate, the lower and the upper confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

References

Agresti, A. (2002) *Categorical Data Analysis*. John Wiley & Sons, pp. 57–59.

Goodman, L. A., & Kruskal, W. H. (1954) Measures of association for cross classifications. *Journal of the American Statistical Association*, 49, 732-764.

Goodman, L. A., & Kruskal, W. H. (1963) Measures of association for cross classifications III: Approximate sampling theory. *Journal of the American Statistical Association*, 58, 310-364.

http://support.sas.com/onlinedoc/913/getDoc/en/statug.hlp/freq_sect18.htm
http://support.sas.com/onlinedoc/913/getDoc/en/statug.hlp/freq_sect20.htm

See Also

ConDisPairs yields concordant and discordant pairs
other association measures:
GoodmanKruskalTauA (tau-a), cor (method="kendall") for tau-b, GoodmanKruskalGamma, SomersDelta
Lambda, UncertCoef, MutInf

Examples

```
# example in:
# http://support.sas.com/documentation/cdl/en/statugfreq/63124/PDF/default/statugfreq.pdf
# pp. S. 1821

tab <- as.table(rbind(c(26,26,23,18,9),c(6,7,9,14,23)))

StuartTauC(tab, conf.level=0.95)
```

TextContrastColor *Choose Textcolor Depending on Background Color*

Description

Text of a certain color when viewed against certain backgrounds can be hard to see. TextContrastColor returns either black or white depending on which has the better contrast.

Usage

```
TextContrastColor(col, method = c("glynn", "sonego"))
```

Arguments

col	vector of any of the three kind of R colors, i.e., either a color name (an element of colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see rgb), or an integer i meaning palette()[i]. Non-string values are coerced to integer.
method	defines the algorithm to be used. Can be one out of `"glynn"` or `"sonego"`. See details.

Details

A simple heuristic in defining a text color for a given background color, is to pick the one that is "farthest" away from "black" or "white". The way Glynn chooses to do this is to compute the color intensity, defined as the mean of the RGB triple, and pick "black" (intensity 0) for text color if the background intensity is greater than 127, or "white" (intensity 255) when the background intensity is less than or equal to 127. Sonego calculates `L <- c(0.2, 0.6, 0) %*% col2rgb(color)/255` and returns #000060 if L >= 0.2 and #FFFFA0 else.

Value

a vector containing the contrast color (either black or white)

Author(s)

Andri Signorell <andri@signorell.net> based on code of Earl F. Glynn, Stowers Institute for Medical Research, 2004

References

http://research.stowers-institute.org/efg/R/Color/Chart

(Reference for Sonego??)

Examples

```
# works fine for grays
PlotArea( y=matrix(rep(1, times=3, each=8), ncol=8), x=1:3,
  col=gray(1:8 / 8), ylab="", xlab="", axes=FALSE )
text( x=2, y=1:8-0.5, levels(d.pizza$driver),
  col=TextContrastColor(gray(1:8 / 8)))

# and not so fine, but still ok, for colors
par(mfrow=c(1,2))
PlotArea( y=matrix(rep(1, times=3, each=12), ncol=12), x=1:3,
  col=rainbow(12), ylab="", xlab="", axes=FALSE, main="method = Glynn" )
text( x=2, y=1:12-0.5, levels(d.pizza$driver),
  col=TextContrastColor(rainbow(12)))

PlotArea( y=matrix(rep(1, times=3, each=12), ncol=12), x=1:3,
  col=rainbow(12), ylab="", xlab="", axes=FALSE, main="method = Sonego" )
text( x=2, y=1:12-0.5, levels(d.pizza$driver),
  col=TextContrastColor(rainbow(12), method="sonego"))
```

TheilU *Theil's U index of inequality*

Description

Calculate Theil's U index of inequality.

Usage

```
TheilU(a, p, method = c(2, 1), na.rm = FALSE)
```

Arguments

a	a numeric vector with the actual values.
p	a numeric vector containing the predictions.
method	method defining the type of Theil's two U measures, see Details. Default is 2.
na.rm	logical, indicating whether NA values should be stripped before the computation proceeds. If set to TRUE complete.cases out of cbind(x,y) will be used. Defaults to FALSE.

Details

Theil proposed two error measures, but at different times and under the same symbol U, which has caused some confusion. U method=1 is taken from Theil (1958, pp. 31-42). a represents the actual observations and p the corresponding predictions. He left it open whether a and p should be used as absolute values or as observed and predicted changes. Theil (1966, chapter 2) proposed U method=2 as a measure of forecast quality, "where Ai and Pi stand for a pair of predicted and observed changes." As U_1 has some serious disadvantages (see Bliemel 1973) it is recommended to use U_2 (default).

Author(s)

Andri Signorell <andri@signorell.net>

References

Theil, H. (1958): *Economic Forecasts and Policy*. Amsterdam: North Holland.

Thiel, H. (1966): *Applied Economic Forecasting*. Chicago: Rand McNally.

Bliemel, F. (1973): Theil's Forecast Accuracy Coefficient: A Clarification, *Journal of Marketing Research* Vol. 10, No. 4 (Nov., 1973), pp. 444-446

See Also

Gini

Examples

```
TheilU(1:10, 2:11, method=1)
TheilU(1:10, 2:11, method=2)
```

TukeyBiweight *Calculate Tukey's Biweight Robust Mean*

Description

This calculates a robust average that is unaffected by outliers.

Usage

```
TukeyBiweight(x, const = 9, na.rm = FALSE)
```

Arguments

x	a numeric vector
const	a constant. *const* is preassigned a value of 9 according to the Cook reference below but other values are possible.
na.rm	logical, indicating whether NA values should be stripped before the computation proceeds. Defaults to FALSE.

Details

This is a one step computation that follows the Affy whitepaper below, see page 22. *const* determines the point at which outliers are given a weight of 0 and therefore do not contribute to the calculation of the mean. *const* = 9 sets values roughly +/-6 standard deviations to 0. *const* = 6 is also used in tree-ring chronology development. Cook and Kairiukstis (1990) have further details.

An exact summation algorithm (Shewchuk 1997) is used. When some assumptions about the rounding of floating point numbers and conservative compiler optimizations hold, summation error is completely avoided. Whether the assumptions hold depends on the platform, i.e. compiler and CPU.

Value

A numeric mean.

Author(s)

Mikko Korpela <mikko.korpela@aalto.fi>

References

Statistical Algorithms Description Document, 2002, Affymetrix.

Cook, E. R. and Kairiukstis, L. A. (1990) *Methods of Dendrochronology: Applications in the Environmental Sciences*. Springer. ISBN-13: 978-0792305866.

Mosteller, F. and Tukey, J. W. (1977) *Data Analysis and Regression: a second course in statistics*. Addison-Wesley. ISBN-13: 978-0201048544.

Shewchuk, J. R. (1997) Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete and Computational Geometry*, 18(3):305-363. Springer.

See Also

HuberM, RobRange, RobScale

Examples

```
TukeyBiweight(rnorm(100))
```

`UncertCoef` *Uncertainty Coefficient*

Description

The uncertainty coefficient U(C|R) measures the proportion of uncertainty (entropy) in the column variable Y that is explained by the row variable X. The function has interfaces for a table, a matrix, a data.frame and for single vectors.

Usage

```
UncertCoef(x, y = NULL, direction = c("symmetric", "row", "column"),
           conf.level = NA, p.zero.correction = 1/sum(x)^2, ...)
```

Arguments

`x`	a numeric vector, a factor, matrix or data frame.
`y`	`NULL` (default) or a vector, an ordered factor, matrix or data frame with compatible dimensions to x.
`direction`	direction of the calculation. Can be `"row"` (default) or `"column"`, where `"row"` calculates UncertCoef (R|C) ("column dependent").
`conf.level`	confidence level of the interval. If set to NA (which is the default) no confidence interval will be calculated.
`p.zero.correction`	slightly nudge zero values so that their logarithm can be calculated
`...`	further arguments are passed to the function `table`, allowing i.e. to set useNA. This refers only to the vector interface.

Details

The uncertainty coefficient is computed as U(C|R) = (H(X) + H(Y) - H(XY)) / H(Y) and ranges from [0, 1].

Value

Either a single numeric value, if no confidence interval is required,
or a vector with 3 elements for estimate, lower and upper confidence intervall.

Author(s)

Andri Signorell <andri@signorell.net> strongly based on code from Antti Arppe <antti.arppe@helsinki.fi>

References

Theil, H. (1972), *Statistical Decomposition Analysis*, Amsterdam: North-Holland Publishing Company.

See Also

Entropy, Lambda, Assocs

Examples

```
# example from Goodman Kruskal (1954)

m <- as.table(cbind(c(1768,946,115), c(807,1387,438), c(189,746,288), c(47,53,16)))
dimnames(m) <- list(paste("A", 1:3), paste("B", 1:4))
m

# direction default is "symmetric"
UncertCoef(m)
UncertCoef(m, conf.level=0.95)

UncertCoef(m, direction="row")
UncertCoef(m, direction="column")
```

Untable *Untable*

Description

Recreates the data.frame out of a contingency table x.

Usage

```
Untable(x, dimnames = NULL, type = NULL, rownames = NULL, colnames = NULL)
```

Arguments

x	a numeric vector, a matrix or a table interpreted as frequencies which are to be inflated to the original list
dimnames	the dimension names of x to be used for expanding. Can be used to expand a weight vector to its original values. If set to NULL (default) the dimnames of x will be used.
type	defines the data type generated. This allows to directly define factors or ordered factors, but also numeric values. See examples.
rownames	A names vector for the rownames of the resulting data.frame. If set to NULL (default) the names will be defined according to the table's dimnames.
colnames	A names vector for the colnames of the resulting data.frame. If set to NULL (default) the names will be defined according to the table's dimnames.

Details

For x being a vector this reduces to `rep(..., n)` with n as vector (which is not supported by rep).

Value

a data.frame with the detailed data (even if x was a 1-dimensional table)

Author(s)

Andri Signorell <andri@signorell.net>

See Also

expand.grid, rep, gl, xtabs

Examples

```
d.titanic <- Untable(Titanic)
str(d.titanic)

# ... not the same as:
data.frame(Titanic)


tab <- table(set1=sample(letters[1:5], size=40, replace=TRUE),
             set2=sample(letters[11:15], size=40, replace=TRUE))
Untable(tab)


# return a numeric vector by setting type and coerce to a vector by [,]
Untable(c(6,2,2), dimnames=list(1:3), type="as.numeric")[,]


# how to produce the original list based on frequencies, given as a data.frame
d.freq <- data.frame(xtabs(Freq ~ Sex + Survived, data=Titanic))

# a data list with each individual
d.data <- Untable( xtabs(c(1364, 126, 367, 344) ~ .,
             expand.grid(levels(d.freq$Sex),levels(d.freq$Survived))))
head(d.data)

# expand a weights vector
Untable(c(1,4,5), dimnames=list(c("Zurich","Berlin","London")))

# and the same with a numeric vector
Untable(c(1,4,5), dimnames=list(c(5,10,15)), type="as.numeric")[,]
# ... which again is nothing else than
rep(times=c(1,4,5), x=c(5,10,15))
```

VarCI *Confidence Interval for the Variance*

Description

Calculates the confidence interval for the variance either the classical way or with the bootstrap approach.

Usage

```
VarCI(x, type = c("classic", "norm", "basic", "stud", "perc", "bca"),
      conf.level = 0.95, na.rm = FALSE, R = 999)
```

Arguments

x	a (non-empty) numeric vector of data values.
type	A vector of character strings representing the type of intervals required. The value should be any subset of the values "norm", "basic", "stud", "perc", "bca". See boot.ci.
conf.level	confidence level of the interval.
na.rm	logical. Should missing values be removed? Defaults to FALSE.
R	The number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case R would be a vector of integers where each component gives the number of resamples from each of the rows of weights. See boot.

Value

a numeric vector with 3 elements:

var	variance
lwr.ci	lower bound of the confidence interval
upr.ci	upper bound of the confidence interval

Author(s)

Andri Signorell <andri@signorell.net>

References

http://wiki.stat.ucla.edu/socr/index.php/AP_Statistics_Curriculum_2007_Estim_Var

See Also

MeanCI, MedianCI

Examples

```
VarCI(d.pizza$price, na.rm=TRUE)
VarCI(d.pizza$price, conf.level=0.99, na.rm=TRUE)

round(VarCI(d.pizza[,1:4], na.rm=TRUE), 3)


x <- c(14.816,14.863,14.814,14.998,14.965,14.824,14.884,14.838,14.916,
       15.021,14.874,14.856,14.860,14.772,14.980,14.919)
VarCI(x, conf.level=0.9)

# and for the standard deviation
```

```
sqrt(VarCI(x, conf.level=0.9))

# some bootstrap intervals
VarCI(x, type="norm")
VarCI(x, type="perc")
VarCI(x, type="bca")
```

VecRot *Vector Rotation*

Description

Shift the elements of a vector in circular mode to the right or to the left by n elements, such that the nth element is the first one of the new vector and the first n-1 elements are appended to the end.

Usage

```
VecRot(x, n)
```

Arguments

x	a vector of any type.
n	the number of elements to shift.

Details

The function will repeat the vector two times and select the appropriate number of elements from the required shift on.

Value

the shifted vector in the same dimensions as x.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

[, rep

Examples

```
VecRot(c(1,1,0,0,3,4,8), 3)

VecRot(letters[1:10], 3)
```

wdConst *Word VBA constants*

Description

This is a list with all VBA constants for MS Word 2010, which is useful for writing R functions based on recorded macros in Word. This way the constants need not be replaced by their numeric values and can only be complemented with the list's name, say the VBA-constant `wd10Percent` for example can be replaced by `wdConst$wd10Percent`.

Usage

```
data(wdConst)
```

Format

The format is:
List of 2755
$ wd100Words: num -4
$ wd10Percent: num -6
$ wd10Sentences: num -2
...

Source

Microsoft

WhichFlags *Get the Flags, Factors or Numerics of a data.frame*

Description

Returns a vector with the names of the columns with the the specific property.
WhichFlags returns the dichotomous columns, WhichFactors the factors, WhichNumerics all the numeric columns.

Usage

```
WhichFlags(d.frm)
WhichFactors(d.frm)
WhichNumerics(d.frm, type = c("all", "numeric", "integer"), excl.flags = FALSE)
WhichCharacters(d.frm)
```

Arguments

`d.frm`	data.frame whose elements should be listed.
`type`	type of numeric variables, defaults to `"all"`. This is only used by `WhichNumerics()`.
`excl.flags`	logical. Only used by WhichNumerics. Should flags be excluded or not. Default is FALSE, meaning flags will be included if they are numerics.

Details

`WhichFlags` returns all apparently dichotomous columns, like logicals, integers and numerics with exactly 2 unique values or factors with 2 levels.
`WhichFactors` returns all factors and ordered factors.
`WhichNumerics` returns all numeric columns. Whether integers should be included or not can be defined by `type`. Default is `"all"`.
`WhichCharacters` returns the names of all character vectors in a data.frame.

Value

a character vector with the names of the specific columns

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`class`, `mode`

Examples

```
data(d.pizza)

WhichFlags(d.pizza)
sapply(d.pizza[,WhichFlags(d.pizza)], class)

WhichFactors(d.pizza)
sapply(d.pizza[,WhichFactors(d.pizza)], class)

WhichNumerics(d.pizza)
WhichNumerics(d.pizza, type="numeric")
WhichNumerics(d.pizza, type="integer")
```

`Winsorize` *Winsorize*

Description

Clean data by means of winsorization, i.e., by shrinking outlying observations to the border of the main part of the data.

Usage

```
Winsorize(x, minval = quantile(x = x, probs = probs[1], na.rm = na.rm),
          maxval = quantile(x = x, probs = probs[2], na.rm = na.rm),
          probs = c(0.05, 0.95), na.rm = FALSE)
```

Arguments

x	a numeric vector to be cleaned.
minval	the low border, all values being lower than this will be replaced by this value. The default is set to the 5%-quantile of x.
maxval	the high border, all values being larger than this will be replaced by this value. The default is set to the 95%-quantile of x.
probs	numeric vector of probabilities with values in [0,1] as used in quantile.
na.rm	should NAs be omitted to calculate the quantiles? Note that NAs in x are preserved and left unchanged anyway.

Details

Consider standardizing (possibly robust) the data before winsorizing. See scale, RobScale

Value

A vector of the same length as the original data x containing the winsorized data.

Author(s)

Andri Signorell <andri@signorell.net>, based on code by Gabor Grothendieck <ggrothendieck@gmail.com>

See Also

Winsorize library(robustHD) contains an option to winsorize multivariate data

Examples

```
## generate data
set.seed(1234)     # for reproducibility
x <- rnorm(10)     # standard normal
x[1] <- x[1] * 10  # introduce outlier

## Winsorize data
x
Winsorize(x)
```

WoolfTest *Woolf Test*

Description

Test for homogeneity on $2 \times 2 \times k$ tables over strata (i.e., whether the log odds ratios are the same in all strata).

Usage

```
WoolfTest(x)
```

Arguments

x	a $2 \times 2 \times k$ table.

Value

A list of class `"htest"` containing the following components:

`statistic`	the chi-squared test statistic.
`parameter`	degrees of freedom of the approximate chi-squared distribution of the test statistic.
`p.value`	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.
`observed`	the observed counts.
`expected`	the expected counts under the null hypothesis.

Note

This function was previously published as `woolf_test()` in the **vcd** package and has been integrated here without logical changes.

Author(s)

David Meyer, Achim Zeileis, Kurt Hornik, Michael Friendly

References

Woolf, B. 1955: On estimating the relation between blood group and disease. *Ann. Human Genet.* (London) **19**, 251-253.

See Also

`mantelhaen.test`, `BreslowDayTest`

Examples

```
migraine <- xtabs(freq ~ .,
            cbind(expand.grid(treatment=c("active","placebo"),
                              response=c("better","same"),
                              gender=c("female","male")),
                  freq=c(16,5,11,20,12,7,16,19))
            )

WoolfTest(migraine)
```

WrdCaption *Insert Caption to Word*

Description

Insert a caption in a given level to a Word document. The caption is inserted at the current cursor position.

Usage

```
WrdCaption(x, stylename = wdConst$wdStyleHeading1, wrd = getOption("lastWord"))
```

Arguments

x	the text of the caption.
stylename	the name of the heading style in local language or the appropiate word constant.
wrd	the pointer to a word instance. Can be a new one, created by GetNewWrd() or an existing one, created by GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").

Author(s)

Andri Signorell <andri@signorell.net>

See Also

WrdText, WrdPlot, GetNewWrd, GetCurrWrd

Examples

```
## Not run:  # Windows-specific example

wrd <- GetNewWrd()
# insert a title in level 1
WrdCaption("My First Caption level 1", stylename=wdConst$wdStyleHeading1, wrd=wrd)

# works as well for several levels
sapply( 1:5, function(i)
  WrdCaption(gettextf("My First Caption level
             stylename=eval(parse(text=gettextf("wdConst$wdStyleHeading
             wrd=wrd)
)


## End(Not run)
```

WrdInsertBookmark *Insert a Bookmark, Goto Bookmark and Update the Text of a Bookmark*

Description

WrdInsertBookmark inserts a new bookmark in a Word document. WrdGotoBookmark allows to set the cursor on the bookmark and WrdUpdateBookmark sets the text within the range of the bookmark.

Usage

```
WrdInsertBookmark(name, wrd = getOption("lastWord"))
WrdGoto(name, what = wdConst$wdGoToBookmark, wrd = getOption("lastWord"))

WrdUpdateBookmark(name, text, what = wdConst$wdGoToBookmark, wrd = getOption("lastWord"))
```

Arguments

name	the name of the bookmark.
text	the text of the bookmark.
what	a word constant, defining the type of object to be used to place the cursor.
wrd	the pointer to a word instance. Can be a new one, created by GetNewWrd() or an existing one, created by GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").

Details

Bookmarks are useful to build structured documents, which can be updated later.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

WrdSetFont, WrdPlot, GetNewWrd, GetCurrWrd

Examples

```
## Not run:  # Windows-specific example

wrd <- GetNewWrd()
WrdText("a)\n\n\nb)", fontname=WrdGetFont()$name, fontsize=WrdGetFont()$size)
WrdInsertBookmark("chap_b")
WrdText("\n\n\nc)\n\n\n", fontname=WrdGetFont()$name, fontsize=WrdGetFont()$size)

WrdGoto("chap_b")
WrdUpdateBookmark("chap_b", "Goto chapter B and set text")


## End(Not run)
```

WrdInsTab *Insert a Table in a Word Document*

Description

Create a table with a specified number of rows and columns in a Word document at the current position of the cursor.

Usage

```
WrdInsTab(nrow = 1, ncol = 1, heights = NULL, widths = NULL, wrd = getOption("lastWord"))
```

Arguments

nrow	number of rows.
ncol	number of columns.
heights	a vector of the row heights (in [cm]). If set to NULL (which is the default) the Word defaults will be used. The values will be recyled, if necessary.
widths	a vector of the column widths (in [cm]). If set to NULL (which is the default) the Word defaults will be used. The values will be recyled, if necessary.
wrd	the pointer to a word instance. Can be a new one, created by GetNewWrd() or an existing one, created by GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").

Value

A pointer to the inserted table.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetNewWrd, WrdText

Examples

```
## Not run:  # Windows-specific example

wrd <- GetNewWrd()
WrdInsTab(nrow=3, ncol=3, wrd=wrd)


## End(Not run)
```

WrdPlot *Insert Active Plot to Word*

Description

This function inserts the plot on the active plot device to Word. The image is transferred by saving the picture to a file in R and inserting the file in Word. The format of the plot can be selected, as well as crop options and the size factor for inserting.

Usage

```
WrdPlot(type = "png", append.cr = TRUE, crop = c(0, 0, 0, 0),
        picscale = 100, height = NA, width = NA, res = 300,
        dfact = 1.6, wrd = getOption("lastWord"))
```

Arguments

type	the format for the picture file, default is "png".
append.cr	should a carriage return be appended? Default is TRUE.
crop	crop options for the picture, defined by a 4-elements-vector. The first element is the bottom side, the second the left and so on.
picscale	scale factor of the picture in percent, default ist 100.
height	height in cm, this overrides the picscale if both are given.
width	width in cm, this overrides the picscale if both are given.
res	resolution for the png file, defaults to 300.
dfact	the size factor for the graphic.
wrd	the pointer to a word instance. Can be a new one, created by GetNewWrd() or an existing one, created by GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").

Value

Returns a pointer to the inserted picture.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

WrdText, WrdCaption, GetNewWrd

Examples

```
## Not run:  # Windows-specific example

# let's have some graphics
plot(1,type="n", axes=FALSE, xlab="", ylab="", xlim=c(0,1), ylim=c(0,1))
rect(0,0,1,1,col="black")
```

```
segments(x0=0.5, y0=seq(0.632,0.67, length.out=100),
  y1=seq(0.5,0.6, length.out=100), x1=1, col=rev(rainbow(100)))
polygon(x=c(0.35,0.65,0.5), y=c(0.5,0.5,0.75), border="white",
  col="black", lwd=2)
segments(x0=0,y0=0.52, x1=0.43, y1=0.64, col="white", lwd=2)
x1 <- seq(0.549,0.578, length.out=50)
segments(x0=0.43, y0=0.64, x1=x1, y1=-tan(pi/3)* x1 + tan(pi/3) * 0.93,
  col=rgb(1,1,1,0.35))


# get a handle to a new word instance
wrd <- GetNewWrd()
# insert plot with a specified height
WrdPlot(wrd=wrd, height=5)
WrdText("Remember?\n", fontname="Arial", fontsize=14, bold=TRUE, wrd=wrd)
# crop the picture
WrdPlot(wrd=wrd, height=5, crop=c(9,9,0,0))


wpic <- WrdPlot(wrd=wrd, height=5, crop=c(9,9,0,0))
wpic


## End(Not run)
```

`WrdR` *Insert a R Command and It's Output in a Word Document*

Description

Insert an R Command and It's Output in Word document. Helpful for documenting tasks.

Usage

```
WrdR(x, wrd = getOption("lastWord"))
```

Arguments

`x`	R command as text to be evaluated.
`wrd`	the pointer to a word instance. Can be a new one, created by `GetNewWrd()` or an existing one, created by `GetCurrWrd()`. Default is the last created pointer stored in `getOption("lastWord")`.

Details

The command text will be placed in a Word document and formatted in italics. The result will be written in bold fontface.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

WrdText

Examples

```
# Windows-specific example
## Not run:
wrd <- GetNewWrd()
WrdR("sapply(iris[,-5], mean)", wrd=wrd)

## End(Not run)
```

WrdSetFont *Set the Font in Word*

Description

WrdSetFont sets the font in Word for the text to be inserted. WrdGetFont returns the font at the current cursor position.

Usage

```
WrdSetFont(fontname = "Consolas", fontsize = 7, bold = FALSE, italic = FALSE,
           wrd = getOption("lastWord"))
WrdGetFont(wrd = getOption("lastWord"))
```

Arguments

fontname	the name of the font as defined by Windows.
fontsize	the size of the font in points.
bold, italic	does the expected.
wrd	the pointer to a word instance. Can be a new one, created by GetNewWrd() or an existing one, created by GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").

Value

a list of the attributes of the font in the current cursor position:

name	the fontname
size	the fontsize
bold	bold
italic	italic

Author(s)

Andri Signorell <andri@signorell.net>

See Also

WrdText, WrdPlot, GetNewWrd, GetCurrWrd

Examples

```
## Not run:  # Windows-specific example

# start word
wrd <- GetNewWrd()

for( i in seq(10, 24, 2)) {
  WrdText(gettextf("This is Arial size %s \n", i), appendCR=FALSE,
    fontname="Arial", fontsize=i)
}
for( i in seq(10, 24, 2)) {
  WrdText(gettextf("This is Times size %s \n", i), appendCR=FALSE,
    fontname="Times", fontsize=i)
}


## End(Not run)
```

WrdTable *Produces a Table in Word*

Description

Creates a table in MS-Word.

Usage

```
WrdTable(tab, wrd = getOption("lastWord"), row.names = FALSE)
```

Arguments

tab	the table to be transferred to Word.
wrd	the pointer to a word instance. Can be a new one, created by GetNewWrd() or an existing one, created by GetCurrWrd(). Default is the last created pointer stored in getOption("lastWord").
row.names	

Details

A tricky problem, still unsolved... This is experimental code.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetNewWrd

Examples

```
# No example
```

`WrdText` *Insert Normal Text to Word*

Description

Write text in defined font and append a carriage return if requested.

Usage

```
WrdText(txt, fontname = "Consolas", fontsize = 7, bold = FALSE, italic = FALSE,
        appendCR = TRUE, wrd = getOption("lastWord"))
```

Arguments

`txt`	the text to be inserted.
`fontname`	the font of the text.
`fontsize`	the fontsize of the text.
`bold`	should the text be bold?
`italic`	should the text be italic?
`appendCR`	should a carriage return be appended to the text. Default is TRUE.
`wrd`	the pointer to a word instance. Can be a new one, created by `GetNewWrd()` or an existing one, created by `GetCurrWrd()`. Default is the last created pointer stored in `getOption("lastWord")`.

Value

Returns a list of the attributes of the font in the current cursor position.

`name`	the fontname
`size`	the fontsize
`bold`	bold
`italic`	italic

Author(s)

Andri Signorell <andri@signorell.net>

See Also

`WrdSetFont`, `WrdPlot`, `GetNewWrd`, `GetCurrWrd`

Examples

```
## Not run:  # Windows-specific example

# Let's write a story
data(d.diamonds)

# start word
wrd <- GetNewWrd()

WrdCaption("My Word-Story", stylename=wdConst$wdStyleHeading1)

WrdText("This will be the structure of d.diamonds:\n\n", appendCR=FALSE,
        fontname="Arial", fontsize=10)
WrdText(capture.output(str(d.diamonds)))

wrd[["Selection"]]$InsertBreak(wdConst$wdPageBreak)
WrdText("Lets insert a table (and this ist written Times)!", fontname="Times",
        fontsize=12, appendCR=FALSE, bold=T)

# insert table
wrd[["ActiveDocument"]][["Tables"]]$Add( wrd[["Selection"]][["Range"]],
                                         NumRows=2, NumColumns=2 )
WrdText("First Cell", fontname="Arial", fontsize=10)
wrd[["Selection"]]$MoveRight( Unit=wdConst$wdCell, Count=1 )
WrdText("Second Cell")
wrd[["Selection"]]$MoveRight( Unit=wdConst$wdCell, Count=1 )

wrd[["Selection"]]$MoveRight( Unit=wdConst$wdCharacter, Count=2,
                              Extend=wdConst$wdExtend )
wrd[["Selection"]][["Cells"]]$Merge()
WrdText("This cell was merged....", fontname="Arial", fontsize=10)


# exit the table range
wrd[["Selection"]]$EndOf( wdConst$wdTable )

wrd[["Selection"]]$MoveRight( wdConst$wdCharacter, 2, 0 )
wrd[["Selection"]]$TypeParagraph()

# let's insert a bookmark
wrd[["ActiveDocument"]][["Bookmarks"]]$Add("myBookmark")
wrd[["Selection"]]$MoveRight( Unit=wdConst$wdCharacter, Count=1 )
WrdText("\n\n", fontname="Arial", fontsize=10)

# set border for a paragraph
BorderBottom <- wrd[["Selection"]][["ParagraphFormat"]][["Borders"]]$Item(-3)
BorderBottom[["LineStyle"]] <- 1
wrd[["Selection"]]$MoveRight( wdConst$wdCharacter, 1, 0 )
WrdText("This paragraph has a Border", fontname="Arial", fontsize=10)
wrd[["Selection"]]$MoveRight( wdConst$wdCharacter, 2, 0 )
WrdText("\n\n", fontname="Arial", fontsize=10)

# insert new landscape section
wrd[["Selection"]]$InsertBreak(wdConst$wdSectionBreakNextPage)
wrd[["Selection"]][["PageSetup"]][["Orientation"]] <- wdConst$wdOrientLandscape
```

```
# new text
WrdText("Text in landscape", fontname="Impact", fontsize=20)
wrd[["Selection"]][["PageSetup"]][["Bottommargin"]] <- 4 * 72
wrd[["Selection"]][["PageSetup"]][["Leftmargin"]] <- 4 * 72
# wrd[["Selection"]][["PageSetup"]][["Topmargin"]] <- 4 * 72
# wrd[["Selection"]][["PageSetup"]][["Rightmargin"]] <- 4 * 72

# return to the bookmark
wrd[["Selection"]]$GoTo( wdConst$wdGoToBookmark, 0, 0, "myBookmark")

# and insert text
WrdText("Back again")

# goto end of document
wrd[["Selection"]]$EndKey(wdConst$wdStory)


## End(Not run)
```

`XLGetRange` *Get the Values of Cell Range(s) in Excel*

Description

Uses the package `RDCOMClient` to open an Excel workbook and return the content of one (or several) given range(s) in a specified sheet. Helpful, if pathologically scattered data on an Excel sheet, which can't simply be saved as CSV-file, has to be imported in R.

Usage

```
XLGetRange(file = NULL, sheet = NULL, range = NULL, as.data.frame = TRUE,
           header = FALSE, stringsAsFactors = FALSE)
```

Arguments

`file`	the fully specified path and filename of the workbook. If it is left as NULL, the function will look for a running Excel-Application and use its current sheet. The parameter sheet will then be ignored.
`sheet`	the name of the sheet containing the range(s) of interest.
`range`	a scalar or a vector of the range(s) to be returned (characters). Use "A1"-address mode to specify the ranges, for example `"A1:F10"`. If set to NULL (which is the default), the function will look for a selection containing more than one cell. If found, the function will use this selection. If there is no selection then the current region of the current selected cell will be used.
`as.data.frame`	logical. Determines if the cellranges should be turned into data.frames. Defaults to TRUE, as this might be the common use of this function.
`header`	a logical value indicating whether the range contains the names of the variables as its first line. Default is FALSE. `header` is ignored if `as.data.frame` has been set to FALSE.

stringsAsFactors
: logical. Should character columns be coerced to factors? Default is FALSE, which will return character vectors.

Details

The result consists of a list of lists, if as.data.frame is set to FALSE. Be then prepared to encounter NULL values. Those will prevent from easily being able to coerce the square data structure to a data.frame.

The following code will replace the NULL values by NAs and coerce to a data.frame.

```
# get the range D1:J69 from an excel file
xlrng <- XLGetRange(file="myfile.xlsx", sheet="Tabelle1",
                    range="D1:J69", as.data.frame=FALSE)

# replace NULL values by NA
xlrng[unlist(lapply(xlrng,is.null))] <- NA

# coerce the square data structure to a data.frame
d.lka <- data.frame(lapply(data.frame(xlrng), unlist))
```

This of course can be avoided by setting as.data.frame to TRUE.

Value

If as.data.frame is set to TRUE, a single data.frame or a list of data.frames will be returned. If set to FALSE a list of the cell content in the specified Excel range, resp. a list of lists will be returned.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetNewXL, XLGetWorkbook

Examples

```
## Not run:  # Windows-specific example

XLGetRange(file="C:\My Documents\data.xls",
           sheet="Sheet1",
           range=c("A2:B5","M6:X23","C4:D40"))


# if the current region is to be read (inkl. a header), place the cursor in the interesting region
# and run:
d.set <- XLGetRange(range=NULL, header=TRUE)

## End(Not run)
```

XLGetWorkbook *Get the Values of All Sheets of an Excel Workbook*

Description

Get the values of all sheets of an Excel workbook.

Usage

```
XLGetWorkbook(file)
```

Arguments

file	the filename of the Excel workbook to be loaded

Details

See details in XLGetRange.

Value

a list of lists of the values in the give workbook.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetNewXL, XLGetRange

XLView *Use Excel as Viewer for a Data.Frame*

Description

XLView can be used to view and edit a data.frame directly in Excel, resp. to create a new data.frame in Excel.

Usage

```
XLView(x, col.names = TRUE, row.names = TRUE)
```

Arguments

x	is a data.frame to be transferred to Excel. If data is missing a new file will be created.
row.names	either a logical value indicating whether the row names of x are to be written along with x, or a character vector of row names to be written.
col.names	either a logical value indicating whether the column names of x are to be written along with x, or a character vector of column names to be written. See the section on 'CSV files' write.table for the meaning of col.names = NA.

Details

The data.frame will be exported in CSV format and then imported in Excel.
Take care: Changes to the data made in Excel will NOT automatically be updated in the original data.frame. The user will have to read the csv-file into R again. See examples how to get this done.

Value

the name/path of the temporary file edited in Excel.

Note

The function obviously works only in Windows and requires **RDCOMClient** to be installed.
RDCOMClient is available here: http://www.omegahat.org

Author(s)

Andri Signorell <andri@signorell.net>

See Also

GetNewXL, XLGetRange, XLGetWorkbook

Examples

```
## Not run: # Windows-specific example
XLView(d.diamonds)

# edit an existing data.frame in Excel, make changes and save there, return the filename
fn <- XLView(d.diamonds)
# read the changed file and store in new data.frame
d.frm <- read.table(fn, header=TRUE, quote="", sep=";")

# Create a new file, edit it in Excel...
fn <- XLView()
# ... and read it into a data.frame when in R again
d.set <- read.table(fn, header=TRUE, quote="", sep=";")


## End(Not run)
```

Year *Part of Date*

Description

Returns the specific part of a given date x.

Usage

```
Year(x)
Quarter(x)
Month(x, format = c("num", "abbr", "full"), lang = c("local", "engl"),
      stringsAsFactors = TRUE)
Week(x)
Day(x)
Weekday(x, format = c("num", "abbr", "full"), lang = c("local", "engl"),
        stringsAsFactors = TRUE)
Yearday(x)
Yearmonth(x)

Day(x) <- value

IsWeekend(x)
```

Arguments

x	the date to be evaluated.
format	defines how the month or the weekday are to be formatted. Defaults to "num" and can be abbreviated. Is ignored for other functions.
value	new value
lang	the language for the months and daynames. This can be either the current locale (default) or english.
stringsAsFactors	logical. Defines if the result should be coerced to a factor, using the local definitions as levels. The result would be an ordered factor. Default is TRUE.

Details

These functions are only convenience wrappers for `format()` and its strange codes...
Based on the requested time component, the output is as follows:

`Year` returns the year of the input date in yyyy format.
`Quarter` returns the quarter of the year (1 to 4) for the input date.
`Month` returns the month of the year (1 to 12) for the input date.
`Week` returns the week of the year for the input date (0 to 53), as defined in ISO8601.
`Weekday` returns the week day of the input date. (1 - Monday, 2 - Tuesday, ... 7 - Sunday). (Names and abbreviations in the current locale!)
`Yearday` returns the day of the year numbering (1 to 366).
`Day` returns the day of the month (1 to 31).
`Yearmonth` returns the yearmonth representation (YYYYMM) of a date as longinteger.

`IsWeekend` returns TRUE, if the date x falls on a weekend.

The day can not only be extracted, but as well defined. See examples.

Value

a vector of the same dimension as x, consisting of either numeric values or characters depending on the function used.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

strptime, DateTimeClasses, as.POSIXlt

Examples

```
x <- Sys.Date()

Year(x)
Quarter(x)

Month(x)
Month(x, format = "abb", lang="engl")
Month(x, format = "abb", lang="local")
Month(x, format = "full", lang="engl")
Month(x, format = "full", lang="local")

Week(x)

Day(x)
Day(x) <- 20
x

Weekday(x)
Weekday(x, format = "abb", lang="engl")
Weekday(x, format = "abb", lang="local")
Weekday(x, format = "full", lang="engl")
Weekday(x, format = "full", lang="local")

Yearday(x)

IsWeekend(x)

# let's generate a time sequence by weeks
Month(seq(from=as.Date(Sys.Date()), to=Sys.Date()+150, by="weeks"), format="a")
```

Zodiac *Calculate the Zodiac of a Date*

Description

Calculate the sign of zodiac of a date.

Usage

```
Zodiac(x, lang = c("engl", "deu"), stringsAsFactors = TRUE)
```

Arguments

x	the date to be transformed.
lang	the language of the zodiac names, can be english (default) or german ("deu").
stringsAsFactors	logical. If set to TRUE (default) the result will consist of a factor with zodiac signs as levels.

Details

The really relevant things can sometimes hardly be found. You just discovered such a function... ;-)

Value

character vector or factor with the zodiac.

Author(s)

Andri Signorell <andri@signorell.net>, based on code from Markus Naepflin

See Also

Year and other date functions

Examples

```
Zodiac(c(Date(1937,7,28), Date(1936,6,1), Date(1966,2,25),
         Date(1964,11,17), Date(1972,4,25)), lang="deu")
```

ZTest *Z Test for Known Population Standard Deviation*

Description

Compute the test of hypothesis and compute confidence interval on the mean of a population when the standard deviation of the population is known.

Usage

```
ZTest(x, ...)

## Default S3 method:
ZTest(x, y = NULL, alternative = c("two.sided", "less", "greater"),
      paired = FALSE, mu = 0, sd_pop, conf.level = 0.95, ... )

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

Arguments

x	numeric vector of data values. Non-finite (e.g. infinite or missing) values will be omitted.
y	an optional numeric vector of data values: as with x non-finite values will be omitted.
mu	a number specifying the hypothesized mean of the population.
sd_pop	known standard deviation of the population.
alternative	is a character string, one of "greater", "less", or "two.sided", or the initial letter of each, indicating the specification of the alternative hypothesis. For one-sample tests, alternative refers to the true median of the parent population in relation to the hypothesized value of the mean.
paired	a logical indicating whether you want a paired z-test.
conf.level	confidence level for the interval computation.
formula	a formula of the form lhs ~ rhs where lhs gives the data values and rhs the corresponding groups.
data	an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula formula. By default the variables are taken from environment(formula).
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
...	further arguments to be passed to or from methods.

Details

Most introductory statistical texts introduce inference by using the z-test and z-based confidence intervals based on knowing the population standard deviation. Most statistical packages do not include functions to do z-tests since the t-test is usually more appropriate for real world situations. This function is meant to be used during that short period of learning when the student is learning about inference using z-procedures, but has not learned the t-based procedures yet. Once the student has learned about the t-distribution the t.test function should be used instead of this one (but the syntax is very similar, so this function should be an appropriate introductory step to learning t.test).

Value

An object of class htest containing the results

Note

This function should be used for learning only, real data should generally use t.test.

Author(s)

Andri Signorell <andri@signorell.net>, based on R-Core code of t.test,
documentation partly from Greg Snow <greg.snow@imail.org>

References

Stahel, W. (2002) *Statistische Datenanalyse, 4th ed*, vieweg

See Also

t.test, print.htest

Examples

```
x <- rnorm(25, 100, 5)
ZTest(x, mu=99, sd_pop=5)

# the classic interface
with(sleep, ZTest(extra[group == 1], extra[group == 2], sd_pop=2))

# the formula interface
ZTest(extra ~ group, data = sleep, sd_pop=2)


# Stahel (2002), pp. 186, 196

d.tyres <- data.frame(A=c(44.5,55,52.5,50.2,45.3,46.1,52.1,50.5,50.6,49.2),
                      B=c(44.9,54.8,55.6,55.2,55.6,47.7,53,49.1,52.3,50.7))
with(d.tyres, ZTest(A, B, sd_pop=3, paired=TRUE))


d.oxen <- data.frame(ext=c(2.7,2.7,1.1,3.0,1.9,3.0,3.8,3.8,0.3,1.9,1.9),
                     int=c(6.5,5.4,8.1,3.5,0.5,3.8,6.8,4.9,9.5,6.2,4.1))
with(d.oxen, ZTest(int, ext, sd_pop=1.8, paired=FALSE))
```

%like% *Like operator*

Description

The like operator is a simple wrapper for grepl, whose complexity is hard to crack for R-newbies.

Usage

```
x %like% pattern
```

Arguments

x	a vector, typically of character or factor type
pattern	simple character string to be matched in the given character vector.

Details

Follows the logic of simple SQL or basic commands.

Value

a vector (numeric, character, factor), matching the mode of x

Author(s)

Andri Signorell <andri@signorell.net>

See Also

match, pmatch, grep, %[]%, %overlaps%

Examples

```
# find names ending on "or"
names(d.pizza) %like% "%or"

# find names starting with "d"
names(d.pizza) %like% "d%"

# ... containing er?
names(d.pizza) %like% "%er%"

# the positions on the vector
which(names(d.pizza) %like% "%er%")

# what do they look like?
names(d.pizza)[names(d.pizza) %like% "%er%"]
```

%nin% *Find Matching (or Non-Matching) Elements*

Description

%nin% is a binary operator, which returns a logical vector indicating if there is a match or not for its left operand. A true vector element indicates no match in left operand, false indicates a match.

Usage

```
x %nin% table
```

Arguments

x	a vector (numeric, character, factor)
table	a vector (numeric, character, factor), matching the mode of x

Value

vector of logical values with length equal to length of x.

Author(s)

Frank E Harrell Jr <f.harrell@vanderbilt.edu>

See Also

match, %in%

Examples

```
c('a','b','c') %nin% c('a','b')
```

`%overlaps%` *Determines If And How Extensively Two Date Ranges Overlap*

Description

%overlaps% determines if two date ranges overlap at all and returns a logical value. Interval returns the number of days of the overlapping part of the two date periods. Inspired by the eponymous SQL-functions.

Usage

```
xp %overlaps% yp

Interval(xp, yp)
```

Arguments

xp	range 1, consisting of 2 numeric values, typically of type Date.
yp	range 2, consisting of 2 numeric values, typically of type Date.

Value

returns a logical vector (match or not for each element of x).
Interval returns an integer value.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

similar operators: Between, %like%
for calculating the overlapping time: difftime

Examples

```
c(Date(2012,1,3), Date(2012,2,3)) %overlaps% c(Date(2012,3,1), Date(2012,3,3))
c(Date(2012,1,3), Date(2012,2,3)) %overlaps% c(Date(2012,1,15), Date(2012,1,21))

Interval(c(Date(2012,1,3), Date(2012,2,3)), c(Date(2012,3,1), Date(2012,3,3)))


# both ranges are recyled if necessary
Date(2012,1,3) %overlaps% c(Date(2012,3,1), Date(2012,3,3))
```

```
# works with numerics as well
c(1, 18) %overlaps% c(10, 45)
```

`%c%` *Concatenates two strings without any separator.*

Description

%c% is just a short operator implementation for paste(x, y, separator="").

Usage

```
x %c% y
```

Arguments

x	first string
y	second string, which will be pasted behind the first one.

Details

Core does not consider it a good idea to use + as an operator not being commutative. So we use c here.
See the discussion: https://www.stat.math.ethz.ch/pipermail/r-devel/2006-August/039013.html
and
http://stackoverflow.com/questions/1319698/why-doesnt-operate-on-characters-in-r?lq=1

Still the paste syntax is clumsy in daily life and so %c% might spare some keys.

Value

returns the concatenation as string.

Author(s)

Andri Signorell <andri@signorell.net>

See Also

Between, %like%

Examples

```
"foo" %c% "bar"

# works with numerics as well
345 %c% 457
```

Index