Package 'fifer'

July 2, 2014

Title A collection of miscellaneous functions.

Description A collection of functions that assist in data manipulation analysis and plotting.

Version 1.0

Author Dustin Fife <fife.dustin@gmail.com>

Maintainer Dustin Fife <fife.dustin@gmail.com>

License GPL-2

Depends xtable, MASS

Collate 'polarCoord.R' 'stratified.R' 'univariate.tests.R'
'string.to.color.R' 'spearman.plot.R' 'r.crit.R' 'boxcoxR.R'
'missing.vals.R' 'make.formula.R' 'scaleB.R' 'par1.R'
'make.null.R' 'demographics.R' 'chisqPostHoc.R' 'cor2cov.R'
'random.correlation.R' 'scale.it.R' 'densityPlotR.R'
'excelMatch.R' 'prism.plots.R' 'hash.R' 'printx.R' 'roxtemp.R'
'intersperse.R' 'subsetString.R' 'auto.layout.R' 'contents.R' 'mvrnorm.R' 'kruskalmc.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2014-04-02 19:17:23

R topics documented:

auto.layout 2
boxcoxR 3
chisq.post.hoc 4
compute.theta 6
contents 6
cor2cov 7
demographics 8
densityPlotR 9
ellipse 10

excelCols . . . 10
excelMatch . . . 11
fakeMedicalData . . . 12
get.cols . . . 12
hash . . . 13
intersperse . . . 13
make.formula . . . 14
make.null . . . 15
missing.vals . . . 16
mv.rnorm . . . 16
par1 . . . 17
par2 . . . 18
plotSigBars . . . 18
printx . . . 19
prism.plots . . . 20
r . . . 21
r.crit . . . 22
random.correlation . . . 22
rotateGraph . . . 23
roxtemp . . . 24
scaleB . . . 25
scaleIt . . . 26
spearman.plot . . . 26
stratified . . . 27
string.to.color . . . 30
subsetString . . . 31
SummarizeContinuousDefault . . . 32
SummarizeFactorDefault . . . 33
SummarizeVar . . . 34
univariate.tests . . . 35

Index **36**

`auto.layout` *Automatically select the layout.*

Description

Given a particular number of plots, `auto.layout` will automatically determine the arrangement of each plot using the `layout` function. See examples.

Usage

```
auto.layout(n, layout = T)
```

Arguments

n	the number of plots
layout	should the fuction return a preallocated layout object? If FALSE, it returns a matrix

Value

either a matrix or a layout object

Author(s)

Dustin Fife

Examples

```
## plot six plots
auto.layout(6)
for (i in 1:6){
plot(rnorm(100), rnorm(100))
}
## same as mar(mfrow=c(3,2))
par(mfrow=c(3,2))
for (i in 1:6){
plot(rnorm(100), rnorm(100))
}
## default for odd number of plots using mfrow looks terrible
par(mfrow=c(3,2))
for (i in 1:5){
plot(rnorm(100), rnorm(100))
}
## much better with auto.layout
auto.layout(5)
for (i in 1:5){
plot(rnorm(100), rnorm(100))
}
```

---

boxcoxR *Transform data using a boxcox transformation*

Description

This function takes a variable as input, computes the optimal lambda using a boxcox transformation, then returnes a transformed version of the variable.

Usage

```
boxcoxR(x, minval = 0.01, ...)
```

Arguments

x	a numerical vector
minval	before a transformation is performed, the variables must often be positive. This tells R what the minimum value should be. Defaults to .01.
...	additional parameters to be used in the model fitting.

Details

The MASS package has a function that computes the optimal lambda value for a particular regression equation. However, it currently (as of version 7.3-23) returns a lambda vector rather than the boxcox transformed variable. This function is a wrapper for boxcox that actually returns a vector that is a transformed version of the original variable.

Note

This function calls the boxcox function in the MASS package. To avoid loading the package, I have branched the function directly into the fifer package.

Author(s)

Dustin Fife <fife.dustin@gmail.com>.

References

Venables, W. N. & Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth Edition. Springer, New York. ISBN 0-387-95457-0

Examples

```
x = rnorm(100)^2
### use original boxcox function
boxcox(x~1, plot=FALSE) ## returns a vector of lambda values and their likelihoods
### use boxcoxR function
boxcoxR(x)
```

---

chisq.post.hoc — *Tests for significant differences among all pairs of populations in a chi-square test.*

Description

Tests for significant differences among all pairs of populations in a chi-square test.

Usage

```
chisq.post.hoc(tbl, test = c("fisher.test"),
  popsInRows = TRUE,
  control = c("fdr", "BH", "BY", "bonferroni", "holm", "hochberg", "hommel"),
  digits = 4)
```

Arguments

tbl	A table object.
test	What sort of test will be used? Defaults to "fisher.test"
popsInRows	A logical indicating whether the populations form the rows (default; =TRUE) of the table or not (=FALSE).
control	A string indicating the method of control to use. See details.
digits	A numeric that controls the number of digits to print.
...	Other arguments sent to print.

Details

Post-hoc tests for which pairs of populations differ following a significant chi-square test can be constructed by performing all chi-square tests for all pairs of populations and then adjusting the resulting p-values for inflation due to multiple comparisons. The adjusted p-values can be computed with a wide variety of methods – fdr, BH, BY, bonferroni, holm, hochberg, and hommel. This function basically works as a wrapper function that sends the unadjusted "raw" p-values from each pair-wise chi-square test to the p.adjust function in the base R program. The p.adjust function should be consulted for further description of the methods used.

Value

A data.frame with a description of the pairwise comparisons, the raw p-values, and the adjusted p-values.

Note

This code was adapted and modified from the NCStats package (see http://www.rforge.net/doc/packages/NCStats/chisqPostHoc.html)

See Also

chisq.test and p.adjust.

Examples

```
# Makes a table of observations -- similar to first example in chisq.test
M <- as.table(rbind(c(76, 32, 46), c(48,23,47), c(45,34,78)))
dimnames(M) <- list(sex=c("Male","Female","Juv"),loc=c("Lower","Middle","Upper"))
M
# Shows post-hoc pairwise comparisons using fdr method
chisq.post.hoc(M)
```

compute.theta *Calculate inverse tangent.*

Description

Computes angle in polar coordinates, accounting for which quadrant the datapoints are in.

Usage

```
compute.theta(x)
```

Arguments

x	a vector of size two that give the cartesian coordinates

Value

returns the theta, expressed in radians.

Author(s)

Dustin Fife

contents *View the contents (functions) of a package*

Description

View the contents (functions) of a package

Usage

```
contents(x = NULL)
```

Arguments

x	a string containing the name of the package. Defaults to "fifer".

Details

It's easy to forget what functions are contained within a package. `contents` will return all the functions within a package

Value

a string of all the functions contained within that package.

Author(s)

Dustin Fife

Examples

```
## see what's in fifer
contents()
## see what's in MASS
## Not run: contents("MASS")
```

---

cor2cov *Correlation Matrix to Covariance Matrix Conversion*

Description

Function to convert a correlation matrix to a covariance matrix.

Usage

```
cor2cov(cor.mat, sd, discrepancy = 1e-05)
```

Arguments

cor.mat	the correlation matrix to be converted
sd	a vector that contains the standard deviations of the variables in the correlation matrix
discrepancy	a neighborhood of 1, such that numbers on the main diagonal of the correlation matrix will be considered as equal to 1 if they fall in this neighborhood

Details

The correlation matrix to convert can be either symmetric or triangular. The covariance matrix returned is always a symmetric matrix.

Note

The correlation matrix input should be a square matrix, and the length of sd should be equal to the number of variables in the correlation matrix (i.e., the number of rows/columns). Sometimes the correlation matrix input may not have exactly 1's on the main diagonal, due to, eg, rounding; discrepancy specifies the allowable discrepancy so that the function still considers the input as a correlation matrix and can proceed (but the function does not change the numbers on the main diagonal).

Author(s)

Ken Kelley (University of Notre Dame; (<KKelley@ND.Edu>) and Keke Lai (the MBESS package), with modifications by Dustin Fife <fife.dustin@gmail.com>.

demographics *Summarize a Data Set (Demographics)*

Description

Creates a summary table of a data set in a matrix object for pretty printing via xtable.

Usage

```
demographics(formula, data, latex = TRUE,
  na.action = na.pass, ...)
```

Arguments

formula	A formula, with the left-hand side being empty or a group variable to summarize by. The right-hand side should include variables to summarize by; they should be either continuous variables or factors/characters.
data	A data.frame where the variables in formula come from; if not specified, variables are looked for in the parent environment.
latex	A logical variable that determines whether the resulting output will be part of a LaTeX document. Defaults to TRUE.
na.action	A function to handle missing data. See na.pass.
...	Additional arguments passed to SummarizeVar, such as decimalFactor, decimalContinuous, ContinuousSummaryFunction, and FactorSummaryFunction.

Details

This is generally used to create demographics table and used with the package xtable to print. To get proper names to display, a data.frame should be constructed such that the variable names are what the users want to be displayed. For factor variables, the user should make use of the levels and labels arguments in factor.

Value

A matrix to be used with xtable, which in turn should be used in print.xtable.

Author(s)

Vinh Nguyen

Examples

```
set.seed(1)
n <- 50
df <- data.frame(trt=sample(0:1, 2*n, replace=TRUE), x1=runif(2*n), x2=rnorm(2*n),
   x3=sample(c("a", "b", "c"), 2*n, replace=TRUE))
demographics(~x1+x2+x3, data=df)
```

```
demographics(trt~x1+x2+x3, data=df)
demographics(~., data=df)
demographics(trt~., data=df, decimalFactor=2)
## Not run: print(xtable(Summarize(trt~., data=df)), sanitize.text.function=identity)
```

`densityPlotR` *Generate a density plot using a formula*

Description

Given two groups with scores on a quantitative variable, `densityPlotR` will draw both distributions, one for each group.

Usage

```
densityPlotR(formula, data = NULL, colors = NULL, ...)
```

Arguments

`formula`	a formula object where the grouping variable is on the right side of the ~ and the response variable (quantitative) is on the left side of the ~.
`data`	a dataset containing the variables listed in the formula
`colors`	a vector the same length as the number of levels of the grouping variable indicating the colors to be used for the density lines
`...`	other arguments passed to the plot function

Author(s)

Dustin Fife

See Also

`boxplot`, `prism.plots`, `plotSigBars`

Examples

```
densityPlotR(Petal.Width~Species, data=iris)
```

ellipse *Plots an Ellipse*

Description

##' Plots an ellipse

Usage

```
ellipse(x0, y0, axisX, axisY, color = "lightgray")
```

Arguments

x0	the x coordinate of the center of the circle
y0	the y coordinate of the center of the circle
axisX	the radius along the x axis
axisY	the radius along the y axis
color	what color should the ellipse be drawn in

Author(s)

Dustin Fife

excelCols *Generate Excel column labels*

Description

Generate a sequence of column labels to match Excel's naming

Usage

```
excelCols(n)
```

Arguments

n	an integer that indicates how many named columns the user wishes to obtain

Details

Excel columns are labeled with letters (e.g., A, B, C, ... AA, AB, AC, etc). Given an integer (n), this function will return labels starting from A until the nth column. See examples.

Value

a vector of strings of length n

Author(s)

Dustin Fife

See Also

excelMatch

Examples

```
excelCols(11)
```

---

excelMatch *Obtain column number or variable name from Excel named Columns*

Description

Obtain column number or variable name from Excel named Columns

Usage

```
excelMatch(..., n = NULL, names = NULL)
```

Arguments

...	An Excel-like string consisting of all capital letters (e.g., "AAQ", "BZ", "RQ", "S", etc.)
n	the number of columns of the data of interest
names	the column names of the data of interest

Details

Excel columns are labeled with letters (e.g., A, B, C, ... AA, AB, AC, etc). This function accepts a string (e.g., "AAC") and returns either a number that indicates where that string falls in the sequence of excel named columns, or it returns the variable name corresponding to that column number.

Value

either a variable name or column number

Author(s)

Dustin Fife

See Also

excelCols

Examples

```
fake.names = paste("Variable", 1:1000, sep="")
# find the Variable name corresponding to AC
excelMatch("AC", names=fake.names)
# find the column number instead
excelMatch("AC", n=1000)
```

`fakeMedicalData` *Fictitious Medical Dataset*

Description

Fictitious Medical Dataset

Author(s)

Dustin Fife <blahblah@roxygen.org>

`get.cols` *Extract column index*

Description

Given a vector of strings and another reference vector, `get.cols` will search the reference vector for matches and return the column indices where each string was located. See examples.

Usage

```
get.cols(string, names)
```

Arguments

`string`	the name of a variable (string)
`names`	the names of the variables for which you wish to extract the column

Value

an integer corresponding to the column index(indices)

Author(s)

Dustin Fife

See Also

`make.null`, `r`

Examples

```
names = LETTERS[1:10]
get.cols(c("A", "B", "D"), LETTERS[1:10])  ### works
#get.cols(c("A", "B", "Z"), LETTERS[1:10])  ### doesn't work
```

---

hash *Create useless hashes*

Description

A function to create useless hashes (####) for ease of commenting

Usage

```
hash(j = 50, i = 4)
```

Arguments

j	How many columns of hashes
i	How many rows of hashes

Author(s)

Dustin Fife

Examples

```
hash(j=100, i=11)
```

---

intersperse *Intersperse elements of Two+ Vectors*

Description

Intersperse elements of two+ vectors Given two vectors (one of length i, the other of length j), `intersperse` will combine the elements of each vector into strings of length i X j, where each element is the concatenation of the elements of the two vectors. See examples.

Usage

```
intersperse(...)
```

Arguments

...	the vectors the user wishes to intersperse

Value

a vector of length i X j, containing the interspersed vectors as strings

Author(s)

Dustin Fife

Examples

```
intersperse(LETTERS[1:3], 1:3)
```

`make.formula` *Convert strings to a formula*

Description

`make.formula` is a function that easily converts a set of strings into a formula. It requires two arguments: a single response variable, and a vector of strings. See examples.

Usage

```
make.formula(response, predictors)
```

Arguments

`response`	a single string used on the left side of a formula
`predictors`	a string (or a vector of strings) representing the predictors. Each will be separated by a plus sign.

Value

a formula object

Author(s)

Dustin Fife

Examples

```
k = data.frame(matrix(rnorm(100), ncol=5))
names(k) = LETTERS[1:5]
formula = make.formula("A", LETTERS[2:5])
formula
lm(formula, data=k)
```

make.null *Drop or keep variables in a dataset*

Description

Given a set of variable names (or integers), remove (or keep) these columns from a dataset

Usage

```
make.null(..., data, keep = FALSE)
```

Arguments

...	objects (either vectors of strings, vectors of numbers, or both)
data	the dataset you're trying to eliminate (or keep) variables from
keep	should the variables be kept (keep=T) or dropped (keep=F)?

Value

a dataset, containing the variables of interest only

Author(s)

Dustin Fife

See Also

subset.

get.cols, r

Examples

```
data = data.frame(matrix(rnorm(100), ncol=5))
names(data) = LETTERS[1:5]

#### extract only the classification
data(iris)
new.data = make.null(c("Sepal.Length"),
       r("Sepal.Width", "Petal.Width", names(iris)), data=iris)

#### extract all but the classification
new.data2 = make.null(c("Sepal.Length"),
   r("Sepal.Width", "Petal.Width", names(iris)), data=iris, keep=TRUE)
```

missing.vals *Summary of Missing Data*

Description

Given a dataset, `missing.vals` will report on the number of missing observations per variable

Usage

```
missing.vals(dataset)
```

Arguments

`dataset`	a matrix or data frame (persons in rows, variables in columns).

Value

a matrix with information about what is missing.

Author(s)

Dustin Fife

mv.rnorm *Randomly Generate Multivariate Normal Data*

Description

This function will randomly generate correlated multivariate normal data with specified means and covariances (or correlations). The user also has the flexibility to generate data with a randomly selected correlation matrix using the `random.correlation` function.

Usage

```
mv.rnorm(n = 1, vars = NULL, mu = NULL, Sigma = NULL,
  names = NULL)
```

Arguments

`n`	The sample size of the randomly generated dataset
`vars`	An integer indicating the number of variables. Ignored unless no `Sigma` is supplied.
`mu`	A vector of means that has the same length as the number of rows/columns of Sigma. Defaults to a vector of zeroes.
`Sigma`	A positive definate matrix. If `NULL`, the user must specify `vars`.
`names`	Optional. A vector of strings indicating the variable names.

Details

mv.norm generates correlated multivariate normal data using a choleski decomposition. If the user does not specify Sigma, a random correlation matrix will be generated. Also, if means are not specified, the function will default to means of zero.

Value

a nxp matrix of pseuodo-random values.

Author(s)

Dustin Fife

See Also

random.correlation cor2cov

Examples

```
set.seed(2)
## generate data with correlation of .6
d = mv.rnorm(n=1000, Sigma=matrix(c(1, .6, .6, 1), 2), names=c("x", "y"))
head(d); cor(d)
## generate data with a random correlation
d = mv.rnorm(n=1000, vars=4, names=letters[1:4])
head(d); cor(d)
## generate non-scaled data
ms = c(100, 10, 5, 0) ### specify means
Sigma = matrix(c(1, .6, .5, .4,
.6, 1, .3, .2,
.5, .3, 1, .1,
.4, .2, .1, 1), 4)
## convert Sigma to covariance matrix
Sigma = cor2cov(Sigma, sd=c(15, 3, 2, 1))
## generate the data
d = mv.rnorm(n=1000, mu=ms, Sigma=Sigma, names=letters[1:4])
head(d); cor(d)
```

---

par1 *Change default par parameters*

Description

Change default par parameters

Usage

```
par1()
```

Author(s)

Dustin Fife

---

par2 *Change default par parameters*

Description

Change default par parameters

Usage

```
par2()
```

Author(s)

Dustin Fife

---

plotSigBars *Add significance bars to a prism plot*

Description

Add significance bars to a prism plot, corrected for multiple comparisons either using Tukey's HSD (parametric), or Dunn's correction for multiple comparison (non-parametric).

Usage

```
plotSigBars(formula, data, type = c("tukey", "dunn"))
```

Arguments

formula	a R formula object
data	a dataset containing the variables in formula
type	either "tukey" or "dunn" indicating which multiple comparison should be used

Note

This function should probably only be used when the number of groups is less than four, otherwise the number of pairwise comparisons becomes too large to display.

When p-values are adjusted using Dunn's multiple comparison, this function calls the `kruskalmc` function in the `pgirmess` package. To avoid having to load the entire package, the function was directly copied into the fifer package. references Patrick Giraudoux (2013). pgirmess: Data analysis in ecology. R package version 1.5.7. http://CRAN.R-project.org/package=pgirmess

Author(s)

Dustin Fife

See Also

boxplot, densityPlotR, prism.plots

Examples

```
prism.plots(Sepal.Length ~ Species, data = iris, centerfunc=mean)
plotSigBars(Sepal.Length ~ Species, data = iris, type="tukey")
```

---

printx *Change Defaults of print.xtable*

Description

Change Defaults for print.xtable

Usage

```
printx(x, file = "", ...)
```

Arguments

x	The xtable object the user wishes to export
file	the location where the file is stored
...	other arguments passed to print.xtable

Details

This is a wrapper for the function print.xtable, where the defaults have been specified for apa-like tables.

Author(s)

Dustin Fife

`prism.plots` *Plot prism-like Plots*

Description

Plot prism-like Plots

Usage

```
prism.plots(formula, data, centerfunc = mean,
  spreadfunc = function(x) {
      return(sd(x)/sqrt(length(x))) }, def.axis = TRUE,
  jitter.y = FALSE, add = FALSE, start = 0, ...)
```

Arguments

`formula`	a formula object with the quantitative variable as the response variable (e.g., Var~group).
`data`	a dataset containing the variables indicated in `formula`
`centerfunc`	what function should be used to indicate the center of the distribution. Defaults to mean.
`spreadfunc`	what function should be used to calculate the spread of the distribution. Defaults to sd. Currently, it must be a symmetrical function. Future implementations will have non-symmetric functions (e.g., interquartile range).
`def.axis`	Logical. Should the default axes be used?
`jitter.y`	Logical. Should the y values be jittered as well?
`add`	Should the plot be added to an existing plot?
`start`	What X value should the plot start at? (defaults to zero)
`...`	other arguments passed to plot

Details

Given a factor (e.g., group membership) and a quantitative variable, this function plots a psuedo-scatterplot of the groups on the x axis (jittered) and the DV on the y axis.

Author(s)

Dustin Fife

See Also

`boxplot`, `densityPlotR`, `plotSigBars`

Examples

```
prism.plots(count ~ spray, data = InsectSprays, centerfunc=mean)
prism.plots(count ~ spray, data = InsectSprays, centerfunc=median)
```

r *Extract variable column indices*

Description

Given a starting string and an ending string, `r` will extract all columns between the first and the last string and return either a vector of strings or a vector of integers. See examples.

Usage

```
r(stringA, stringB, data.names, names = F)
```

Arguments

`stringA`	the name of the first variable you wish to extract the column for in sequence
`stringB`	the name of the last variable you wish to extract the column for in sequence
`data.names`	the names of the dataset
`names`	Should the names of the variables be returned? Or the column indices? Defaults to FALSE (meaning the column indices will be returned).

Value

a vector of numbers corresponding to the column names

Author(s)

Dustin Fife

See Also

`make.null`, `get.cols`

Examples

```
var.names = LETTERS[1:20]
r("C", "F", var.names)
```

r.crit *Compute critical r or p.*

Description

Compute the critical r value, or return the p value of an r, assuming a given number of degrees of freedom.

Usage

```
r.crit(n, r = NULL, p = 0.025)
```

Arguments

n	The sample size.
p	the probability. Defaults to .025.
r	The observed r value

Value

the critical r value or the observed p value for a given r

Author(s)

Dustin Fife

Examples

```
r.crit(n=100, p=.025)
r.crit(n=20, r=.6, p=.05)
```

random.correlation *Generate a random correlation matrix*

Description

Generate a random correlation matrix from specified eigenvalues. If eigenvalues are not specified, they are randomly generated from a uniform [0,10] distribution.

Usage

```
random.correlation(n, ev = runif(n, 0, 10))
```

Arguments

n	the number of rows/dimensions of the correlation matrix
ev	Eigenvalues. Defaults to sampling from a uniform distribution betwen 0 and 10.

Value

a correlation matrix, of size nxn

Author(s)

Dustin Fife

References

https://stat.ethz.ch/pipermail/r-help/2008-February/153708.html

---

rotateGraph *Rotate a graph using polar coordinates*

Description

Rotates the x-y coordinates by choosing one datapoint as the origin, choosing another to be fixed on the x axis, and choosing a third to be positive or negative.

Usage

```
rotateGraph(coords, scale = NULL, origin, axis,
  fixedPos = 2)
```

Arguments

coords	a nx2 dimensional matrix corresponding to the cartesian coordinates of the n datapoints.
scale	a nx2 dimensional matrix. This is used when comparing two graphs to one another. This parameter will scale one graph to the other by ensuring the average of the radi are the same across the two graphs. Defaults to NULL.
origin	an integer indicating which datapoint (i.e., which row) should be fixed as the origin.
axis	an integer indicating which datapoint (i.e., which row) should be fixed on the x-axis.
fixedPos	an integer indicating which datapoint (i.e., which row) must be positive on y.

Details

Many algorithms exist for projecting m-dimensional datapoints in two-dimensions (e.g., tsne and MDS). However, often they begin the algorithm by randomly placing datapoints in an arbitrary position. Unfortunately, this makes the axes meaningless from one iteration of the algorithm to the next, making comparisons across datasets (for example) impossible. One solution is to fix one datapoint to the origin, while rotating the others about the origin. This algorithm does just that by using polar coordinates.

Value

`coords`	the new coordinates obtained after rotation
`radi`	a vector of the radi for each of the datapoints

Author(s)

Dustin Fife

References

http://www.mathsisfun.com/polar-cartesian-coordinates.html

See Also

`compute.theta`.

`roxtemp` *Generate a Roxygen Template*

Description

Generate a Roxygen Template

Usage

```
roxtemp()
```

Author(s)

Dustin Fife

scaleB *Standardize coefficients*

Description

Generic function for obtaining scaled coefficients

Usage

```
scaleB(object, scale.response = F)
```

Arguments

object	an object resulting from glm, lm, or lda
scale.response	should the response variable be scaled as well? (Usually not for glm or lda).

Details

Given an object of class lm, glm, or lda, this function will first standardize the variables, then run the model again. The resulting coefficients will be standardized betas.

Value

an object of the same class as the one outputted

Author(s)

Dustin Fife

Examples

```
#### create random data with different means and variances
d = data.frame(matrix(rnorm(5*50, c(10,5,14,100, 33), c(3,5,4,3,5)), nrow=50, byrow=TRUE))
names(d) = LETTERS[1:5]
g = lm(C~B + A + D + E, data=d)
scaleB(g, TRUE)
##### make a logistic
d$A = as.factor(as.numeric(cut(d$A, 2, labels=c(1,0)) ))
object = glm(A~B + C + D + E, data=d, family=binomial)
scaleB(object)
##### LDA
object = lda(A~B + C + D + E, data=d, family=binomial)
scaleB(object)
```

scaleIt *Scale a variable*

Description

Scale a variable to have a particular mean/sd (or min/max)

Usage

```
scaleIt(x, mean = NULL, sd = NULL, min = NULL,
  max = NULL)
```

Arguments

x	the variable to be scaled
mean	the mean you wish the distribution to have
sd	the sd you wish the distribution to have
min	the min you wish the distribution to have
max	the max you wish the distribution to have

Value

the scaled variable

Author(s)

Dustin Fife

spearman.plot *Spearman plot*

Description

Plots the relationship between two variables using a Spearman Plot

Usage

```
spearman.plot(x, y = NULL, dcol = "blue", lhist = 20,
  num.dnorm = 5 * lhist, plot.cor = TRUE, ...)
```

Arguments

x	either a matrix with two columns or a vector (if y is not NULL)
y	a vector
dcol	the color of the lines drawn for the density plot
lhist	the number of breaks in the histogram
num.dnorm	the number of breaks in the density line
plot.cor	logical. Should the spearman correlation be outputted in the plot?
...	arguments passed to plot

Details

Often data are not normally distributed, requiring the use of a spearman correlation to determine their relationship. However, doing so makes it difficult to visualize the data since scatterplots of raw data present the data as if a pearson correlation were used. This function plots the ranks of the data, while plotting along the axes the distributions of the raw data.

Author(s)

Dustin Fife

Examples

```
### generate skewed data
x = rnorm(1000)^2
y = .6*x + rnorm(1000, 0, sqrt(1-.6^2))

spearman.plot(cbind(x,y), col="red", lhist=50)
spearman.plot(x=iris$Sepal.Length, y=iris$Sepal.Width)
```

---

stratified *Sample from a* data.frame *according to a stratification variable*

Description

The stratified function samples from a data.frame in which one of the columns can be used as a "stratification" or "grouping" variable. The result is a new data.frame with the specified number of samples from each group.

Usage

```
stratified(df, group, size, select = NULL, seed = NULL,
  ...)
```

Arguments

df	The source data.frame.
group	Your grouping variables. Generally, if you are using more than one variable to create your "strata", you should list them in the order of *slowest* varying to *quickest* varying. This can be a vector of names or column indexes.
size	The desired sample size. • If size is a value between 0 and 1 expressed as a decimal, size is set to be proportional to the number of observations per group. • If size is a single positive integer, it will be assumed that you want the same number of samples from each group. • If size is a vector, the function will check to see whether the length of the vector matches the number of groups and use those specified values as the desired sample sizes. The values in the vector should be in the same order as you would get if you tabulated the grouping variable (usually alphabetic order); alternatively, you can name each value to ensure it is properly matched.
select	A named list containing levels from the "group" variables in which you are interested. The list names must be present as variable names for the input data.frame.
seed	The seed that you want to use (using set.seed *within* the function, if any. Defaults to NULL.
...	Further arguments to be passed to the sample function.

Note

Slightly different sizes than requested

Because of how computers deal with floating-point arithmetic, and because R uses a "round to even" approach, the size per strata that results when specifying a proportionate sample may be slightly higher or lower per strata than you might have expected.

"Seed" argument

This is different from using set.seed before using the function. Setting a seed using this argument is equivalent to using set.seed each time that you go to take a sample from a different group (in other words, the same seed is used for each group).

The inclusion of a seed argument is mostly a matter of convenience, to be able to have a single seed with which the samples can be verified later. However, by using the seed argument, the same seed is used to sample from each group. This may be a problem if there are many groups that have the same number of observations, since it means that the same observation number will be selected from each of those grops. For instance, if group "AA" and "DD" both had the same number of observations (say, 5) and you were sampling 3 cases using a seed of 1, the second, fifth, and fourth observation would be taken from each of those groups. To avoid this, you can set the seed using set.seed *before* you run the stratified function.

As a user, you need to weigh the benefits and drawbacks of setting the seed *before* running the function as opposed to setting the seed *with* the function. Setting the seed before would be useful if there are several groups with the same number of observations; however, in the slim chance that you need to verify the samples manually, you *may* run into problems.

Author(s)

Ananda Mahto

References

The evolution of this function can be traced at the following links. The version in this package is entirely reworked and does not require an additional package to be loaded.

- http://news.mrdwab.com/2011/05/15/stratified-random-sampling-in-r-beta/
- http://news.mrdwab.com/2011/05/20/stratified-random-sampling-in-r-from-a-data-frame/
- http://stackoverflow.com/a/9714207/1270695

Examples

```
# Generate a couple of sample data.frames to play with
set.seed(1)
dat1 <- data.frame(ID = 1:100,
             A = sample(c("AA", "BB", "CC", "DD", "EE"), 100, replace = TRUE),
             B = rnorm(100), C = abs(round(rnorm(100), digits=1)),
             D = sample(c("CA", "NY", "TX"), 100, replace = TRUE),
             E = sample(c("M", "F"), 100, replace = TRUE))
dat2 <- data.frame(ID = 1:20,
             A = c(rep("AA", 5), rep("BB", 10),
                   rep("CC", 3), rep("DD", 2)))
# What do the data look like in general?
summary(dat1)
summary(dat2)

# Let's take a 10% sample from all -A- groups in dat1, seed = 1
stratified(dat1, "A", .1, seed = 1)

# Let's take a 10% sample from only "AA" and "BB" groups from -A- in dat1, seed = 1
stratified(dat1, "A", .1, select = list(A = c("AA", "BB")), seed = 1)

# Let's take 5 samples from all -D- groups in dat1,
#   seed = 1, specified by column number
stratified(dat1, group = 5, size = 5, seed = 1)

# Let's take a sample from all -A- groups in dat1, seed = 1,
#   where we specify the number wanted from each group
stratified(dat1, "A", size = c(3, 5, 4, 5, 2), seed = 1)

# Use a two-column strata: -E- and -D-
#   -E- varies more slowly, so it is better to put that first
stratified(dat1, c("E", "D"), size = .15, seed = 1)

# Use a two-column strata (-E- and -D-) but only interested in
#   cases where -E- == "M"
stratified(dat1, c("E", "D"), .15, select = list(E = "M"), seed = 1)

## As above, but where -E- == "M" and -D- == "CA" or "TX"
```

```
stratified(dat1, c("E", "D"), .15,
     select = list(E = "M", D = c("CA", "TX")), seed = 1)

# Use a three-column strata: -E-, -D-, and -A-
s.out <- stratified(dat1, c("E", "D", "A"), size = 2, seed = 1)

list(head(s.out), tail(s.out))

# How many samples were taken from each strata?
table(interaction(s.out[c("E", "D", "A")]))

# Can we verify the message about group sizes?
names(which(table(interaction(dat1[c("E", "D", "A")])) < 2))

names(which(table(interaction(s.out[c("E", "D", "A")])) < 2))
```

`string.to.color` *Convert between strings to colors*

Description

Automatically convert a vector of strings into a color for easy plotting

Usage

```
string.to.color(string, colors = NULL)
```

Arguments

`string`	a vector of strings representing groups.
`colors`	a vector of colors, one for each unique element in `string`.

Value

a vector of colors, one for each element in `string`

Note

This function can also be used to specify pch values, cex values, or any other plotting values the user may wish to differ across groups. See examples.

Author(s)

Dustin Fife

Examples

```
groups = sample(LETTERS[1:5], size=100, replace=TRUE)
plot(rnorm(100), rnorm(100), col=string.to.color(groups))
plot(rnorm(100), rnorm(100), col=string.to.color(groups),
   pch=as.numeric(string.to.color(groups, colors=c(16:20))))
```

---

subsetString *Extract only part of a string*

Description

Extract only part of a string, given a separator

Usage

```
subsetString(string, sep = " ", position = 3)
```

Arguments

string	a vector of strings
sep	the separator that separates the parts of the strings
position	the position of the element you wish to extract

Details

Given a string with a separator (e.g., "Subject 001", where the separator is a space), this function can be used to extract only whatever follows the separator (in this case, "001"). It is often used when data comes in with a conglomorated identifier (such as case-matchNumber-drawNumber-Month).

Value

the element the user wishes to extract

Author(s)

Dustin Fife

Examples

```
barcode = c("Case-001-B", "Control-001-A", "Case-002-A")
subsetString(barcode, sep="-", position=2)
subsetString(barcode, sep="-", position=3)
```

`SummarizeContinuousDefault` *Summarize a continuous vector with mean plus/minus standard deviation*

Description

Summarize a continuous variable with mean plus/minus standard deviation.

Usage

```
SummarizeContinuousDefault(x, group = rep(1, length(x)),
  decimal = 2, latex = TRUE, na.rm = TRUE, ...)
```

Arguments

`x`	Vector of values.
`group`	Group identifier to return summaries by group.
`decimal`	The number of decimal values to format the results; defaults to 2.
`latex`	Return LaTeX characters if `TRUE`; for example, the LaTeX code for the plus-minus symbol.
`na.rm`	Remove missing values if `TRUE`.
`...`	Nothing.

Details

Default `continuous.summary.function` for use in `SummarizeVar`. Returns formatted text of mean plus/minus standard deviation, possibly by group. For use in construction of demographics tables.

Value

Formatted text of mean plus/minus standard deviation in a vector or matrix.

Author(s)

Vinh Nguyen

Examples

```
SummarizeContinuousDefault(x=c(rnorm(100, 5), rnorm(100, 0)), group=rep(0:1, each=100))
```

SummarizeFactorDefault *Summarize a factor vector with count and percentages*

Description

Summarize a factor variable with count and percentages.

Usage

```
SummarizeFactorDefault(x, group = rep(1, length(x)),
  decimal = 0, latex = TRUE, useNA = "ifany", ...)
```

Arguments

x	Vector of values.
group	Group identifier to return summaries by group.
decimal	The number of decimal values to format the results; defaults to 0.
latex	Return LaTeX characters if TRUE (default). For example, the LaTeX code for the percentage symbol should be preceeded by the escape character }.
useNA	Defaults to ifany and passed to table.
...	Nothing.

Details

Default factor.summary.function for use in SummarizeVar. Returns formatted text of count and percentages. For use in construction of demographics tables.

Value

Formatted text of counts with percentages in parentheses, in a vector or matrix.

Author(s)

Vinh Nguyen

Examples

```
SummarizeFactorDefault(x=c(sample(1:5, 100, replace=TRUE), sample(1:5, 100, replace=TRUE)),
  group=rep(0:1, each=100))
```

SummarizeVar *Summarize a vector (continuous or factor)*

Description

Summarize a continuous or discrete (factor) vector.

Usage

```
SummarizeVar(x, group = rep(1, length(x)), latex = TRUE,
  decimalFactor = 0, decimalContinuous = 2,
  ContinuousSummaryFunction = SummarizeContinuousDefault,
  FactorSummaryFunction = SummarizeFactorDefault, ...)
```

Arguments

x	Vector of values.
group	Group identifiers to return summaries by group.
latex	Return LaTeX characters if TRUE (default). For example, the LaTeX code for the percentage symbol should be preceeded by the escape character \.
decimalFactor	The number of decimals to display in percentages for factor variables. This is passed to the decimal in factor.summary.function.
decimalContinuous	The number of decimals to display in percentages for numeric variables. This is passed to the decimal in ContinuousSummaryFunction.
ContinuousSummaryFunction	Function to use to summarize a continuous variable; defaults to SummarizeContinuousDefault. Function must take in the following arguments: x: a vector of values. group: a vector that identifies group. decimal: a numeric value to indicate the decimal places in the formatted output. latex: a logical value that indicates whether the resulting output contains LaTeX code; should default to TRUE. ...: additional arguments.
FactorSummaryFunction	Function to use to summarize a factor variable; defaults to SummarizeFactorDefault. See ContinuousSummaryFunction.
...	Arguments to be passed to ContinuousSummaryFunction and FactorSummaryFunction.

Value

Formatted text in a vector or matrix.

Author(s)

Vinh Nguyen

References

This function was borrowed (and modified) from Vinh Nguyen's day2day package.

`univariate.tests` *Extract p values for a data frame*

Description

Given a dataframe, this function predicts the specified categorical variable using each column in the dataset, one at a time. The function will automatically select whether to do a chi-square test, a t-test, or an ANOVA. See details.

Usage

```
univariate.tests(dataframe, exclude.cols = NULL, group)
```

Arguments

`dataframe`	a data frame containing both the variables and the grouping variable
`exclude.cols`	a vector indicating which columns should not have a significance test
`group`	a string with the name of the grouping variable

Details

Extract the p value from a univariate significance test

`univariate.tests` will look at each column in the dataframe, then perform a t-test, ANOVA, or chi-square test where the grouping variable serves as the independent variable. The computer will chose a chi-square test of one of the following three conditions is met: (1) the variable is a factor, (2) the variable is a character variable, or (3) the variable has less than four unique values. An ANOVA will be used if the number of levels of the grouping variable is greater than two. In all other cases, a t-test will be used.

Value

a vector of p values

Author(s)

Dustin Fife

Examples

```
k = data.frame(cbind(ID=1:100,
A = rnorm(100),
B = rnorm(100),
C = rnorm(100),
Group = rep(1:2, times=50)))
univariate.tests(dataframe = k, exclude.cols=1, group="Group")
```

Index

∗Topic **datasets**
fakeMedicalData, 12
∗Topic **data**
fakeMedicalData, 12
∗Topic **htest**
chisq.post.hoc, 4

auto.layout, 2

boxcox, *4*
boxcox.R *(*boxcoxR*)*, 3
boxcoxR, 3
boxplot, *9*, *19, 20*

calcT *(*compute.theta*)*, 6
chisq.post.hoc, 4
compute.theta, 6, *24*
contents, 6
cor2cov, 7, *17*

data.frame, *8*, *27*
demographics, 8
densityPlotR, 9, *19, 20*

ellipse, 10
excelCols, 10, *11*
excelMatch, *11*, 11

factor, *8*
fakeMedicalData, 12
formula, *8*

get.cols, 12, *15*, *21*

hash, 13

intersperse, 13

logical, *8*

make.formula, 14
make.null, *12*, 15, *21*
missing.Vals *(*missing.vals*)*, 16
missing.vals, 16
missingVals *(*missing.vals*)*, 16
missingvals *(*missing.vals*)*, 16
mv.Rnorm *(*mv.rnorm*)*, 16
mv.rnorm, 16
mvrnorm *(*mv.rnorm*)*, 16

na.pass, *8*

par1, 17
par2, 18
plots.prism *(*prism.plots*)*, 20
plotSigBars, *9*, 18, *20*
print.xtable, *8*, *19*
printx, 19
prism.plots, *9*, *19*, 20
prismPlots *(*prism.plots*)*, 20
prismplots *(*prism.plots*)*, 20

r, *12*, *15*, 21
r.crit, 22
random.correlation, *16, 17*, 22
rotate.graph *(*rotateGraph*)*, 23
rotateGraph, 23
rotategraph *(*rotateGraph*)*, 23
roxtemp, 24

sample, *28*
scaleB, 25
scaleIt, 26
set.seed, *28*
spearman.plot, 26
stratified, *27*, 27, *28*
string.to.color, 30
subset, *15*
subsetString, 31
SummarizeContinuousDefault, 32, *34*
SummarizeFactorDefault, 33, *34*
SummarizeVar, *8*, *32, 33*, 34

table, *33*

univariate.tests, 35

xtable, *8*