Package ‘fifer’

February 21, 2017

Title  A Biostatisticians Toolbox for Various Activities, Including
Plotting, Data Cleanup, and Data Analysis

Description  Functions and datasets that can be used for data cleanup (e.g., functions for eliminat-
ing all but a few columns from a dataset, selecting a range of columns, quickly editing col-
umn names), plotting/presenting data (prism-like reproductions, spearman plots for ordi-
nal data, making colored tables, plotting interactions with quantitative variables), and analy-
ses common to biostatistics (e.g., random forest, multiple compar-
isons with chi square tests). See the package vignette for a brief introduc-
tion to many of the main functions.

Version  1.1

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

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

License  GPL-2

Depends  MASS, stats

Imports  party, plotrix, randomForest, xtable, randomForestSRC, Hmisc,
rpart, fields

RoxygenNote  5.0.1

NeedsCompilation  no

Repository  CRAN

Date/Publication  2017-02-21 16:46:22

R topics documented:

anchored.gradient .................................................. 3
auto.layout .......................................................... 4
boxcoxR ............................................................... 5
chisq.post.hoc ....................................................... 6
clear ................................................................. 7
colored.table ......................................................... 8
compute.theta ....................................................... 9
contents ............................................................. 9
cor2cov .............................................................. 10
R topics documented:

demographics .................................................. 11
densityPlotR .................................................. 12
drop.columns .................................................. 13
elipse .......................................................... 14
excelCols ....................................................... 14
excelMatch ..................................................... 15
fakeMedicalData .............................................. 16
get.cols ......................................................... 16
gradient.legend ............................................... 17
hash ............................................................ 18
imageInteraction .............................................. 18
intersperse ..................................................... 19
last.sample ..................................................... 20
make.formula .................................................. 21
make.null ....................................................... 22
make.symmetric ............................................... 23
missing.vals ................................................... 23
mv.rnorm ....................................................... 24
number.to.colors ............................................ 25
par1 ............................................................. 26
par2 ............................................................. 26
plot.rfInterp .................................................. 27
plot.rfPred .................................................... 27
plot.rfThresh .................................................. 28
plotSigBars ................................................... 28
print.rfInterp ............................................... 29
print.rfPred ................................................... 29
print.rfThresh ............................................... 30
printx ........................................................ 30
prism.plots ................................................... 31
pval.xtable .................................................... 32
r ............................................................... 33
r.crit .......................................................... 34
random.correlation .......................................... 34
read.fife ....................................................... 35
rfInterp ....................................................... 36
rfPred .......................................................... 37
rfSensitivity .................................................. 38
rfThresh ....................................................... 39
rotateGraph ................................................... 41
roxtemp ....................................................... 42
scaleB ........................................................ 42
scaleBreak ..................................................... 43
scaleIt ........................................................ 44
spearman.plot ............................................... 45
stratified ....................................................... 46
string.to.colors ............................................. 48
subsetString ................................................... 49
anchored.gradient

Description

Create a color gradient with a color for zero

Usage

anchored.gradient(minColor = "blue", maxColor = "red", zeroColor = "white", z = NULL, vals = 100)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>minColor</td>
<td>what color corresponds to the minimum value?</td>
</tr>
<tr>
<td>maxColor</td>
<td>what color corresponds to the maximum value?</td>
</tr>
<tr>
<td>zeroColor</td>
<td>what color corresponds to the middle point value?</td>
</tr>
<tr>
<td>z</td>
<td>The vector of numbers corresponding to the colors?</td>
</tr>
<tr>
<td>vals</td>
<td>how many points should the final gradient have?</td>
</tr>
</tbody>
</table>

Details

Other functions (e.g., `number.to.colors`) create a color gradient by taking a range (min and max) and beginning with color 1 and ending with color 2. The problem is that sometimes, there's a critical middle point (e.g., zero) and the min and max are not symmetric around the middle point. This function alleviates that problem by creating a middle point for a fixed value (set to zero).

Value

a gradient of color

Author(s)

Dustin Fife
auto.layout

See Also

string.to.colors, number.to.colors.

Examples

# not shown yet

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 function 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){
Transform data using a boxcox transformation

Description

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

Usage

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

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.
- `lam`: Should the lambda value be returned?
- `...`: 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

chisq.post.hoc

Examples

```r
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

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tbl</td>
<td>A table object.</td>
</tr>
<tr>
<td>test</td>
<td>What sort of test will be used? This must have an object called p.value so it can correct the p-values. Defaults to &quot;fisher.test&quot;</td>
</tr>
<tr>
<td>popsInRows</td>
<td>A logical indicating whether the populations form the rows (default; =TRUE) of the table or not (=FALSE).</td>
</tr>
<tr>
<td>control</td>
<td>A string indicating the method of control to use. See details.</td>
</tr>
<tr>
<td>digits</td>
<td>A numeric that controls the number of digits to print.</td>
</tr>
<tr>
<td>...</td>
<td>Other arguments sent to whatever test the user specifies.</td>
</tr>
</tbody>
</table>

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.
clear

Note
This code was adapted and modified from the NCStats package

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)

clear

Clear memory of all objects

Description
This function is a wrapper for the command rm(list=ls()). It’s just less keystrokes.

Usage
clear(obj = NULL, keep = TRUE)

Arguments
obj The object (as a string) that needs to be removed (or kept)
keep Should obj be kept (i.e., everything but obj removed)? Or dropped?

Author(s)
Dustin Fife

Examples
#NOTE: ONLY RUN THIS IF YOU DON’T MIND ERASING YOUR R SESSION
a = 5
ls() ### a shows up
clear()
l$s() ### nothing shows up
### create objects
a=1; b=2; c=3; d=4; e=5
### remove c
clear("c", keep=FALSE)
l$s()
### remove all but a and b
colored.table

Description

Create a color-coded table in latex

Usage

colored.table(data, file, dep.var, row.factors, col.factors, breaks = 4,
round = 2, rng = NULL, FUN = mean, row.prefix = NULL,
col.prefix = NULL, custom.breaks = NULL, return.table = F, ...)

Arguments

data the dataset in "long" format, where the conditions are on columns.
file the location where the file should be exported.
dep.var A string naming the dependent variable
row.factors A vector of one or two elements, each of which is a string indicating the name
of the row factor(s)
col.factors A vector of one or two elements, each of which is a string indicating the name
of the col factor(s)
breaks The number of breaks in color. Defaults to 4.
round How many digits should the matrix be rounded to? Defaults to 2.
rng The range of values used for coloring the plot.
FUN The function used to summarize the table (defaults to mean)
row.prefix Any values that precede the rows (e.g., "r^2 = ").
col.prefix Any values that precede the columns.
custom.breaks The points at which the colored table changes colors. If left to NULL, the func-
tion will automatically choose them.
return.table Should the function return the table as output?
... Other arguments passed to Hmisc

Author(s)

Dustin Fife

See Also

Hmisc::latex

Examples

## do this later
**compute.theta**

**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**

**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.
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**

```r
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**

```r
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)
```
densityPlotR

Generate a density plot using a formula

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

Description

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

Usage

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

Author(s)

Dustin Fife

See Also

boxplot, prism.plots, plotSigBars

Examples

densityPlotR(Petal.Width~Species, data=iris)
**drop.columns**

*Drop (or keep) all columns containing a vector of strings*

**Description**

Drop (or keep) all columns containing a vector of strings.

**Usage**

```r
drop.columns(string, data, drop = TRUE)
```

**Arguments**

- `string`: The string (or vector of strings) to be kept or omitted.
- `data`: The dataset.
- `drop`: Logical. Should the columns containing `string` be dropped?

**Value**

A dataset with the column(s) dropped or kept.

**Author(s)**

Dustin Fife

**References**

http://stackoverflow.com/questions/7597559/grep-in-r-with-a-list-of-patterns

**See Also**

`make.null`

**Examples**

```r
### drop all columns with the words "Length"
data(iris)
iris.dropped = drop.columns("Length", data=iris, drop=TRUE)

### keep only those columns with HemoLeptin in the string
data(fakeMedicalData)
medical = drop.columns("HemoLeptin", data=fakeMedicalData, drop=FALSE)
head(medical)
```
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
**Description**

Obtain column number or variable name from Excel named Columns

**Usage**

```r
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

```r
defake.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**

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

gradient.legend
Create a gradiented legend

Description
Create a gradiented legend

Usage

\[
\text{gradient.legend}(y = \text{NULL}, \ yrange = \text{NULL}, \ cols = c("red", "blue"), \\
\text{location} = c(0.075, 0.3, 0.575, 0.975), \ n = 100, \ ...)
\]

Arguments

- **y**: the variable used to create the gradient, typically in `number.to.colors`
- **yrange**: The range of y values. If y is supplied, it will pull these from the actual y values.
- **cols**: The color gradients to be used that are passed to `colorRampPalette`.
- **location**: The location of the subplot, expressed in fractions of the entire plot (left x, right x, bottom y, top y).
- **n**: the number of values used for the gradient. Higher numbers make a higher resolution
- **...**: other arguments passed to `image`

Author(s)
Dustin Fife

See Also

`number.to.colors` `colorRampPalette`

Examples

\[
y = \text{rnorm}(100); \ x = .6*y + \text{rnorm}(100,0,\text{sqrt}(1-0.6^2)) \\
\text{randnum} = \text{runif}(100) \\
\text{plot}(\text{x,y, col=number.to.colors(randnum), pch=16)} \\
\text{gradient.legend(randnum, xlab="", ylab="")}
\]
# 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\(j=100, i=11\)

---

# imageInteraction

Plot an two-way quantitative interaction in an image plot

---

**Description**

Plot an two-way quantitative interaction in an image plot

**Usage**

imageInteraction(data, x, y, z, plot = TRUE, legend = TRUE, ...)

**Arguments**

- **data** a data matrix with columns indicating the values of \(x\), \(y\), and \(z\)
- **x** the name of the \(x\) variable (a string)
- **y** the name of the \(y\) variable (a string)
- **z** the name of the \(z\) variable (a string)
- **plot** Should an image be plotted?
- **legend** add a legend to the plot?
- **...** other arguments passed to anchored.gradient.
The `image` function is quite useful for displaying two-way quantitative interactions (variable X on the X axis, variable Y on the Y axis, and z represented as a color). However, this function requires a matrix where the cells \((r_{ij})\) represent the Z value in the ith value of x and jth value of y. Getting it into matrix form can be annoying if it natively comes in long-column format. This function takes a data matrix, conditions as well as z in the columns, and returns an image matrix.

**Value**

the x range, y range, and image matrix

**Author(s)**

Dustin Fife

**See Also**

`image`

**Examples**

```r
## do this later
```

---

**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 \(i \times j\), 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 \(i \times j\), containing the interspersed vectors as strings

**Author(s)**

Dustin Fife
last.sample

Return only one row per ID

Description

Often times, an individual has multiple observations in the dataset. last.sample will loop through the entire dataset and return only one observation per individual, giving the first (or last) draw for a person, or performing some function on the variable of interest.

Usage

last.sample(ID, sort.var = NULL, decreasing = TRUE, data, FUN = NULL, fun.var = NULL, ...)

Arguments

ID The name of the unique identifier, expressed as a string (e.g., "Match.Group")
sort.var The variable to be sorted on in order to take the first (or last) sample, expressed as a string.
decreasing How should the sort.var variable be sorted? Defaults to T.
data The dataset with multiple rows per individual
FUN What should be done with the multiple samples? This function can be used to extract the last (or first) sample using the decreasing/sort.var options, or a function can be performed (such as the mean) on one or more columns. See examples.
fun.var If FUN if not null, the variable (or a vector of variables), expressed as strings to have the function applied to.
... Other arguments passed to the chosen function.

Value

a new dataframe containing one row per ID

Author(s)

Dustin Fife
Examples

```r
### take only group 2 values
last.sample(ID="ID", sort.var="group", data=sleep)

### take only group 1 values
last.sample(ID="ID", sort.var="group", decreasing=FALSE, data=sleep)

### average group 1 and 2 values
last.sample(ID="ID", data=sleep, FUN=mean, fun.var="extra")

### take the maximum extra value
last.sample(ID="ID", data=sleep, FUN=max, fun.var="extra")

### take the mean of two columns extra value
sleep$group = as.numeric(as.character(sleep$group))
last.sample(ID="ID", data=sleep, FUN=mean, fun.var=c("group","extra"))
```

---

**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, random = NULL)
```

**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.
- `random`  
a string that indicates the random component in an lmer-like object (e.g., ``(1|group)``). Defaults to NULL.

**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)
# do a random model
make.formula("A", LETTERS[2:5], random="(1|group)"

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

```r
new.data2 = make.null(c("Sepal.Length"),
  r("Sepal.Width", "Petal.Width", names(iris)), data=iris, keep=TRUE)
```

---

**make.symmetric**  
*Force a matrix to be symmetric*

#### Description

Force a matrix to be symmetric

#### Usage

```r
make.symmetric(a, lower.tri = TRUE)
```

#### Arguments

- `a`: A matrix you wish to force to be symmetrical
- `lower.tri`: Should the upper triangle be replaced with the lower triangle?

#### Value

A symmetric matrix

#### Author(s)

Dustin Fife

#### Examples

```r
a = matrix(rnorm(16), ncol=4)
make.symmetric(a, lower.tri=FALSE)
```

---

**missing.vals**  
*Summary of Missing Data*

#### Description

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

#### Usage

```r
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

mvNrnorm
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
mvNrnorm(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 definite matrix. If NULL, the user must specify vars.
names Optional. A vector of strings indicating the variable names.

Details
`mvNrnorm` 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 n*p matrix of pseudo-random values.

Author(s)
Dustin Fife
number.to.colors

Convert from numbers to colors

Description

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

Usage

number.to.colors(value, colors = c("red", "blue"), num = 100)

Arguments

value a vector of numbers.

colors a vector of two or more colors representing the inflection points of the gradients, passed to colorRampPalette.

num The number of unique intervals for colors. Chose larger numbers for finer gradients (higher resolution).

Value

a vector of colors.
Author(s)
Dustin Fife

See Also
string.to.colors colorRampPalette gradient.legend

Examples

```r
### plot three variables, represent the third with a color
d = mvrnorm(100, mu=c(0,0,0), Sigma=matrix(c(1, .6, .6, .6, 1, .6, .6, .6, .6, 1), ncol=3))
plot(d[,1:2], col=number.to.colors(d[,3], c("black", "pink")), pch=16)
```

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
plot.rfInterp  

Plot rfInterp Summary

Description

Plot rfInterp Summary

Usage

```r
## S3 method for class 'rfInterp'
plot(x, y, ...)
```

Arguments

- `x` an rfInterp object
- `y` ignored
- `...` other parameters passed to `plot`

plot.rfPred  

Plot rfPred Summary

Description

Plot rfPred Summary

Usage

```r
## S3 method for class 'rfPred'
plot(x, y, ...)
```

Arguments

- `x` an rfPred object
- `y` ignored
- `...` other parameters passed to `plot`
plot.rfThresh  

rfThresh Summary

Description
Plot a rfThresh object

Usage
## S3 method for class 'rfThresh'
plot(x, y, ...)

Arguments
- x: an rfThresh object
- y: ignored
- ...: other parameters passed to plot

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")

Description
Print rfInterp Summary

Usage
## S3 method for class 'rfInterp'
print(x, ...)

Arguments
x an rfInterp object
... ignored

Description
Print rfPred Summary

Usage
## S3 method for class 'rfPred'
print(x, ...)

Arguments
x an rfPred object
... ignored
print.rfThresh

Print rfThresh Summary

Description

Print a rfThresh object

Usage

```r
## S3 method for class 'rfThresh'
print(x, ...)
```

Arguments

- `x` an rfThresh object
- `...` ignored

printx

Change Defaults of print.xtable

Description

Change Defaults for print.xtable

Usage

```r
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 = median, interquartile = TRUE,
spreadfunc = function(x) { return(1.96 * sd(x)/sqrt(length(x))) },
def.axis = TRUE, jitter.y = FALSE, add = FALSE, start = 0,
col = "gray", ...)

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 median.
interquartile Should the interquartile range be plotted? Defaults to TRUE.
spreadfunc what function should be used to calculate the spread of the distribution? If interquartile=TRUE, this argument will be ignored. The default (when not ignored) is to produce a 95% confidence interval (1.96*sd(x)/sqrt(n)).
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)
col What color should the dots be painted? Defaults to gray.
... 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

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

---

**pval.xtable**  
Convert p-values into strings with inequalities.

### Description

Often times the p values are so small they must either be expressed with bajillions of digits, or in scientific notation. It is common to simply state that $p < .0001$, or something of the sort. This function takes a vector of p-values and converts them into characters so they can be expressed with inequalities.

### Usage

```r
pval.xtable(p.values, round.digits = 4, threshold = 1e-04, scipen = FALSE)
```

### Arguments

- **p.values**: A vector (or scalar) of p-values.
- **round.digits**: How many digits should the decimals be rounded to during display?
- **threshold**: The threshold for the inequalities. This value will then be joined with the inequality via `paste`. (e.g., threshold = .01 becomes "<.01")
- **scipen**: Should scientific notation be used? Defaults to zero. (Note: it will not entirely suppress scientific notation, but will make it so extreme it will likely never happen).

### Value

A vector of strings yielding the formatted p-values

### Author(s)

Dustin Fife

### Examples

```r
p.values = runif(100,0,1)
pval.xtable(p.values, round.digits=2, threshold=.01)
pval.xtable
```
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**

\[
\text{r.crit}(n, \, r = \text{NULL}, \, p = 0.025, \, \text{two.tailed} = T)
\]

**Arguments**

- **n**: The sample size.
- **r**: The observed r value.
- **p**: the probability. Defaults to .025.
- **two.tailed**: Should the probability be cut in half?

**Value**

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

**Author(s)**

Dustin Fife

**Examples**

\[
\begin{align*}
\text{r.crit}(n=100, \, p=.025) \\
\text{r.crit}(n=20, \, r=.6, \, p=.05)
\end{align*}
\]

---

**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**

\[
\text{random.correlation}(n, \, ev = \text{runif}(n, \, 0, \, 10))
\]
**read.fife**

**Arguments**

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

**Value**

a correlation matrix, of size \( n \times n \)

**Author(s)**

Dustin Fife

**References**


---

**read.fife** Read in a dataset and load the meta-data for a file

**Description**

Oftentimes the original data matrix is too large to practically read in every time you want to do analysis. This often means creating a separate file for analysis. Unfortunately, if the original file is changed, the separate file doesn’t reflect those changes. read.fife and write.fife both read and write meta-data, then display the original file name for the meta data.

**Usage**

```
read.fife(file, file.type = ".csv", ...)
```

**Arguments**

- `file` The location of the file to be read.
- `file.type` The extension of the dataset. Defaults to `.csv`.
- `...` other arguments passed to to `read.csv`

**Details**

Technically, read.fife and write.fife don’t actually read and write meta-data. Instead, they create (or read) a separate file that has the same name (though different extension) than the subsetted dataset. The extension of the meta data file is `.file`.

**Value**

An R object containing the subsetted dataset. Also, an object called `original.file` will be loaded into the R environment that contains a string of the original file location.
**Author(s)**
Dustin Fife

**See Also**
write.fife

---

**rfInterp**

**Variable Selection with Random Forest**

---

**Description**

Find best variables using Random Forest (Interpretation Step). Find complete documentation at `rfThresh`

**Usage**

`rfInterp(object, nruns = 20, nsd = 1, importance = "permutation", ...)`

**Arguments**

- **object**: a rfThresh object
- **nruns**: how many forests should be grown?
- **nsd**: defaults to one.
- **importance**: method of calculating importance (permutation or gini)
- **...**: other arguments passed to `cforest` or `randomForest`

**Value**

- **varselect.interp**: The variables selected for Interpretation (sorted)
- **err.interp**: The error at each stage of the stepwise variable inclusion.
- **sd.min**: The standard deviation of the minimum fitted value.
- **num.varselect.interp**: The number of variables selected for interpretation.
- **comput.time**: Computation time of the procedure.
- **data**: The dataset used for fitting the RF algorithm
- **formula**: The formula of all variables included after the interpretation step.

**Author(s)**
Robin Genuer, Jean-Michel Poggi and Christine Tuleau-Malot, with modifications by Dustin Fife

**See Also**

`rfInterp, rfPred`
Examples

```r
### do threshold step
# Not run: data(iris);
data = iris;
formula = as.formula("Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width")
object = rfThresh(formula, data=iris, nruns=2, importance="gini")
### run interpretation step
rfInterp(object, nruns=10, importance="permutation")
# End(Not run)
```

---

**rfPred**

### Variable selection in Random Forest

**Description**

Variable selection for prediction purposes using Random Forest. See rfThresh for complete documentation.

**Usage**

```r
rfpred(object, importance = "permutation", nfor.pred = 25, nmj = 1,
       outfile = "rfPred.file", named.file = "rfPredResults", ...)
```

**Arguments**

- **object**: an object returned from rfInterp
- **importance**: what importance measure should be used? Either "permutation" or "gini."
- **nfor.pred**: number of forests to grow
- **nmj**: a constant used for setting the threshold for variable selection. Higher values indicate a less stringent threshold.
- **outfile**: The file location where the rfPred object should be stored. Defaults to storing it in rfPred.file in the default directory.
- **named.file**: What should the rfPred object be named when saved? Defaults to "rfPredResults".
- **...**: other arguments passed to cforest or randomforest

**Value**

- **varselect.pred**: The variables selected for Prediction (sorted)
- **err.interp**: The error at each stage of the stepwise variable inclusion.
- **mean.jump**: The threshold for variable inclusion.
- **stepwise.error**: The OOB error rate at each iteration of the stepwise procedure.
- **num.varselect.pred**: The number of variables selected for prediction.
- **comput.time**: Computation time of the procedure.
- **model**: The final model, either a randomforest or cforest object.
rfSensitivity

Author(s)
Robin Genuer, Jean-Michel Poggi and Christine Tuleau-Malot, with modifications by Dustin Fife

See Also
rfInterp, rfThresh

Examples

```r
## Not run: data(iris);
data = iris;
formula = as.formula("Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width")
thresh = rfThresh(formula, data=iris, nruns=2, importance="permutation");
interp = rfInterp(thresh, importance="permutation");
predic = rfPred(interp, importance="gini");
predic
## End(Not run)
```

rfSensitivity

Output accuracy, sensitivity, specificity, NPV, and PPV.

Description

Output accuracy, sensitivity, specificity, NPV, and PPV.

Usage

rfSensitivity(object)

Arguments

object An rfPred object

Value

A list containing the accuracy, sensitivity, etc.
**Description**

Using a set of predictors, this function uses random forests to select the best ones in a stepwise fashion. Both the procedure and the algorithm were borrowed heavily from the VSURF package with some modifications. These modifications allow for unbiased computation of variable importance via the `cforest` function in the party package.

**Usage**

```r
rftThresh(formula, data, nruns = 50, silent = FALSE, importance = "permutation", nmin = 1, ...)
```

**Arguments**

- `formula`: a formula, such as `y~x1 + x2`, where `y` is the response variable and anything following `~` are predictors.
- `data`: the dataset containing the predictors and response.
- `nruns`: How many times should random forests be run to compute variable importance? Defaults to 50.
- `silent`: Should the algorithm talk to you?
- `importance`: Either "permutation" or "gini."
- `nmin`: Number of times the "minimum value" is multiplied to set threshold value.
- `...`: other arguments passed to `cforest` or `randomForest`

**Details**

What follows is the documentation for the original algorithm in VSURF:

Three steps variable selection procedure based on random forests for supervised classification and regression problems. First step ("thresholding step") is dedicated to eliminate irrelevant variables from the dataset. Second step ("interpretation step") aims to select all variables related to the response for interpretation purpose. Third step ("prediction step") refines the selection by eliminating redundancy in the set of variables selected by the second step, for prediction purpose.

- First step ("thresholding step"): first, `nfor.thres` random forests are computed using the function `randomForest` with arguments `importance=TRUE`. Then variables are sorted according to their mean variable importance (VI), in decreasing order. This order is kept all along the procedure. Next, a threshold is computed: `min.thres`, the minimum predicted value of a pruned CART tree fitted to the curve of the standard deviations of VI. Finally, the actual "thresholding step" is performed: only variables with a mean VI larger than `nmin * min.thres` are kept.
• Second step ("intepretation step"): the variables selected by the first step are considered. nfor_interp embedded random forests models are grown, starting with the random forest build with only the most important variable and ending with all variables selected in the first step. Then, err_min the minimum mean out-of-bag (OOB) error of these models and its associated standard deviation sd_min are computed. Finally, the smallest model (and hence its corresponding variables) having a mean OOB error less than err_min + nsd * sd_min is selected.

• Third step ("prediction step"): the starting point is the same than in the second step. However, now the variables are added to the model in a stepwise manner. mean_jump, the mean jump value is calculated using variables that have been left out by the second step, and is set as the mean absolute difference between mean OOB errors of one model and its first following model. Hence a variable is included in the model if the mean OOB error decrease is larger than nmj * mean_jump.

Value

The object returned has the following attributes:

variable.importance A sorted vector of each variable importance measures.
importance.sd the standard deviation of variable importance, measured across the nruns iterations.
stepwise.error The OOB error after each variable is added to the model
response The response variable that was modeled.
variables A vector of strings that indicate which variables were included in the initial model.
nruns How many times the random forest was initially run.
formula the formula used for the last model.
data the dataset used to fit the model.
oob the oob error of the entire model.
time how long the algorithm ran for
rfmodel The final model used, a randomForest object.

Author(s)

Robin Genuer, Jean-Michel Poggi and Christine Tuleau-Malot, with modifications by Dustin Fife

References


See Also

rfInterp, rfPred
**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**

```r
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**

See Also
- compute.theta.

---

**scaleB**

**Description**

Generic function for obtaining scaled coefficients

**Usage**

```r
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

```r
### 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)
```

Description

With skewed data, important relationships are often scaled so small because the graph attempts to capture the outliers. `scaleBreak` will plot two different scales on the same plot, shrinking the range where the outliers lay. The break in scale is indicated with a mark.

Usage

```r
scaleBreak(x, y, axis = 2, breakpos = 1, plot.numbers = c(1, 2), ...)
```

Arguments

- `x`: The x variables to be plotted.
- `y`: The y variables to be plotted.
- `axis`: Which axis should have two different scales. Currently, only implemented for “2,” which is the y axis.
- `breakpos`: At what point should the break occur? Defaults to 1.
- `plot.numbers`: Which plots to include. Can be 1, 2, or c(1,2).
- `...`: other arguments passed to plot.
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

```r
### 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, replace = FALSE, bothSets = FALSE)`

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.
replace  Logical. Should the sampling be done with replacement?
bothSets  Logical. Should just the samples be returned, or a list with two items: the sampled subset and the unsampled subset?

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.
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
stratified(dat1, "A", .1)

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

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

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

# 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)

# 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"))

## 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")))

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

# How many samples were taken from each strata?
String to Colors

Convert between strings to colors

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

Usage
string.to.colors(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

See Also
number.to.colors

Examples
groups = sample(LETTERS[1:5], size=100, replace=TRUE)
plot(rnorm(100), rnorm(100), col=string.to.colors(groups))
plot(rnorm(100), rnorm(100), col=string.to.colors(groups),
pch=as.numeric(string.to.colors(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, flexible = FALSE)
```

**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
- `flexible`: Force the function to find the string "sep"? See details

**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 conglomerated identifier (such as case-matchNumber-drawNumber-Month).

The "flexible" function is for instances where a particular string (i.e., sep) is found in only some elements of string. For example, if the string is c("...call..ID", "...call..Gender", "ethnicity"), if flexible is false, it will search for the string "...call.." in the string "ethnicity", and, not finding it, will return NA. To overcome this, flexible tells the function to only perform the operation on those parts of the vector that contain the string sep.

**Value**

the element the user wishes to extract

**Author(s)**

Dustin Fife

**Examples**

```r
code = c("Case-001-B", "Control-001-A", "Case-002-A", "001")
subsetString(code, sep="-", position=2, flexible=TRUE)
subsetString(code, sep="-", position=3, flexible=TRUE)
subsetString(code, sep="-", position=3, flexible=FALSE)
```
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))
Summarize Factor Default

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 preceded by the escape character \\.
- useNA: Defaults to ifany and passed to table.
- ...: Nothing.

Details

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))
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 preceded 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
**summary.rfPred**  
*Print a Summary Table of rfPred*

---

## References
This function was borrowed (and modified) from Vinh Nguyen’s day2day package.

---

## Summary

**summary.rfPred** is best for those non-Latex users to produce a table that shows each stage of the variable selection algorithm.

### Usage

```r
## S3 method for class 'rfPred'
summary(object, ...)
```

### Arguments

- **object**: an rfPred object
- **...**: additional arguments affecting the summary produced.

### Details

Print a Summary Table of rfPred

---

## unfactor

*Convert a factor to a character (or number)*

---

## Description

Analogous to the function `factor`, `unfactor` will convert a factor to a character or integer.

### Usage

```r
unfactor(x, levels, labels, numeric = FALSE)
```

### Arguments

- **x**: The vector of factors you wish to replace
- **levels**: A numeric vector of the values that will replace the factors
- **labels**: A character vector of the values to be replaced
- **numeric**: Should the labels be returned (i.e., characters)? Or the numbers?
Value
A numeric or character vector

Author(s)
Dustin Fife

See Also
factor

Examples

```r
### create sample of male/female participants (at random)
x = as.factor(sample(c("Male", "Female"), 12, replace=TRUE))
unfactor(x, levels=c(0,1), labels=c("Male", "Female"))
```

---

**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**

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

**Arguments**

- **dataframe**: a data frame containing both the variables and the grouping variable
- **exclude.cols**: a vector indicating (either numeric or character) which columns should not have a significance test
- **group**: a string with the name of the grouping variable
- **parametric**: Should parametric tests be used? Defaults to TRUE.
- **...**: other arguments passed to t.test or wilcox.test
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 (or `wilcoxon.test` if `parametric=TRUE`), ANOVA (or `kruskal.test` if `parametric=TRUE`), 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 (or Kruskall) will be used if the number of levels of the grouping variable is greater than two. In all other cases, a t-test (or `wilcoxon`) will be used.

Value

a vector of p values

Author(s)

Dustin Fife

Examples

```r
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")
```

write.fife

Write a dataset and load the meta-data

Description

Oftentimes the original data matrix is too large to practically read in everytime you want to do analysis. This often means creating a separate file for analysis. Unfortunately, if the original file is changed, the separate file doesn’t reflect those changes. `read.fife` and `write.fife` both read and write meta-data, then display the original file name for the meta data.

Usage

```r
write.fife(object, newfile, originalfile = NULL, file.type = ".csv",
row.names = F, fullpath = T, ...)
```
**Arguments**

- `object`: An R object to be written as a .csv (or whatever) file.
- `newfile`: The location of the subsetted dataset to be written.
- `originalfile`: The location of the original file that was subsetted.
- `file.type`: The file type to be read. Defaults to .csv.
- `row.names`: Should row names be written? Defaults to FALSE.
- `fullpath`: Should the full path be written to the meta-data? (e.g., "documents/research/datasets/medical_data_ap9_2014.csv"). Defaults to T.
- `...`: Other arguments passed to `write.csv`.

**Details**

Technically, `read.fife` and `write.fife` don’t actually read and write meta-data. Instead, they create (or read) a separate file that has the same name (though different extension) than the subsetted dataset. The extension of the meta data file is .file.

**Author(s)**

Dustin Fife

**See Also**

- `read.fife`

---

**xtable.rfInterp**

Prepare xtable Summary

**Description**

Prepare xtable Summary

**Usage**

```r
## S3 method for class 'rfInterp'
xtable(x, caption = NULL, label = NULL, align = NULL,
       digits = NULL, display = NULL, ...)
```

**Arguments**

- `x`: an rfInterp object
- `caption`: Character vector of length 1 or 2 containing the table’s caption or title. If length 2, the second item is the "short caption" used when LaTeX generates a "List of Tables". Set to NULL to suppress the caption. Default value is NULL.
- `label`: Character vector of length 1 containing the LaTeX label or HTML anchor. Set to NULL to suppress the label. Default value is NULL.
align Character vector of length equal to the number of columns of the resulting table indicating the alignment of the corresponding columns. Also, "|" may be used to produce vertical lines between columns in LaTeX tables, but these are effectively ignored when considering the required length of the supplied vector. If a character vector of length one is supplied, it is split as `strsplit(align, "")[[1]]` before processing. Since the row names are printed in the first column, the length of align is one greater than ncol(x) if x is a data.frame. Use "l", "r", and "c" to denote left, right, and center alignment, respectively. Use "p\{3cm\}" etc for a LaTeX column of the specified width. For HTML output the "p" alignment is interpreted as "1", ignoring the width request. Default depends on the class of x.

digits Numeric vector of length equal to one (in which case it will be replicated as necessary) or to the number of columns of the resulting table or matrix of the same size as the resulting table indicating the number of digits to display in the corresponding columns. Since the row names are printed in the first column, the length of the vector digits or the number of columns of the matrix digits is one greater than ncol(x) if x is a data.frame. Default depends on class of x. If values of digits are negative, the corresponding values of x are displayed in scientific format with `abs(digits)` digits.

display Character vector of length equal to the number of columns of the resulting table indicating the format for the corresponding columns. Since the row names are printed in the first column, the length of display is one greater than ncol(x) if x is a data.frame. These values are passed to the `formatC` function. Use "d" (for integers), "f", "e", "E", "g", "G", "fg" (for reals), or "s" (for strings). "f" gives numbers in the usual xxx.xxx format; "e" and "E" give n.ddde+nn or n.ddde+nn (scientific format); "g" and "G" put x[i] into scientific format only if it saves space to do so. "fg" uses fixed format as "f", but digits as number of significant digits. Note that this can lead to quite long result strings. Default depends on the class of x.

... other arguments passed to `xtable`

---

**xtable.rfPred**

Prepare xtable Summary

**Description**

Print xtable Summary

**Usage**

```
## S3 method for class 'rfPred'
xtable(x, caption = NULL, label = NULL, align = NULL, digits = NULL, display = NULL, ...)
```
z.test

Perform a z test

Description

This function does the standard z-test, which tests a particular dataset against a specified value

Arguments

x an rfPred object
caption Character vector of length 1 or 2 containing the table's caption or title. If length 2, the second item is the "short caption" used when LaTeX generates a "List of Tables". Set to NULL to suppress the caption. Default value is NULL.
label Character vector of length 1 containing the LaTeX label or HTML anchor. Set to NULL to suppress the label. Default value is NULL.
align Character vector of length equal to the number of columns of the resulting table indicating the alignment of the corresponding columns. Also, "|" may be used to produce vertical lines between columns in LaTeX tables, but these are effectively ignored when considering the required length of the supplied vector. If a character vector of length one is supplied, it is split as strsplit(align, "")[[1]] before processing. Since the row names are printed in the first column, the length of align is one greater than ncol(x) if x is a data.frame. Use "1", "r", and "c" to denote left, right, and center alignment, respectively. Use "p\{3cm\}" etc for a LaTeX column of the specified width. For HTML output the "p" alignment is interpreted as "1", ignoring the width request. Default depends on the class of x.
digits Numeric vector of length equal to one (in which case it will be replicated as necessary) or to the number of columns of the resulting table or matrix of the same size as the resulting table indicating the number of digits to display in the corresponding columns. Since the row names are printed in the first column, the length of the vector digits or the number of columns of the matrix digits is one greater than ncol(x) if x is a data.frame. Default depends on class of x. If values of digits are negative, the corresponding values of x are displayed in scientific format with abs(digits) digits.
display Character vector of length equal to the number of columns of the resulting table indicating the format for the corresponding columns. Since the row names are printed in the first column, the length of display is one greater than ncol(x) if x is a data.frame. These values are passed to the formatC function. Use "d" (for integers), "f", "e", "E", "g", "G", "fg" (for reals), or "s" (for strings). "f" gives numbers in the usual xxx.xxx format; "e" and "E" give n.ddde+nn or n.ddde+nn (scientific format); "g" and "G" put x[1] into scientific format only if it saves space to do so. "fg" uses fixed format as "f", but digits as number of significant digits. Note that this can lead to quite long result strings. Default depends on the class of x.
... other arguments passed to xtable
Usage

z.test(data, mean = 0, sd = 1, direction = "both")

Arguments

data a vector of values which will be used to compute a mean
mean the value the z-test is tested against
sd the standard deviation of the null distribution
direction either "positive", "negative", or "both" (for a two-tailed test)

Details

Perform a z-test in R
z.test will take a vector and determine whether it differs from a chosen value specified by mean

Value

a list containing the computed mean, the z-statistic, and the p-value

Author(s)

Dustin Fife

Examples

k = rnorm(15, 10, 3)
z.test(k, mean=5, sd=3)
Index

*Topic **data**
  fakeMedicalData, 16
*Topic **htest**
  chisq.post.hoc, 6

anchor.gradient (anchored.gradient), 3
anchored.gradient, 3, 18
auto.layout, 4
boxcox, 5
boxcox.R (boxcoxR), 5
boxcoxR, 5
boxplot, 12, 29, 31
calcT (compute.theta), 9
cforest, 36, 37, 39
chisq.post.hoc, 6
clear, 7
col.table (colored.table), 8
color.gradient (anchored.gradient), 3
color.table (colored.table), 8
colored.table, 8
colorRampPalette, 17, 25, 26
compute.theta, 9, 42
contents, 9
cor2cov, 10, 25
data.frame, 11, 46
demographics, 11
densityPlotR, 12, 29, 31
drop.columns, 13
ellipse, 14
excels, 14, 15
excelMatch, 15, 15
factor, 11, 53, 54
fakeMedicalData, 16
formula, 11
get.cols, 16, 22, 33

gradient (gradient.legend), 17
gradient.legend, 17, 26
hash, 18
image, 19
imageInteraction, 18
integer.to.colors (number.to.colors), 25
integers.to.colors (number.to.colors), 25
intersperse, 19
last.row (last.sample), 20
last.sample, 20
lastSample (last.sample), 20
lastsample (last.sample), 20
logical, 11
make.formula, 21
make.null, 13, 16, 22, 33
make.symmetric, 23
missing.Vals (missing.vals), 23
missing.vals, 23
missing.Vals (missing.vals), 23
missingvals (missing.vals), 23
mv.Rnorm (mv.rnorm), 24
mv.rnorm, 24
mv.rnorm (mv.rnorm), 24
na.pass, 11
number.to.color (number.to.colors), 25
number.to.colors, 3, 4, 17, 25, 48
numbers.to.colors (number.to.colors), 25
numberToColor (number.to.colors), 25
numberToColors (number.to.colors), 25
one.row (last.sample), 20
p.val.xtable (pval.xtable), 32
p.value (pval.xtable), 32
p.values (pval.xtable), 32
INDEX

p.xtable(pval.xtable), 32
par1, 26
par2, 26
plot(plot.plot.rfInterp), 27
plot.plot.rfInterp, 27
plot.plot.rfPred, 27
plot.plot.rfThresh, 28
plots.prism(prism.plots), 31
print.plot.plot.rfInterp, 29
print.plot.plot.rfPred, 29
print.plot.plot.rfThresh, 30
print.plot.xtable, 11, 30
printx, 30
prism.plots, 12, 29, 31
prismPlots(prism.plots), 31
prismplots(prism.plots), 31
pval.xtable, 32
r, 16, 22, 33
r.crit, 34
random.correlation, 24, 25, 34
randomForest, 36, 37, 39
read.csv, 35
read.fife, 35, 56
read.fifer(read.fife), 35
read.meta(read.fife), 35
read.metadata(read.fife), 35
rfInterp, 36, 37, 38, 40
rfPred, 36, 37, 40
rfSensitivity, 38
rfThresh, 36–38, 39
rotate.graph(rotateGraph), 41
rotateGraph, 41
rotategraph(rotateGraph), 41
roxtemp, 42
scale.Break(scaleBreak), 43
scale.break(scaleBreak), 43
scaleB, 42
scaleBreak, 43
scalebreak(scaleBreak), 43
scaleIt, 44
spearman.plot, 45
stratified, 46, 46
string.to.color(string.to.colors), 48
string.to.colors, 4, 26, 48
stringtocolor(string.to.colors), 48
stringToColors(string.to.colors), 48
subset, 22
substring, 49
SummarizeContinuousDefault, 50, 52
SummarizeFactorDefault, 51, 52
SummarizeVar, 11, 50, 51, 52
summary.rfPred, 53
table, 51
unfactor, 53
univariate.tests, 54
write.csv, 56
write.fife, 35, 36, 55, 56
write.fifer(write.fife), 55
write.meta(write.fife), 55
write.metadata(write.fife), 55
xtable, 11
xtable.rfInterp, 56
xtable.rfPred, 57
z.test, 58