Package 'useful'

October 12, 2022

Type Package

Title A Collection of Handy, Useful Functions

Version 1.2.6

Date 2018-10-07

Description A set of little functions that have been found useful to do little odds and ends such as plotting the results of K-means clustering, substituting special text characters, viewing parts of a data.frame, constructing formulas from text and building design and response matrices.

License BSD_3_clause + file LICENSE

Depends ggplot2

Imports plyr, dplyr (>= 0.5.0), magrittr, purrr (>= 0.1.0), stats, scales, utils, Matrix, assertthat

LazyLoad yes

ByteCompile TRUE

Suggests testthat, tibble, covr

URL https://github.com/jaredlander/useful

BugReports https://github.com/jaredlander/useful/issues

RoxygenNote 6.1.0

NeedsCompilation no

Author Jared P. Lander [cre, aut],
Nicholas Galasinao [ctb],
Joshua Kraut [ctb],
Daniel Chen [ctb]

Maintainer Jared P. Lander <packages@jaredlander.com>

Repository CRAN

Date/Publication 2018-10-08 16:00:03 UTC

R topics documented:

`autoplot.acf` *autoplot.acf*

Description

Plot acf objects

Usage

```
## S3 method for class 'acf'
autoplot(object, xlab = x, ylab = y,
  title = sprintf("%s Plot", y), ...)
```

Arguments

`object`	An `acf` object.
`xlab`	X-axis label.
`ylab`	y-axis label.
`title`	Graph title.
`...`	Further arguments.

Details

Plot acf (and pacf) objects.

Value

A ggplot object.

Author(s)

Jared P. Lander

Examples

```
autoplot(acf(sunspot.year, plot=FALSE))
autoplot(pacf(sunspot.year, plot=FALSE))
```

`binary.flip` *binary.flip*

Description

Flip binary numbers

Usage

```
binary.flip(x)
```

Arguments

x	A vector of 0/1 numbers.

Value

X with 0's flipped to 1's and 1's flipped to 0's

Author(s)

Jared P. Lander

Examples

```
binary.flip(c(1,1,0,1,0,0,1))
```

`bottomleft` *Grabs the bottom left corner of a data set*

Description

Display the bottom left corner of a rectangular data set

Usage

```
bottomleft(x, r = 5L, c = 5L, ...)
```

Arguments

x	The data
r	Number of rows to display
c	Number of columns to show
...	Arguments passed on to other functions

Details

Displays the bottom left corner of a rectangular data set.

This is a wrapper function for corner

Value

... The bottom left corner of the data set that was requested. The size depends on r and c.

Author(s)

Jared P. Lander www.jaredlander.com

See Also

head tail corner topright topleft bottomright left right

Examples

```
data(diamonds)
head(diamonds)      # displays all columns
bottomleft(diamonds)    # displays last 5 rows and only the first 5 columns
```

---

bottomright *Grabs the bottom right corner of a data set*

Description

Display the bottom right corner of a rectangular data set

Usage

```
bottomright(x, r = 5L, c = 5L, ...)
```

Arguments

x	The data
r	Number of rows to display
c	Number of columns to show
...	Arguments passed on to other functions

Details

Displays the bottom right corner of a rectangular data set.

This is a wrapper function for corner

Value

... The bottom right corner of the data set that was requested. The size depends on r and c.

Author(s)

Jared P. Lander www.jaredlander.com

See Also

head tail corner topright bottomleft topleft left right

Examples

```
data(diamonds)
head(diamonds)      # displays all columns
bottomright(diamonds)    # displays last 5 rows and only the last 5 columns
```

---

build.formula *Formula Builder*

Description

Formula Builder

Usage

```
build.formula(lhs, rhs)
```

Arguments

lhs	Character vector for left side of formula
rhs	Character vector for right side of formula

Details

Builds a formula easily given the left and right hand sides. Right now it only handles additive formulas and not interactions unless that is specified in the character.

Value

A formula object

Author(s)

Jared P. Lander www.jaredlander.com

See Also

formula as.formula

Examples

```
build.formula("Y", "X")
build.formula(c("Y", "Z"), "X")
build.formula("Z", c("X", "Q"))
build.formula(c("Y", "Z"), c("X", "Q"))
```

---

build.x *build.x*

Description

Build the x matrix for a glmnet model

Usage

```
build.x(formula, data, contrasts = TRUE, sparse = FALSE)
```

Arguments

formula	A formula
data	A data.frame
contrasts	Logical indicating whether a factor's base level is removed. Can be either one single value applied to every factor or a value for each factor. Values will be recycled if necessary.
sparse	Logical indicating if result should be sparse.

Details

Given a formula and a data.frame build the predictor matrix

Value

A matrix of the predictor variables specified in the formula

Author(s)

Jared P. Lander

Examples

```
require(ggplot2)
head(mpg)
head(build.x(hwy ~ class + cyl + year, data=mpg))

testFrame <- data.frame(First=sample(1:10, 20, replace=TRUE),
Second=sample(1:20, 20, replace=TRUE),
Third=sample(1:10, 20, replace=TRUE),
Fourth=factor(rep(c("Alice","Bob","Charlie","David"), 5)),
Fifth=ordered(rep(c("Edward","Frank","Georgia","Hank","Isaac"), 4)),
Sixth=factor(rep(c("a", "b"), 10)), stringsAsFactors=F)
head(build.x(First ~ Second + Fourth + Sixth, testFrame,
contrasts=c("Fourth"=TRUE, "Fifth"=FALSE, "Sixth"=TRUE)))
head(build.x(First ~ Second + Fourth + Fifth + Sixth, testFrame,
contrasts=c(Fourth=TRUE, Fifth=FALSE, Sixth=TRUE)))
head(build.x(First ~ Second + Fourth + Fifth + Sixth, testFrame, contrasts=TRUE))
head(build.x(First ~ Second + Fourth + Fifth + Sixth, testFrame,
contrasts=FALSE))
head(build.x(First ~ Second + Fourth + Fifth + Sixth - 1, testFrame,
contrasts=TRUE))
build.x(First ~ Second + Fourth + Fifth + Sixth - 1, testFrame,
contrasts=TRUE, sparse=TRUE)
head(build.x(First ~ Second + Fourth + Fifth + Fourth*Sixth, testFrame, contrasts=TRUE))
head(build.x(First ~ Second + Fourth + Fifth + Third*Sixth, testFrame, contrasts=TRUE))
#' head(build.x(First ~ Second + Fourth + Fifth + Fourth*Sixth, testFrame, contrasts=FALSE))
head(build.x(First ~ Second + Fourth + Fifth + Third*Sixth, testFrame, contrasts=FALSE))
build.x(First ~ Second + Fourth + Fifth + Third*Sixth, testFrame, contrasts=FALSE, sparse=TRUE)

## if contrasts is a list then you can specify just certain factors
```

`build.y` *build.y*

Description

Build the y object from a formula and data

Usage

```
build.y(formula, data)
```

Arguments

`formula`	A formula
`data`	A data.frame

Details

Given a formula and a data.frame build the y object

Value

The y object from a formula and data

Author(s)

Jared P. Lander

Examples

```
require(ggplot2)
head(mpg)
head(build.y(hwy ~ class + cyl + year, data=mpg))
```

---

cart2pol *cart2pol*

Description

Converts polar coordinates to cartesian coordinates

Usage

```
cart2pol(x, y, degrees = FALSE)
```

Arguments

x	The x-coordinate of the point
y	The y-coordinate of the point
degrees	Logical indicating if theta should be returned in degrees

Details

Converts polar coordinates to cartesian coordinates using a simple conversion. The angle, theta must be in radians.

Somewhat inspired by http://www.r-bloggers.com/convert-polar-coordinates-to-cartesian/ and https://www.mathsisfun.com/p
cartesian-coordinates.html

Value

A data.frame holding the polar coordinates and the original (x,y) coordinates

Author(s)

Jared P. Lander

Examples

```
library(dplyr)
x1 <- c(1, sqrt(3)/2, sqrt(2)/2, 1/2, 0)
y1 <- c(0, 1/2, sqrt(2)/2, sqrt(3)/2, 1)
d1 <- data_frame(x=x1, y=y1, Q='I')

x2 <- c(0, -1/2, -sqrt(2)/2, -sqrt(3)/2, -1)
y2 <- c(1, sqrt(3)/2, sqrt(2)/2, 1/2, 0)
d2 <- data_frame(x=x2, y=y2, Q='II')

x3 <- c(-1, -sqrt(3)/2, -sqrt(2)/2, -1/2, 0)
y3 <- c(0, -1/2, -sqrt(2)/2, -sqrt(3)/2, -1)
d3 <- data_frame(x=x3, y=y3, Q='III')

x4 <- c(0, 1/2, sqrt(2)/2, sqrt(3)/2, 1)
y4 <- c(-1, -sqrt(3)/2, -sqrt(2)/2, -1/2, 0)
d4 <- data_frame(x=x4, y=y4, Q='IV')

dAll <- bind_rows(d1, d2, d3, d4)

cart2pol(dAll$x, dAll$y)
cart2pol(dAll$x, dAll$y, degrees=TRUE)
```

`classdf` *classdf*

Description

Get class information for each column in a `data.frame`.

Usage

```
classdf(data, cols)
```

Arguments

`data`	`link{data.frame}` that is to be inspected.
`cols`	The columns (named or numeric) to be included in the check.

Details

Get class information for each column in a `data.frame`.

Value

A vector detailing the class of each column.

Author(s)

Jared P. Lander

Examples

```
classdf(CO2)
classdf(iris)
classdf(mtcars)
```

`colsToFront` *colsToFront*

Description

Moves column names to the front or back of the names

Usage

```
colsToFront(data, cols = names(data))

colsToBack(data, cols = names(data))
```

Arguments

`data`	data.frame or tbl
`cols`	Columns that should be moved

Details

Moves column names to the front or back of the names

Value

Character vector of column names

Author(s)

Jared P. Lander

Examples

```
theDF <- data.frame(A=1:10, B=11:20, C=1:10, D=11:20)
colsToFront(theDF, c('B', 'C'))
colsToFront(theDF, c('C', 'B'))
colsToFront(theDF, c('C', 'C'))
colsToBack(theDF, c('C', 'C'))
colsToBack(theDF, c('C', 'B'))
colsToBack(theDF, c('C', 'C'))
```

compare.list *List Comparison*

Description

List Comparison

Usage

```
compare.list(a, b)
```

Arguments

a	A List
b	A List

Details

Compare elements of two equal length lists.

Value

A vector with a logical indicator for equality of each element author Jared P. Lander www.jaredlander.com

Examples

```
vect <- c(mean, mode, mean)
vect2 <- c(mean, mode, max)
vect3 <- c(mean, mean)
compare.list(vect, vect)
compare.list(vect, vect2)
tryCatch(compare.list(vect, vect3), error=function(e) print("Caught error"))
```

ComputeHartigan *Compute Hartigan's Number*

Description

Runs the computation found in http://www.stat.columbia.edu/~madigan/DM08/descriptive.ppt.pdf

Usage

```
ComputeHartigan(FitActualWSS, FitPlus1WSS, nrow)
```

Arguments

FitActualWSS	the WSS from a kmeans fit
FitPlus1WSS	the WSS from a kmeans fit
nrow	the number of rows in the original dataset

Details

Not exported, only used by FitKMeans

Value

The computed Hartigan Number

Author(s)

Jared P. Lander www.jaredlander.com

References

http://www.stat.columbia.edu/~madigan/DM08/descriptive.ppt.pdf

See Also

kmeans FitKMeans

Examples

```
data(iris)
hartiganResults <- FitKMeans(iris[, -ncol(iris)])
PlotHartigan(hartiganResults)
```

---

constant *constant*

Description

Helper function for imputing constants

Usage

```
constant(n = 1)
```

Arguments

n	The value to return

Details

Returns a function that always returns the value of n.

Value

A function that when used simply returns n.

Author(s)

Jared P. Lander

Examples

```
constant(4)(1:10)

theDF <- data.frame(A=1:10, B=1:10, C=1:10)
theDF[c(1, 4, 6), c(1)] <- NA
theDF[c(3, 4, 8), c(3)] <- NA
simple.impute(theDF, constant(4))
```

---

corner *corner*

Description

Display a corner section of a rectangular data set

Usage

```
corner(x, ...)

## S3 method for class 'data.frame'
corner(x, r = 5L, c = 5L, corner = "topleft",
  ...)

## S3 method for class 'matrix'
corner(x, r = 5L, c = 5L, corner = "topleft", ...)

## S3 method for class 'table'
corner(x, r = 5L, c = 5L, corner = "topleft", ...)

## Default S3 method:
corner(x, r = 5L, ...)
```

Arguments

x	The data
...	Arguments passed on to other functions
r	Number of rows to display
c	Number of columns to show
corner	Which corner to grab. Possible values are c("topleft", "bottomleft", "topright", "bottomright")

Details

Grabs a corner of a data set

Display a corner section of a rectangular data set

Displays a corner of a rectangular data set such as a data.frame, matrix or table. If showing the right side or bottom, the order of the data is preserved.

The default method reverts to simply calling head

corner of a rectangular data set such as a data.frame, matrix or table. If showing the right side or bottom, the order of the data is preserved.

Value

... The part of the data set that was requested. The size depends on r and c and the position depends on corner.

Author(s)

Jared P. Lander

See Also

head tail topleft topright bottomleft bottomright left right

Examples

```
data(diamonds)
head(diamonds)      # displays all columns
corner(diamonds)    # displays first 5 rows and only the first 5 columns
corner(diamonds, corner="bottomleft")     # displays the last 5 rows and the first 5 columns
corner(diamonds, corner="topright")     # displays the first 5 rows and the last 5 columns
```

find.case *find.case*

Description

Checks if strings are all upper or all lower case

Usage

```
find.case(string, case = c("upper", "lower", "mixed", "numeric"))
```

Arguments

string	Character vector of strings to check cases
case	Whether checking for upper or lower case

Details

Checks if strings are all upper or all lower case. If string is all numbers it returns TRUE.

Value

A vector of TRUE AND FALSE

Author(s)

Jared P. Lander

See Also

upper.case lower.case numeric.case mixed.case

Examples

```
toCheck <- c('BIG', 'little', 'Mixed', 'BIG WITH SPACE', 'little with space', 'MIXED with SPACE')
find.case(toCheck, 'upper')
find.case(toCheck, 'lower')
```

`FitKMeans` *Fit a series of kmeans clusterings and compute Hartigan's Number*

Description

Given a numeric dataset this function fits a series of kmeans clusterings with increasing number of centers. k-means is compared to k+1-means using Hartigan's Number to determine if the k+1st cluster should be added.

Usage

```
FitKMeans(x, max.clusters = 12L, spectral = FALSE, nstart = 1L,
  iter.max = 10L, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
  "MacQueen"), seed = NULL)
```

Arguments

`x`	The data, numeric, either a matrix or data.frame
`max.clusters`	The maximum number of clusters that should be tried
`spectral`	logical; If the data being fit are eigenvectors for spectral clustering
`nstart`	The number of random starts for the kmeans algorithm to use
`iter.max`	Maximum number of tries before the kmeans algorithm gives up on conversion
`algorithm`	The desired algorithm to be used for kmeans. Options are c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"). See `kmeans`
`seed`	If not null, the random seed will be reset before each application of the kmeans algorithm

Details

A consecutive series of kmeans is computed with increasing k (number of centers). Each result for k and k+1 are compared using Hartigan's Number. If the number is greater than 10, it is noted that having k+1 clusters is of value.

Value

A data.frame consisting of columns, for the number of clusters, the Hartigan Number and whether that cluster should be added, based on Hartigan's Number.

Author(s)

Jared P. Lander www.jaredlander.com

References

http://www.stat.columbia.edu/~madigan/DM08/descriptive.ppt.pdf

See Also

kmeans PlotHartigan

Examples

```
data(iris)
hartiganResults <- FitKMeans(iris[, -ncol(iris)])
PlotHartigan(hartiganResults)
```

ForceDataFrame *ForceDataFrame*

Description

Force matrix and arrays to data.frame

Usage

```
ForceDataFrame(data)
```

Arguments

data	matrix, data.frame, array, list, etc. . .

Details

This is a helper function for build.x and build.y to convert arrays and matrices–which are not accepted in model.frame–into data.frames

Value

a data.frame of the data

Author(s)

Jared P. Lander

`fortify.acf` *fortify.acf*

Description

Fortify an acf/pacf object

Usage

```
## S3 method for class 'acf'
fortify(model, data = NULL, ...)
```

Arguments

`model`	An `acf` object.
`data`	Not used. Just for consistency with the fortify method.
`...`	Other arguments

Details

Prepares acf (and pacf) objects for plotting with ggplot.

Value

`data.frame` for plotting with ggplot.

Author(s)

Jared P. Lander

Examples

```
fortify(acf(sunspot.year, plot=FALSE))
fortify(pacf(sunspot.year, plot=FALSE))
```

`fortify.kmeans` *fortify.kmeans*

Description

Fortify a kmeans model with its data

Usage

```
## S3 method for class 'kmeans'
fortify(model, data = NULL, ...)
```

Arguments

`model`	`kmeans` model
`data`	Data used to fit the model
`...`	Not Used

Details

Prepares a kmeans object to be plotted using `cmdscale` to compute the projected x/y coordinates. If `data` is not provided, then just the center points are calculated.

Value

The original data with extra columns:

`.x`	The projected x position.
`.y`	The projected y position.
`.Cluster`	The cluster that point belongs to.

Author(s)

Jared P. Lander

See Also

kmeans fortify ggplot plot.kmeans

Examples

```
k1 <- kmeans(x=iris[, 1:4], centers=3)
hold <- fortify(k1, data=iris)
head(hold)
hold2 <- fortify(k1)
head(hold2)
```

fortify.ts *fortify.ts*

Description

Fortify a ts object.

Usage

```
## S3 method for class 'ts'
fortify(model, data = NULL, name = as.character(m[[2]]),
  ...)
```

Arguments

model	A ts object.
data	A vector of the same length of x that specifies the time component of each element of x.
name	Character specifying the name of x if it is to be different than the variable being inputed.
...	Further arguments.

Details

Prepares a ts object for plotting with ggplot.

Value

data.frame for plotting with ggplot.

Author(s)

Jared P. Lander

Examples

```
fortify(sunspot.year)
```

indexToPosition *indexToPosition*

Description

Given a long matrix index convert to row and column positions

Usage

```
indexToPosition(x, nrow = 1)
```

Arguments

x	Position of indices
nrow	The number of rows in the matrix

Details

Using which on a matrix returns a number that iterates down rows then across columns. This function returns the (row, column) position of that index.

Value

A Matrix with row and column columns and a row for each value of x

Author(s)

Jared P. Lander

Examples

```
indexToPosition(3, 2)
indexToPosition(c(1, 4, 5, 7, 9), 3)
indexToPosition(1:16, 4)
indexToPosition(c(1, 3, 5, 6, 8, 10, 11, 13, 15), 5)
```

interval.check *interval.check*

Description

Check which interval a number belongs to

Usage

```
interval.check(data, input = "Stop", times, fun = "<=")
```

Arguments

data	data.frame
input	character name of column we wish to compare
times	vector in ascending order where the differences between sequential elements are the intervals
fun	character containing comparator

Details

This function takes in a data.frame with a specified column and compares that to a vector of times

Value

Vector indicating which element of `times` that row belongs to. If the row is beyond any element NA is in it's spot.

Author(s)

Jared P. Lander

Examples

```
head(cars)
interval.check(cars, input="speed", times=seq(min(cars$speed), max(cars$speed), length=10))
```

`left` *Grabs the left side of a data set*

Description

Display the left side of a rectangular data set

Usage

```
left(x, c = 5L, ...)
```

Arguments

`x`	The data
`c`	Number of columns to show
`...`	Arguments passed on to other functions

Details

Displays the left side of a rectangular data set.

This is a wrapper function for `corner`

Value

... The left side of the data set that was requested. The size depends on c.

Author(s)

Jared P. Lander www.jaredlander.com

See Also

`head` `tail` `corner` `topright` `bottomleft` `bottomright` `topleft` `right`

Examples

```
data(diamonds)
head(diamonds)      # displays all columns
left(diamonds)    # displays all rows and only the first 5 columns
```

load_packages *load_packages*

Description

Loads multiple packages

Usage

```
load_packages(packages)
```

Arguments

packages	A ‘character‘ vector of packages to be installed

Details

Allows the user to load multiple packages with one line of code. Delivers a message saying which packages have been loaded. If a user requests packages that are not installed there is an error.

Value

Nothing, loads packages

Author(s)

Jared P. Lander

Examples

```
load_packages('ggplot2')
load_packages(c('ggplot2', 'dplyr'))
```

lower.case *lower.case*

Description

Checks if strings are all lower case

Usage

```
lower.case(string)
```

Arguments

string	Character vector of strings to check cases

Details

Checks if strings are all lower case. This is a wrapper for `find.case('text', 'lower')`. If string is all numbers it returns TRUE.

Value

A vector of TRUE AND FALSE

Author(s)

Jared P. Lander

See Also

find.case upper.case mixed.case numeric.case

Examples

```
toCheck <- c('BIG', 'little', 'Mixed', 'BIG WITH SPACE', 'little with space', 'MIXED with SPACE')
lower.case(toCheck)
```

MapToInterval *Map numbers to interval*

Description

Maps a range of numbers to a given interval

Usage

```
MapToInterval(nums, start = 1, stop = 10)
```

Arguments

nums	The vector of numbers to be mapped
start	The start of the interval
stop	The end of the interval

Details

formula: a + (x - min(x)) * (b - a) / (max(x) - min(x))

Value

The original numbers mapped to the given interval

Author(s)

Jared P. Lander www.jaredlander.com

See Also

mapping

Examples

```
MapToInterval(1:10, start=0, stop=1)
mapping(1:10, start=0, stop=1)
```

`mixed.case` *mixed.case*

Description

Checks if strings are all lower case

Usage

```
mixed.case(string)
```

Arguments

`string`	Character vector of strings to check cases

Details

Checks if strings are a mix of upper and lower case. This is a wrapper for `find.case('text', 'mixed')`. If string is all numbers it returns FALSE.

Value

A vector of TRUE AND FALSE

Author(s)

Jared P. Lander

See Also

find.case all.upper

Examples

```
toCheck <- c('BIG', 'little', 'Mixed', 'BIG WITH SPACE', 'little with space', 'MIXED with SPACE')
mixed.case(toCheck)
```

`moveToFront` *moveToFront*

Description

Rearranges column order by moving specified columns to the front or back.

Usage

```
moveToFront(data, cols)

moveToBack(data, cols)
```

Arguments

`data`	data.frame
`cols`	Character vector specifying the columns to be moved to the front or back

Details

Rearranges column order by moving specified columns to the front or back.

Value

A data.frame with the columns in the right order

Author(s)

Jared P. Lander

Examples

```
theDF <- data.frame(A=1:10, B=11:20, C=1:10, D=11:20)
moveToFront(theDF, c('B', 'C'))
moveToFront(theDF, c('C', 'B'))
moveToFront(theDF, c('C', 'C'))
moveToBack(theDF, c('C', 'C'))
moveToBack(theDF, c('C', 'B'))
moveToBack(theDF, c('C', 'C'))
```

multiple *multiple*

Description

Order of Magnitude Formatter

Usage

```
multiple(x, multiple = c("K", "M", "B", "T", "H", "k", "m", "b", "t",
  "h"), big.mark = ",", extra, digits = 0, prefix = "",
  scientific = FALSE)
```

Arguments

x	Vector of numbers to be formatted.
multiple	The multiple to display numbers in. This symbol will be added to the end of the numbers.
big.mark	Character specifying the thousands separator
extra	DEPRECATED, use big.mark and prefix instead: Function for perform any further formatting.
digits	Number of decimal places for rounding.
prefix	Symbol to put in front of the numbers such as a dollar sign.
scientific	Logical (default: FALSE) indicating if the numbers should be returned in scientific notation.

Details

This divides the number by the appropriate amount and adds on the corresponding symbol at the end of the number.

Value

Character vector of formatted numbers.

Author(s)

Jared P. Lander

Examples

```
require(scales)
vect <- c(1000, 1500, 23450, 21784, 875003780)
multiple(vect)
multiple(vect, extra=dollar)
```

```
multiple(vect, extra=identity)

require(ggplot2)
data(diamonds)
ggplot(diamonds, aes(x=x, y=y, color=price*100)) + geom_point() +
scale_color_gradient2(labels=multiple)
```

`multiple.comma` *multiple.comma*

Description

Order of Magnitude Formatter

Usage

```
multiple.comma(x, ...)
```

Arguments

`x`	Vector of numbers to be formatted.
`...`	Further arguments to be passed on to `link{multiple}`

Details

Simply a wrapper for multiple that prespecifies the extra comma.

Value

Character vector of comma formatted numbers.

Author(s)

Jared P. Lander

Examples

```
require(scales)
vect <- c(1000, 1500, 23450, 21784, 875003780)
multiple.comma(vect)
multiple.comma(vect, multiple="k")
multiple.comma(vect, multiple="h")

require(ggplot2)
data(diamonds)
ggplot(diamonds, aes(x=x, y=y, color=price*100)) + geom_point() +
scale_color_gradient2(labels=multiple.comma)
```

`multiple.dollar` *multiple.dollar*

Description

Order of Magnitude Formatter

Usage

```
multiple.dollar(x, ...)
```

Arguments

`x`	Vector of numbers to be formatted.
`...`	Further arguments to be passed on to `multiple`

Details

Simply a wrapper for multiple that prespecifies the extra dollar.

Value

Character vector of dollar formatted numbers.

Author(s)

Jared P. Lander

Examples

```
require(scales)
vect <- c(1000, 1500, 23450, 21784, 875003780)
multiple.dollar(vect)
multiple.dollar(vect, multiple="k")
multiple.dollar(vect, multiple="h")

require(ggplot2)
data(diamonds)
ggplot(diamonds, aes(x=x, y=y, color=price*100)) + geom_point() +
scale_color_gradient2(labels=multiple.dollar)
```

multiple.identity *multiple.identity*

Description

Order of Magnitude Formatter

Usage

```
multiple.identity(x, ...)
```

Arguments

x	Vector of numbers to be formatted.
...	Further arguments to be passed on to link{multiple}

Details

Simply a wrapper for multiple that prespecifies the extra identity.

Value

Character vector of formatted numbers.

Author(s)

Jared P. Lander

Examples

```
vect <- c(1000, 1500, 23450, 21784, 875003780)
multiple.identity(vect)
multiple.identity(vect, multiple="k")
multiple.identity(vect, multiple="h")

require(ggplot2)
data(diamonds)
ggplot(diamonds, aes(x=x, y=y, color=price*100)) + geom_point() +
scale_color_gradient2(labels=multiple.identity)
```

multiple_format *multiple_format*

Description

Multiple Style Formatting

Usage

```
multiple_format(...)
```

Arguments

...	Arguments to be passed onto multiple

Details

Since ggplot requires a function for formatting this allows the user to specify the function's arguments, which will return a function that can be used by ggplot.

Value

The function multiple.

Author(s)

Jared P. Lander

Examples

```
library(scales)
vect <- c(1000, 1500, 23450, 21784, 875003780)
multiple_format()(vect)
multiple_format(extra=dollar)(vect)
multiple_format(extra=identity)(vect)

require(ggplot2)
data(diamonds)
ggplot(diamonds, aes(x=x, y=y, color=price*100)) + geom_point() +
scale_color_gradient2(labels=multiple_format(extra=dollar))
```

numeric.case *numeric.case*

Description

Checks if strings are all numbers or spaces

Usage

```
numeric.case(string)
```

Arguments

string	Character vector of strings to check cases

Details

Checks if strings are all numbers and spaces. This is a wrapper for `find.case('text', 'numeric')`.

Value

A vector of TRUE AND FALSE

Author(s)

Jared P. Lander

See Also

find.case upper.case lower.case numeric.case

Examples

```
toCheck <- c('BIG', 'little', 'Mixed', 'BIG WITH SPACE',
     'little with space', 'MIXED with SPACE', '17')
numeric.case(toCheck)
```

plot.acf *plot.acf*

Description

Plotting an ACF object

Usage

```
## S3 method for class 'acf'
plot(x, ...)
```

Arguments

x	An ACF object
...	Arguments passed on to autoplot

Details

This function has been deprecated in favor of autoplot

Value

A ggplot2 object

Author(s)

Jared P. Lander

plot.kmeans *plot.kmeans*

Description

Plot the results from a k-means object

Usage

```
## S3 method for class 'kmeans'
plot(x, data = NULL, class = NULL, size = 2,
  legend.position = c("right", "bottom", "left", "top", "none"),
  title = "K-Means Results", xlab = "Principal Component 1",
  ylab = "Principal Component 2", ...)
```

Arguments

x	A kmeans object.
data	The data used to kit the kmeans object.
class	Character name of the "true" classes of the data.
size	Numeric size of points
legend.position	Character indicating where the legend should be placed.
title	Title for the plot.
xlab	Label for the x-axis.
ylab	Label for the y-axis.
...	Not Used.

Details

Plots the results of k-means with color-coding for the cluster membership. If data is not provided, then just the center points are calculated.

Value

A ggplot object

Author(s)

Jared P. Lander

See Also

kmeans fortify ggplot plot.kmeans

Examples

```
k1 <- kmeans(x=iris[, 1:4], centers=3)
plot(k1)
plot(k1, data=iris)
```

PlotHartigan *Plot a series of Hartigan's Numbers*

Description

After fitting a series of Hartigan's Numbers (see FitKMeans) this will plot the results so it is easy to visualize

Usage

```
PlotHartigan(hartigan, title = "Hartigan's Rule", smooth = FALSE,
  linecolor = "grey", linetype = 2L, linesize = 1L, minor = TRUE)
```

Arguments

hartigan	The results from FitKMeans
title	Title to be used in the plot
smooth	logical; if true a smoothed line will be fit to the points, otherwise it will be a piecewise line
linecolor	Color of the horizontal line denoting 10
linetype	Type of the horizontal line denoting 10
linesize	Size of the horizontal line denoting 10
minor	logical; if true minor grid lines will be plotted

Details

Displays a graphical representation of the results of FitKMeans

Value

a ggplot object

Author(s)

Jared P. Lander www.jaredlander.com

References

#' http://www.stat.columbia.edu/~madigan/DM08/descriptive.ppt.pdf

See Also

kmeans FitKMeans

Examples

```
data(iris)
hartiganResults <- FitKMeans(iris[, -ncol(iris)])
PlotHartigan(hartiganResults)
```

`plotTimesSeries` *plotTimesSeries*

Description

Plot ts object

Usage

```
plotTimesSeries(x, time = NULL, acf = FALSE, lag.max = NULL,
  na.action = na.fail, demean = TRUE, title = sprintf("%s Plot",
  name), xlab = "Time", ylab = name, ...)
```

Arguments

`x`	a `ts` object.
`time`	A vector of the same length of `x` that specifies the time component of each element of `x`.
`acf`	Logical indicating if the acf and pacf should be plotted.
`lag.max`	maximum lag at which to calculate the acf. Default is 10*log10(N/m) where N is the number of observations and m the number of series. Will be automatically limited to one less than the number of observations in the series.
`na.action`	function to be called to handle missing values. na.pass can be used.
`demean`	logical. Should the covariances be about the sample means?
`title`	Graph title.
`xlab`	X-axis label.
`ylab`	Y-axis label.
`...`	Further arguments.

Details

Plot a ts object and, if desired, it's acf and pacf.

Value

A ggplot object if `acf` is `FALSE`, otherwise `TRUE` indicating success.

Author(s)

Jared P. Lander

See Also

ts.plotter plot.acf fortify.ts

Examples

```
plot(sunspot.year)
plot(sunspot.year, acf=TRUE)
```

`pol2cart` *pol2cart*

Description

Converts polar coordinates to cartesian coordinates

Usage

```
pol2cart(r, theta, degrees = FALSE)
```

Arguments

`r`	The radius of the point
`theta`	The angle of the point, in radians
`degrees`	Logical indicating if theta is specified in degrees

Details

Converts polar coordinates to cartesian coordinates using a simple conversion. The angle, `theta` must be in radians.

Somewhat inspired by http://www.r-bloggers.com/convert-polar-coordinates-to-cartesian/ and https://www.mathsisfun.com/p
cartesian-coordinates.html

Value

A data.frame holding the (x,y) coordinates and original polar coordinates

Author(s)

Jared P. Lander

Examples

```
polarRadPosTop <- data.frame(r=c(3, 5, 3, 5, 4, 6, 4, 6, 2),
     theta=c(0, pi/6, pi/4, pi/3, pi/2, 2*pi/3, 3*pi/4, 5*pi/6, pi))
polarRadPosBottom <- data.frame(r=c(3, 5, 3, 5, 4, 6, 4, 6, 2),
     theta=c(pi, 7*pi/6, 5*pi/4, 4*pi/3, 3*pi/2, 5*pi/3, 7*pi/4, 9*pi/6, 2*pi))
polarRadNegTop <- data.frame(r=c(3, 5, 3, 5, 4, 6, 4, 6, 2),
     theta=-1*c(0, pi/6, pi/4, pi/3, pi/2, 2*pi/3, 3*pi/4, 5*pi/6, pi))
polarRadNegBottom <- data.frame(r=c(3, 5, 3, 5, 4, 6, 4, 6, 2),
     theta=-1*c(pi, 7*pi/6, 5*pi/4, 4*pi/3, 3*pi/2, 5*pi/3, 7*pi/4, 9*pi/6, 2*pi))

pol2cart(polarRadPosTop$r, polarRadPosTop$theta)
pol2cart(polarRadPosBottom$r, polarRadPosBottom$theta)
pol2cart(polarRadNegTop$r, polarRadNegTop$theta)
pol2cart(polarRadNegBottom$r, polarRadNegBottom$theta)
```

`positionToIndex` *positionToIndex*

Description

Given row and column positions calculate the index.

Usage

```
positionToIndex(row, col, nrow = max(row))
```

Arguments

`row`	Vector specifying row positions
`col`	Vector specifying column positions
`nrow`	The number of rows in the matrix

Details

With row and column positions this computes the index, starting at (1,1) working down rows then across columns.

Value

A vector of indices

Author(s)

Jared P. Lander

Examples

```
positionToIndex(1, 2, 2)
positionToIndex(row=c(1, 1, 2, 1, 3), col=c(1, 2, 2, 3, 3), nrow=3)
positionToIndex(rep(1:4, 4), rep(1:4, each=4), nrow=4)
positionToIndex(rep(c(1, 3, 5), 3), rep(1:3, each=3), nrow=5)
```

---

reclass *reclass*

Description

Adds a class to an x.

Usage

```
reclass(x, value)

reclass(x) <- value
```

Arguments

x	The x getting the new class
value	The new class

Details

Adds a class to an x by putting the new class at the front of the vector of classes for the x.

Value

The original x with the class containing `value` in addition to the previous class(es)

Author(s)

Jared P. Lander

Examples

```
theDF <- data.frame(A=1:10, B=1:10)
reclass(theDF) <- 'newclass'
class(theDF)
theDF <- reclass(theDF, 'another')
class(theDF)
```

`right` *Grabs the right side of a data set*

Description

Display the right side of a rectangular data set

Usage

```
right(x, c = 5L, ...)
```

Arguments

`x`	The data
`c`	Number of columns to show
`...`	Arguments passed on to other functions

Details

Displays the right side of a rectangular data set.

This is a wrapper function for `corner`

Value

... The left side of the data set that was requested. The size depends on c.

Author(s)

Jared P. Lander www.jaredlander.com

See Also

`head` `tail` `corner` `topright` `bottomleft` `bottomright` `topleft` `topleft`

Examples

```
data(diamonds)
head(diamonds)      # displays all columns
right(diamonds)    # displays all rows and only the last 5 columns
```

shift.column *shift.column*

Description

Shift a column of data

Usage

```
shift.column(data, columns, newNames = sprintf("%s.Shifted", columns),
  len = 1L, up = TRUE)
```

Arguments

data	data.frame
columns	Character vector specifying which columns to shift.
newNames	Character vector specifying new names for the columns that will be created by the shift. Must be same length as columns.
len	Integer specifying how many rows to shift the data.
up	logical indicating if rows should be shifted up or down.

Details

Shifts a column of data up or down a certain number of rows

Value

data.frame with the specified columns shifted.

Author(s)

Jared P. Lander

Examples

```
myData <- data.frame(Upper=LETTERS, lower=letters)
shift.column(data=myData, columns="lower")
shift.column(data=myData, columns="lower", len=2)
```

simple.impute *simple.impute*

Description

Generic function for simple imputation.

Usage

```
simple.impute(x, fun = median, ...)
```

Arguments

x	An object to be imputed
fun	The function with which to fill in missing values
...	Further arguments

Details

Provides the ability to simply impute data based on a simple measure such as mean or median. For more robust imputation see the packages Amelia, mice or mi.

Value

An object with the missing values imputed.

Author(s)

Jared P. Lander

Examples

```
theDF <- data.frame(A=1:10, B=1:10, C=1:10)
theDF[c(1, 4, 6), c(1)] <- NA
theDF[c(3, 4, 8), c(3)] <- NA

simple.impute(theDF$A)
simple.impute(theDF$A, mean)
simple.impute(theDF$A, constant(4))
simple.impute(theDF)
simple.impute(theDF, mean)
simple.impute(theDF, constant(4))
```

simple.impute.data.frame *simple.impute.data.frame*

Description

Function for imputing a data.frame with missing data.

Usage

```
## S3 method for class 'data.frame'
simple.impute(x, fun = stats::median, ...)
```

Arguments

x	A data.frame
fun	The function with which to fill in missing values
...	Further arguments

Details

Provides the ability to simply impute data based on a simple measure such as mean or median. For more robust imputation see the packages Amelia, mice or mi.

Each column is imputed independently.

Value

A data.frame with the missing values imputed.

Author(s)

Jared P. Lander

Examples

```
theDF <- data.frame(A=1:10, B=1:10, C=1:10)
theDF[c(1, 4, 6), c(1)] <- NA
theDF[c(3, 4, 8), c(3)] <- NA

simple.impute.data.frame(theDF)
simple.impute.data.frame(theDF, mean)
simple.impute.data.frame(theDF, constant(4))
```

`simple.impute.default` *simple.impute.default*

Description

Function for imputing a vector with missing data.

Usage

```
## Default S3 method:
simple.impute(x, fun = median, ...)
```

Arguments

`x`	A numeric or integer vector
`fun`	The function with which to fill in missing values
`...`	Further arguments

Details

Provides the ability to simply impute data based on a simple measure such as mean or median. For more robust imputation see the packages Amelia, mice or mi.

Value

An object with the missing values imputed.

Author(s)

Jared P. Lander

Examples

```
theDF <- data.frame(A=1:10, B=1:10, C=1:10)
theDF[c(1, 4, 6), c(1)] <- NA
theDF[c(3, 4, 8), c(3)] <- NA

simple.impute.default(theDF$A)
simple.impute.default(theDF$A, mean)
simple.impute.default(theDF$A, constant(4))
```

simple.impute.tbl_df *simple.impute.tbl_df*

Description

Function for imputing a tbl_df with missing data.

Usage

```
## S3 method for class 'tbl_df'
simple.impute(x, fun = median, ...)
```

Arguments

x	A data.frame
fun	The function with which to fill in missing values
...	Further arguments

Details

Provides the ability to simply impute data based on a simple measure such as mean or median. For more robust imputation see the packages Amelia, mice or mi.

Each column is imputed independently.

Value

A data.frame with the missing values imputed.

Author(s)

Jared P. Lander

Examples

```
theDF <- data.frame(A=1:10, B=1:10, C=1:10)
theDF[c(1, 4, 6), c(1)] <- NA
theDF[c(3, 4, 8), c(3)] <- NA

simple.impute.data.frame(theDF)
simple.impute.data.frame(theDF, mean)
simple.impute.data.frame(theDF, constant(4))
```

subOut *Sub special characters out of a character vector.*

Description

Converts each of the special characters to their escaped equivalents in each element of a single vector.

Usage

```
subOut(toAlter, specialChars = c("!", "(", ")", "-", "=", "*", "."))
```

Arguments

toAlter	Character vector that will be altered by subbing the special characters with their escaped equivalents
specialChars	The characters to be subbed out

Details

Each element in the specialChar vector is subbed for its escaped equivalent in each of the elements of toAlter

Value

toAlter is returned with any of the defined specialChars subbed out for their escaped equivalents

Author(s)

Jared P. Lander www.jaredlander.com

See Also

sub subSpecials

Examples

```
subOut(c("Hello", "(parens)", "Excited! Mark"))
subOut(c("Hello", "(parens)", "Excited! Mark"), specialChars=c("!", "("))
```

subSpecials *Sub special characters out of character vectors.*

Description

Converts each of the special characters to their escaped equivalents in each element of each vector.

Usage

```
subSpecials(..., specialChars = c("!", "(", ")", "-", "=", "*", "."))
```

Arguments

...	Character vectors that will be altered by subbing the special characters with their escaped equivalents
specialChars	The characters to be subbed out

Details

Each element in the specialChar vector is subbed for its escaped equivalent in each of the elements of each vector passed in

Value

The provided vectors are returned with any of the defined specialChars subbed out for their escaped equivalents. Each vector is returned as an element of a list.

Author(s)

Jared P. Lander www.jaredlander.com

See Also

sub subOut

Examples

```
subSpecials(c("Hello", "(parens)", "Excited! Mark"))
subSpecials(c("Hello", "(parens)", "Excited! Mark"), specialChars=c("!", "("))
subSpecials(c("Hello", "(parens)", "Excited! Mark"),
 c("This is a period. And this is an asterisk *"), specialChars=c("!", "("))
subSpecials(c("Hello", "(parens)", "Excited! Mark"),
 c("This is a period. And this is an asterisk *"), specialChars=c("!", "(", "*"))
```

subVector *subVector*

Description

Substitutes multiple patterns and corresponding replacements

Usage

```
subVector(x, toSub)

subMultiple(x, pattern, replacement)
```

Arguments

x	Vector of text to search
toSub	Named vector where the elements are the pattern and the names are the replacement values
pattern	Vector of patterns to find in each element of x
replacement	Vector of replacement values corresponding to each value of pattern

Details

Given a vector of text replaces all patterns each each element

Value

The text in x with substitutions made

Author(s)

Jared P. Lander

Examples

```
theText <- c('Hi Bob & Cooper how is life today',
'Anything happening now?',
'Sally & Dave are playing with Jess & Julio | with their kids')
subVector(theText, toSub=c("and"='&', 'or'='\\|'))
subVector(theText)

theText <- c('Hi Bob & Cooper how is life today',
'Anything happening now?',
'Sally & Dave are playing with Jess & Julio | with their kids')
subMultiple(theText, pattern=c('&', '\\|'), replacement=c('and', 'or'))
```

timeSingle *timeSingle*

Description

Convenience function that takes in a time object and calculates a difference with a user specified prompt

Usage

```
timeSingle(string = "Time difference", startTime, endTime = Sys.time(),
  sep = ":")
```

Arguments

string	string of what was timed
startTime	"POSIXct" "POSIXt" object, usually from Sys.time
endTime	"POSIXct" "POSIXt" object, usually from Sys.time
sep	string, usually character that is used as the separator between user prompt and time difference

Value

prompt_string string user prompt with time difference

Author(s)

Daniel Y. Chen

Examples

```
x <- 3.14
strt <- Sys.time()
sq <- x ** 2
timeSingle('Squaring value', strt)
```

topleft *Grabs the top left corner of a data set*

Description

Display the top left corner of a rectangular data set

Usage

```
topleft(x, r = 5L, c = 5L, ...)
```

Arguments

x	The data
r	Number of rows to display
c	Number of columns to show
...	Arguments passed on to other functions

Details

Displays the top left corner of a rectangular data set.

This is a wrapper function for corner

Value

... The top left corner of the data set that was requested. The size depends on r and c.

Author(s)

Jared P. Lander www.jaredlander.com

See Also

head tail corner topright bottomleft bottomright left right

Examples

```
data(diamonds)
head(diamonds)      # displays all columns
topleft(diamonds)    # displays first 5 rows and only the first 5 columns
```

topright *Grabs the top right corner of a data set*

Description

Display the top right corner of a rectangular data set

Usage

```
topright(x, r = 5L, c = 5L, ...)
```

Arguments

x	The data
r	Number of rows to display
c	Number of columns to show
...	Arguments passed on to other functions

Details

Displays the top right corner of a rectangular data set.

This is a wrapper function for corner

Value

... The top right corner of the data set that was requested. The size depends on r and c.

Author(s)

Jared P. Lander www.jaredlander.com

See Also

head tail corner topleft bottomleft bottomright left right

Examples

```
data(diamonds)
head(diamonds)      # displays all columns
topright(diamonds)    # displays first 5 rows and only the last 5 columns
```

`ts.plotter` *ts.plotter*

Description

Plot a ts object

Usage

```
ts.plotter(data, time = NULL, title = "Series Plot", xlab = "Time",
  ylab = "Rate")
```

Arguments

`data`	A `ts` object to be plotted.
`time`	A vector of the same length of `data` that specifies the time component of each element of `data`.
`title`	Title of plot.
`xlab`	X-axis label.
`ylab`	Y-axis label.

Details

Fortifies, then plots a `ts` object.

Value

A ggplot object

Author(s)

Jared P. Lander

Examples

```
ts.plotter(sunspot.year)
```

uniqueBidirection *uniqueBidirection*

Description

Find unique rows of a data.frame regardless of the order they appear

Usage

```
uniqueBidirection(x)
```

Arguments

x	a data.frame

Details

Sorts individual rows to get uniques regardless of order of appearance.

Value

A data.frame that is unique regardless of direction

Author(s)

Jared P. Lander

Examples

```
ex <- data.frame(One=c('a', 'c', 'a', 'd', 'd', 'c', 'b'),
Two=c('b', 'd', 'b', 'e', 'c', 'd', 'a'),
stringsAsFactors=FALSE)

# make a bigger version
exBig <- ex
for(i in 1:1000)
{
    exBig <- rbind(exBig, ex)
}

dim(exBig)

uniqueBidirection(ex)
uniqueBidirection(exBig)

ex3 <- dplyr::bind_cols(ex, dplyr::data_frame(Three=rep('a', nrow(ex))))
uniqueBidirection(ex3)
```

upper.case *upper.case*

Description

Checks if strings are all upper case

Usage

```
upper.case(string)
```

Arguments

string	Character vector of strings to check cases

Details

Checks if strings are all upper case. This is a wrapper for `find.case('text', 'upper')`. If string is all numbers it returns TRUE.

Value

A vector of TRUE AND FALSE

Author(s)

Jared P. Lander

See Also

find.case lower.case mixed.case numeric.case

Examples

```
toCheck <- c('BIG', 'little', 'Mixed', 'BIG WITH SPACE', 'little with space', 'MIXED with SPACE')
upper.case(toCheck)
```

useful *Helper functions*

Description

A collection of handy, helper functions

vplayout *vplayout*

Description

Viewport

Usage

```
vplayout(x, y)
```

Arguments

x	The x cell of the viewport to push into.
y	The y cell of the viewport to push into.

Details

Creates viewport for pushing ggplot objects to parts of a console.

Value

An R object of class viewport.

Author(s)

Jared P. Lander

Examples

```
library(ggplot2)
library(grid)
```

WhichCorner *WhichCorner*

Description

Function to build the right row selection depending on the desired corner.

Usage

```
WhichCorner(corner = c("topleft", "bottomleft", "topright",
  "bottomright"), r = 5L, c = 5L, object = "x")
```

Arguments

corner	(character) which corner to display c("topleft", "bottomleft", "topright", "bottomright")
r	(numeric) the number of rows to show
c	(numeric) the number of columns to show
object	The name of the object that is being subsetted

Details

Function to build the right row selection depending on the desired corner. Helper function for getting the indexing for data.frame's, matrices

Value

An expression that is evaluated to return the proper portion of the data

Author(s)

Jared P. Lander

Examples

```
## Not run:
WhichCorner('topleft')
WhichCorner('bottomleft')
WhichCorner('topright')
WhichCorner('bottomright')

WhichCorner('topleft', r=6)
WhichCorner('bottomleft', r=6)
WhichCorner('topright', r=6)
WhichCorner('bottomright', r=6)

WhichCorner('topleft', c=7)
WhichCorner('bottomleft', c=7)
WhichCorner('topright', c=7)
WhichCorner('bottomright', c=7)

WhichCorner('topleft', r=8, c=3)
WhichCorner('bottomleft', r=8, c=3)
WhichCorner('topright', r=8, c=3)
WhichCorner('bottomright', r=8, c=3)

## End(Not run)
```

Index