Package ‘taRifx’

February 20, 2015

Type Package
Title Collection of utility and convenience functions.
Version 1.0.6
Date 2012-11-10
Author Ari B. Friedman
Maintainer Ari B. Friedman <abfriedman@gmail.com>
Description A collection of various utility and convenience functions.
License GPL (>= 2)
LazyLoad yes
Suggests xtable, grid, lattice, caTools, pspline, ggplot2, gdata, RSQLite, data.table, stringr
Imports reshape2, plyr
Collate 'Contributed.R' 'Rfunctions.R'
NeedsCompilation no
Repository CRAN
Date/Publication 2014-05-29 15:01:42

R topics documented:

as.data.frame.by .......................................................... 2
as.matrix.by ............................................................. 3
autoplot.microbenchmark ................................................. 4
between ................................................................. 4
bytable ............................................................... 5
categorize .......................................................... 6
compareplot .......................................................... 7
daysofweek .......................................................... 8
destring ............................................................. 9
distinct .............................................................. 9
evens ............................................................... 10
expandDF ........................................................... 10
as.data.frame.by

Convert the results of by() to a data.frame.

Description

Converts the results of by() to a data.frame if possible, (reducing dimensionality and adding repetition as necessary)

Usage

```r
## S3 method for class 'by'
as.data.frame(x, row.names = NULL,
             optional = FALSE,
             colnames = paste("IDX", seq(length(dim(x))), sep = ","),
             na.rm = TRUE, ...)
```
as.matrix.by

Arguments

x The by object
row.names Names of the rows. If NULL, function tries guessing them
optional Ignored.
colnames Names of columns
na.rm Remove NAs or not.
... Pass-alongs.

Value

A data.frame.

Examples

test.by <- by( ChickWeight$weight, ChickWeight$Diet, mean)
test.by
class(test.by)
str(test.by)
test.df <- as.data.frame(test.by)
str(test.df)

as.matrix.by  Coerces a by object into a matrix (only tested on a 2d objects).

Description

Coerces a by object into a matrix (only tested on a 2d objects).

Usage

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

Arguments

x is a by object to convert to a matrix
... ignored

Value

a matrix
### autoplot.microbenchmark

*Autoplot method for microbenchmark objects: Prettier graphs for microbenchmark using ggplot2*

**Description**

Uses ggplot2 to produce a more legible graph of microbenchmark timings

**Usage**

```r
## S3 method for class 'microbenchmark'
autoplot(object, ..., y_max = max(by(object$time, object[["expr"]], uq)) * 1.05)
```

**Arguments**

- **object**: A microbenchmark object
- **...**: Ignored
- **y_max**: The upper limit of the y axis (defaults to 5 percent more than the maximum value)

**Value**

A ggplot2 plot

---

### between

*Classify values into groups based on which numbers they're between*

**Description**

Classify values into groups based on which numbers they're between. `quantile.cutpoints` creates a data.frame of quantiles for feeding into e.g. `categorize()`

**Usage**

```r
between(vec, cutpoints)

bin(vec, n = 10)

quantile_cutpoints(vec, probs)
```
bytable

Arguments

vec Numeric vector to classify
cutpoints Vector listing what values the grouping should be done on. Should include the max and the min in this list as well.
n Number of groups to bin into
probs Probabilities at which to create cutpoints

Value

Vector of length(vec) indicating which group each element is in (for between). Or vector of length(vec) indicating the lower bound of the group that it’s in.

See Also
categorize

Examples

test <- runif(100)
between(test,c(0,.1,.5,.9,1))
bin(test,n=5)

bytable Produces a nice summary table by groupings

Description

produces a nice summary table by groupings, suitable for use with latex.table.by().

Usage

bytable(datavec, indices, ops = c(quote(mean)),
ops.desc = list(mean = "Mean"), na.rm = TRUE)

Arguments

datavec Vector to be analyzed
indices Indices should be a list of grouping vectors, just like you would pass to -by-, but with sensible names for each vector
ops Vector of quote’d operations to perform
ops.desc Vector of length length(ops) containing the column labels for the operations.
na.rm Remove NAs or not
... other arguments to pass to by
Value
data.frame

See Also
latex.table.by

Examples
bytable(runif(100), indices = list(rep('a', 'b'), 50))

categorize Categorize a vector based on a data.frame with two columns, the low and high end points of each category.

Description
Categorize a vector based on a data.frame with two columns, the low and high end points of each category.

Usage
categorize(vec, cutpoints.df, match.min = TRUE, names = TRUE)

Arguments
vec vector to categorize
cutpoints.df quantile_cutpoints will create a data.frame of the proper format here
match.min Whether to include or exclude the minimum value
names Return names or row numbers

Value
Categorized values

See Also
quantile_cutpoints
**Description**

Bar plot divided by three groupings

**Usage**

```r
compareplot(formula, data.frame, show.outlines = FALSE,
             main = "", x.label = "", div.axis.major = 10,
             div.axis.minor = 20, log.x = FALSE,
             colors.plot = c("salmon", "blue", "olivedrab", "cyan", "brown", "green", "purple"),
             panel = "panel.tuftebox", box.width.large.scale = 0.4,
             box.width.small.scale = 0.25, box.show.mean = TRUE,
             box.show.box = FALSE, box.show.whiskers = FALSE, ...) 
```

**Arguments**

- `formula`: Plot formula. Of the form: `~cts|group1*group2*group3`, where cts is the continuous data you want to make boxplots out of, and group_ are factors to group by in descending hierarchical order.
- `data.frame`: Data.frame containing data
- `show.outlines`: Whether to include boxes around plots or leave it open
- `main`: Plot text
- `x.label`: X axis label
- `div.axis.major`: How many major axis ticks to use
- `div.axis.minor`: How many minor axis ticks to use
- `log.x`: Log transform the x data?
- `colors.plot`: Plot colors
- `panel`: Panel function to use
- `box.width.large.scale`: box.width.large.scale here~~
- `box.width.small.scale`: box.width.small.scale here~~
- `box.show.mean`: here~~
- `box.show.box`: here~~
- `box.show.whiskers`: box.show.whiskers here~~
- `...`: Other arguments to pass to lattice function

**Value**

Plot
Examples

library(datasets)
cw <- transform(ChickWeight, 
    Time = cut(ChickWeight$Time,4))
cw$Chick <- as.factor( sample(LETTERS[seq(3)], nrow(cw), replace=TRUE) )
levels(cw$Diet) <- c("Low Fat","Hi Fat","Low Prot.","Hi Prot.")
compareplot(~weight | Diet * Time * Chick, 
data.frame=cw, 
    main = "Chick Weights", 
    box.show.mean=FALSE, 
    box.show.whiskers=FALSE, 
    box.show.box=FALSE
)

daysofweek

Return a vector of the days of the week, in order

Description

Return a vector of the days of the week, in order

Usage

daysofweek(start.day = "Monday")

Arguments

start.day Day of the week to begin the week with (as a text item)

Value

Character vector of length 7

Examples

daysofweek("Sunday")
**destring**

*Convert character vector to numeric, ignoring irrelevant characters.*

**Description**

Convert character vector to numeric, ignoring irrelevant characters.

**Usage**

```r
destring(x, keep = "0-9.-")
```

**Arguments**

- `x`: A vector to be operated on
- `keep`: Characters to keep in, in bracket regular expression form. Typically includes 0-9 as well as the decimal separator (. in the US and , in Europe).

**Value**

vector of type numeric

**Examples**

```r
test <- "50,762.83a"
destring(test)
```

---

**distinct**

*Returns number of distinct observations in each column of a data frame or in a vector*

**Description**

Returns number of distinct observations in each column of a data frame or in a vector.

**Usage**

```r
distinct(input, na.rm = TRUE)
```

**Arguments**

- `input`: data.frame or vector
- `na.rm`: remove nas or not

**Value**

Num of distinct obs
Examples

```r
x <- sample(letters[1:3], 10, replace=TRUE)
#distinct(x)
```

---

**Description**

Takes an integer vector and returns every odd or even element

**Usage**

```r
evens(vec)
```

**Arguments**

- `vec` Integer vector

**Value**

Returns an integer vector consisting of only the odd/even elements.

**Examples**

```r
x <- as.integer(c(6,3,4,7,8,10,47,82,7))
evens(x)
odds(x)
```

---

**Description**

expandDF takes a dataframe and replicates the chosen observations n times

**Usage**

```r
expandDF(df, obs, numtimes = 1)

splitDF(df, splitvar)

unsplitDF(splitdfs)
```
Arguments

- **df**: Data.frame to be manipulated
- **obs**: Vector to select rows of df (e.g. vector of row numbers or a boolean of length nrow(df))
- **numtimes**: Number of times to replicate
- **splitvar**: Name of variable which defines groups on which df will be split
- **splitdfs**: List of data.frames to recombine (generally created by splitDF)

Details

splitDF takes a dataframe and splits it into a bunch of data.frames held in a list, according to one variable
unsplitDF takes a list of data.frames produced by splitDF and returns them as one appended data.frame

Value

expandDF and unsplitDF return a data.frame splitDF returns a list of data.frames

Examples

```r
library(Hdatasets)
# Duplicate a dataset
expandDF(sleep,TRUE)
# Expand the final observation
expandDF(sleep, nrow(sleep), numtimes=10)
# Split a data.frame by group
s.df <- splitDF(sleep, 'group')
s.df
# Reconstitute original data.frame
unsplitDF(s.df)
```

---

**fpart**

*Obtain the fractional part of a numeric*

Description

Takes a numeric vector and returns a vector of the numbers after the decimal place

Usage

```r
fpart(vec)
```

Arguments

- **vec**: A numeric vector of any length
Value

A vector of the same length as the input vec containing only the decimal component.

Examples

```r
x <- runif(100)
fpart(x)
```


hist_horiz

---

Kludgy horizontal histogram function (really should just fix the lattice equivalent)

Description

Kludgy horizontal histogram function (really should just fix the lattice equivalent)

Usage

```r
hist_horiz(formula, data, n = 20)
```

Arguments

- `formula`  Plot formula
- `data` Data.frame
- `n` Number of groups

Value

plot

See Also

hist

Examples

```r
library(lattice)
library(datasets)
hist_horiz(~ len | supp, data=ToothGrowth, n=5)
```
homogenous

Returns whether a vector is homogenous or not

Description

Returns TRUE/FALSE if every element of vector is identical/not.

Usage

homogenous(vec)

Arguments

vec Vector to be compared

Value

TRUE if every element of a vector is identical; FALSE otherwise.

See Also

See also all any

Examples

homogenous(c(rep("A",10),"A"))
homogenous(c(rep("A",10),"B"))

iapply

Iteratively (recursively) apply a function to its own output

Description

Iteratively (recursively) apply a function to its own output

Usage

iapply(X, FUN, init, ...)

Arguments

X a vector of first arguments to be passed in
FUN a function taking a changing (x) and an initial argument (init)
init an argument to be "worked on" by FUN with parameters x[1], x[2], etc.
... Arguments passed to FUN.
Value

the final value, of the same type as init

Examples

```r
vec <- "xy12"
mylist <- list(c("x","a"), c("y","b"), c("a","f"))
iapply(mylist, FUN=function(repvec,x) {
    gsub(repvec[1],repvec[2],x)
}, init=vec)
```

Description

`japply` is a wrapper around `sapply` that only `sapply`s to certain columns

Usage

```r
japply(df, sel, FUN = function(x) x, ...)
```

Arguments

- `df`: data.frame
- `sel`: A logical vector or vector of column numbers to select
- `FUN`: The function to apply to selected columns
- `...`: Pass-alongs to `sapply`

Value

A data.frame

---

Value

Convenience functions to return the last/first element of a vector

Usage

```r
last(vec)
```
**latex.table.by**

**Arguments**

vec

Vector of any type

**Value**

Vector of length 1 of same type as vec

**Examples**

```r
test <- seq(10)
first(test)
last(test)
```

---

\[ \text{latex.table.by} \quad \text{Exports a latex table with the first N columns being multirow grouping variables.} \]

**Description**

Given a data.frame with the first N columns of grouping variables, makes each group print nicely in a LaTeX table.

**Usage**

```r
latex.table.by(df, num.by.vars = 1, ...)
```

**Arguments**

df

data.frame with first num.by.vars columns being grouping variables

num.by.vars

Number of columns to interpret as grouping vars

...

Other arguments to pass to xtable

**Value**

A modified xtable object.

**See Also**

xtable, bytable
merge.list

Method to merge two lists Matches names of each list element and combines any sub-elements

Description

Method to merge two lists Matches names of each list element and combines any sub-elements

Usage

## S3 method for class 'list'
merge(x, y, ...)

Arguments

<table>
<thead>
<tr>
<th>x</th>
<th>First list</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>Second list</td>
</tr>
<tr>
<td>...</td>
<td>Other arguments</td>
</tr>
</tbody>
</table>

Value

A list

Examples

```r
x <- list( A=list(p=runif(5)), B=list(q=runif(5)) )
y <- list( A=list(r=runif(5)), C=list(s=runif(5)) )
merge.list(x,y)
```
middle.group

Return a vector containing the locations of the middle of every group in a vector, either as a numerical index or as a TRUE/FALSE boolean.

Description

This function uses run length encoding to determine the middle of every group of repeated values within a larger vector.

Usage

middle.group(vec, type = "tf")

Arguments

vec Any vector which you want to know the middle of.

Arguments

type Either "tf" to return a boolean or "loc" to return a set of numerical locations.

Value

If type=="tf": Boolean of length length(vec) containing TRUE if the middle of a grouping and FALSE if not. If type=="loc": Vector of length equal to the number of groups in vec, containing locations of the group centers. Ties (for groups of even length) are broken by rounding up.

Examples

test <- c(1,2,2,2,2,2,2,2,2,2,1)
middle.group(test)
middle.group(test,type="loc")

munch

Recursively delete entries containing 'what' before entry pointed to by 'which'

Description

Recursively delete entries containing 'what' before entry pointed to by 'which'

Usage

munch(x, wch, what = "")

Arguments

x data vector

Arguments

wch Vector of indices to check preceding element for 'what'

what What to check for and delete if found in preceding element
panel.ecdf

Value
A vector of the same type as x with all the ‘what’’s removed if they were at the ‘which’-(1,2,3...) locations

Examples

```R
x <- c("a", "", "b", "", "", "", "", "c", "d", "", "", "", "", "e", "")
munch(x, c(3,8,9,13))
```

panel.ecdf

Various panel functions

Description
panel.ecdf is a panel function for xyplot to create lattice plots of the empirical CDF. panel.densityplot.enhanced is a panel function for densityplot to add in descriptives as text. panel.xyplot_rug is an xyplot panel function with rug plots on x and y axes.

Usage

```R
panel.ecdf(x, y, lines = TRUE, ...)
panel.densityplot.enhanced(x, ...)
panel.xyplot_rug(x, y, rug.color = "grey", ...)
```

Arguments

- **x**: Numerical vector
- **y**: Numerical vector
- **lines**: Whether to connect the points with lines or not
- **...**: Arguments to pass along to other lattice functions
- **rug.color**: Color of rugplots

Value
Lattice panel object
### prettify

*Function to prettify the output of another function using a 'var.labels' attribute. This is particularly useful in combination with `read.dta` et al.*

#### Description

Function to prettify the output of another function using a 'var.labels' attribute. This is particularly useful in combination with `read.dta` et al.

#### Usage

```r
prettify(dat, expr)
```

#### Arguments

- **dat**
  - A data.frame with attr `var.labels` giving descriptions of variables
- **expr**
  - An expression to evaluate with pretty `var.labels`

#### Value

The result of the expression, with variable names replaced with their labels

#### Examples

```r
testdf <- data.frame( a=seq(10), b=rnorm(10), c=rnorm(10) )
attr(testDF,"var.labels") <- c("Identifier","Important Data","Lies, Damn Lies, Statistics")
prettify( testDF, quote(str(dat)) )
```

### readdir

*Loads all readable files in a directory into a list, with names according to the filenames*

#### Description

Loads all readable files in a directory into a list, with names according to the filenames

#### Usage

```r
readdir(path, exclude = "", filename.as.variable = "filename", stack = FALSE)
```
Arguments

- **path**: is the directory path
- **exclude**: is a regular expression. Matching filenames will be excluded
- **filename.as.variable**: is a variable name to store the filename. "" means it will not be stored.
- **stack**: if true attempts to stack the resultant data.frames together into a single data.frame

Value

A list of data.frames or a single data.frame

---

**remove.factors**  
*Converts all factors in a data.frame to character.*

---

Description

Converts all factors in a data.frame to character.

Usage

```r
remove.factors(df)
```

Arguments

- **df**: A data.frame

Value

data.frame

Examples

```r
my.test.df <- data.frame(grp=rep(c("A","B"),10),data=runif(20))
remove.factors(my.test.df)
```
**rep_along**

*Repeat a vector until it matches the length of another vector*

**Description**

Repeat a vector until it matches the length of another vector

**Usage**

```
rep_along(x, along.with)
```

**Arguments**

- `x`: Vector to be repeated
- `along.with`: Vector whose length to match

**Value**

A vector of same type as `x`

**Examples**

```
rep_along(1:4, letters)
```

---

**reshapeasy**

*reshapeasy: Easier reshaping from "wide" to "long" and back again*

**Description**

reshapeasy is a wrapper around base R’s reshape which allows for saner syntax. In particular, it makes it possible to reverse the operation by only specifying that the direction change (e.g. the names of the arguments are consistent between the direction of reshaping).

**Usage**

```
reshapeasy(data, direction,
    id = (sapply(data, is.factor) | sapply(data, is.character)),
    vary = sapply(data, is.numeric), omit = c("_", "."),
    vars = NULL, ...)
```
roundnear

**Arguments**

- **data**: A data.frame to be reshaped
- **direction**: "wide" or "long"
- **vars**: The names of the (stubs of) the variables to be reshaped (if omitted, defaults to everything not in id or vary)
- **id**: The names of the variables that identify unique observations
- **vary**: The variable that varies. Going to wide this variable will cease to exist. Going to long it will be created.
- **omit**: vector of characters which are to be omitted if found at the end of variable names (e.g. price_1 becomes price in long)
- **...**: Options to be passed to stats::reshape

**Value**

A data.frame

**Author(s)**

Written with the help of the StackOverflow R community, see http://stackoverflow.com/questions/10055602/wrapping-base-r-reshape-for-ease-of-use

---

**roundnear**

*Rounds a numeric vector to arbitrary values (not just decimal values as with round) or to a specified number of significant digits.*

**Description**

Rounds a numeric vector to arbitrary values (not just decimal values as with round). E.g. allows you to round to nearest 0.3 instead of just nearest 1 or 0.1

**Usage**

```r
roundnear(vec, roundvec)
round_sigfig(vec, digits = 2)
```

**Arguments**

- **vec**: numeric vector
- **roundvec**: What value to round things to (e.g. nearest 1, 10, 0.3, etc.). Typically a single item to apply to all of vec. If of length greater than 1, usual wrapping rules apply.
- **digits**: Number of significant digits to round to
searchPattern

Value

Rounded numeric vector of length length(vec)

References

http://en.wikipedia.org/wiki/Significant_figures

Examples

roundnear( runif(10) , .03 )

searchPattern Create a vector that starts with a given number and widens out

Description

Create a vector that starts with a given number and widens out

Usage

searchPattern(center = 0, length = 5, interval = 1)

Arguments

center Number to center search pattern around

length Number of elements in search pattern

interval Distance between each element

Value

numeric vector

Examples

library(gdata)

searchPattern()
shift

Shifts a vector’s elements left or right by N elements.

Description

Shifts a vector’s elements left or right by N elements.

Usage

shift(x, ...)

## Default S3 method:
shift(x, n = 1, wrap = TRUE,
    pad = FALSE, ...)

## S3 method for class 'data.frame'
shift(x, ...)

Arguments

x A vector to be operated on
n Number of rows to shift by (if negative, shift to right instead of left)
wrap Whether to wrap elements or not (adds the entry at the beginning to the end)
pad Whether to pad with NAs or not. pad does nothing unless wrap is false, in which 
case it specifies whether to pad with NAs
... Other items to pass along

Value

vector of the same type as vec

Examples

test <- seq(10)
shift(test)
sides

Figure out how many "sides" a formula has See also SimonO101’s answer at http://stackoverflow.com/a/16376939/636656

Description

Figure out how many "sides" a formula has See also SimonO101’s answer at http://stackoverflow.com/a/16376939/636656

Usage

sides(x, ...)

## Default S3 method:
sides(x, ...)

## S3 method for class 'formula'
sides(x, ...)

Arguments

x The object to calculate the sidedness of
...
Other items to pass along

Value

An integer of the number of sides

Examples

test <- list( ~ a + b, a ~ b + c, b + c ~ a, ~ a ~ b, a ~ b ~ c, a+b+c|d-c~d-e-f~g )
sapply(test,sides)

sort.data.frame Sort a data.frame

Description

Sorts a data frame by one or more variables

Usage

## S3 method for class 'data.frame'
sort(x, decreasing = NULL, formula, ...)

## S3 method for class 'formula'
sort(x, order = NULL, formula, ...)

## Default method
sort(x, order = NULL, method = NULL, ...)

Arguments

x The data frame to be sorted
order The order of the sorting
formula A formula
method The sorting method
... Other items to pass along

Value

A data frame sorted by the specified variables

Examples

test <- matrix(c(1, 2, 3, 4, 5), nrow = 2, ncol = 2)
sort(test, 1)
sort(test, 2)
sort(test, 2, decreasing = TRUE)
sort(test, formula = ~ ., decreasing = TRUE)
sort(test, method = "quicksort")

## S3 method for class 'formula'
sort(x, formula = ~ ., method = NULL, ...)

## Default method
sort(x, formula = ~ ., method = NULL, ...)

Arguments

x The data frame to be sorted
formula A formula
method The sorting method
... Other items to pass along

Value

A data frame sorted by the specified variables

Examples

test <- matrix(c(1, 2, 3, 4, 5), nrow = 2, ncol = 2)
sort(test, 1)
sort(test, 2)
sort(test, 2, decreasing = TRUE)
sort(test, formula = ~ ., decreasing = TRUE)
sort(test, method = "quicksort")
splitc

Split data over columns

Description

Split data column-wise on data.frame, matrix and array or element-wise on a list.

Arguments

- **x**: Data.frame to sort
- **formula**: Formula by which to sort the data.frame (e.g. ~group1+group2 sorts first by group1 then by group2)
- **decreasing**: Ignored. Exists for compatibility with generic S3 method.
- **...**: Used to pass .drop=FALSE to

Value

Returns a sorted data.frame

Note


If you are Kevin Wright, please contact me. I have attempted to reach you by every means thinkable, to no avail. My assumption is that this is in the public domain since you posted it for others to use, but please tell me if that is not the case.

Author(s)

Kevin Wright, with generic compatibility by Ari B. Friedman

See Also

- arrange

Examples

```r
library(datasets)
sort.data.frame(ChickWeight, formula=~weight+Time)

mydf <- data.frame(col1 = runif(10))
rownames(mydf) <- paste("x", 1:10, sep="")
sort(mydf, f = ~col1) # drops a dimension
sort(mydf, f = ~col1, drop = FALSE) # does not drop a dimension (returns a data.frame)
```
Usage

\texttt{splitc(X, INDEX, FUN = NULL, ...)}

Arguments

- \texttt{X} A \texttt{data.frame}, \texttt{matrix}, \texttt{array} or a \texttt{list}.
- \texttt{INDEX} A factor of length(X) (number of columns or list elements). If not a factor, it will be coerced into one.
- \texttt{FUN} A function to be applied to individual subset of data (each factor level). If not provided (\texttt{null}), raw (split) data is returned.
- \texttt{...} Additional arguments to \texttt{FUN}.

Details

Function splits a \texttt{data.frame}, \texttt{matrix} and \texttt{array} column-wise according to \texttt{INDEX} and list is sliced according to \texttt{INDEX}. Output is returned as a list of the same length as the number of levels in \texttt{INDEX}.

Value

A list of the same length as there are factor levels in \texttt{INDEX}.

Note

Simplification sensu \texttt{tapply} is not yet implemented.

Author(s)

Roman Lustrik <roman.lustrik@biolitika.si>

See Also

\texttt{tapply, by, aggregate, apply, split}

Examples

```r
my.list <- list(a = runif(5), b = runif(5), c = runif(5), d = runif(5), e = runif(10),
                  f = runif(10), g = runif(10), h = runif(10), i = runif(10), j = runif(10))
my.df <- as.data.frame(my.list)
my.matrix <- as.matrix(my.df)

ind <- factor(c(1,1,1,1,2,3,4,4,4,4))
ind2 <- factor(c(1,1,1,1,2,3,4,4,4,4), levels = 1:5)

# Applies mean to each, you must use \texttt{code(colMeans)},
# as \texttt{code(mean)} is deprecated for \texttt{code(data.frame)}
\texttt{splitc(X = my.df, INDEX = ind, FUN = colMeans)}
\texttt{splitc(X = my.matrix, INDEX = ind2)} # level 5 empty because not populated
\texttt{splitc(X = my.list, INDEX = ind, FUN = sum)} # applied to elements INDEX-wise
```
stack.list  

Stack lists into data.frames

Description

Takes two types of data: (1) a list of data.frames, (2) a list of vectors, which it interprets as rows of a data.frame

Usage

```r
## S3 method for class 'list'
stack(x, label = FALSE, ...)
```

Arguments

- `x`: A list of rbindable objects (typically data.frames)
- `label`: If false, drops labels
- `...`: Ignored

Details

Method of stack for lists of data.frames (e.g. from replicate() ) Takes two types of data:

Value

Typically a data.frame

Examples

```r
dat <- replicate(10, data.frame(x=runif(2),y=rnorm(2)), simplify=FALSE)
str(dat)
stack(dat)
```

---

`tab`  

Table function which lists NA entries by default This is a simple wrapper to change defaults from the base R table()

Description

Table function which lists NA entries by default This is a simple wrapper to change defaults from the base R table()

Usage

```r
tab(..., exclude = NULL,
    useNA = c("no", "ifany", "always"), deparse.level = 1)
```
Arguments

... one or more objects which can be interpreted as factors (including character strings), or a list (or data frame) whose components can be so interpreted. (For as.table and as.data.frame, arguments passed to specific methods.)

exclude levels to remove for all factors in .... If set to NULL, it implies useNA = "al- ways". See 'Details' for its interpretation for non-factor arguments.

useNA whether to include NA values in the table. See 'Details'.

deparse.level controls how the default dnn is constructed. See 'Details'.

Value

tab() returns a contingency table, an object of class "table", an array of integer values

See Also

table

Description

Plot a title page containing the given text. Good for breaking up sections of plot PDFs.

Usage

title.page.new(title.text = "")

Arguments

title.text Text to plot on its own page

Value

Plot

Examples

  title.page.new("Page break!")
trues  

Return vector of equal length containing all TRUEs

Description
Takes a vector and returns a vector of equal length containing all trues (used for selecting all of a given vector)

Usage
trues(vec)

Arguments
vec any vector (or valid object for length)

Value
a vector of TRUEs of the length of the object passed to it

Examples
x <- runif(100)
trues(x)

unfactor.data.frame  Convert all factors to character

Description
Convert all factors to character

Usage
unfactor.data.frame(x)

Arguments
x data.frame

Value
data.frame
write.sanitized.csv  Outputs a sanitized CSV file for fussy input systems e.g. ArcGIS and Mechanical Turk Performs three cleansing actions: converts text to latin1 encoding, eliminates funny characters in column names, and writes a CSV without the leading row.names column

Description

Outputs a sanitized CSV file for fussy input systems e.g. ArcGIS and Mechanical Turk Performs three cleansing actions: converts text to latin1 encoding, eliminates funny characters in column names, and writes a CSV without the leading row.names column

Usage

write.sanitized.csv(x, file = "", ...)

Arguments

x  The data.frame to clean and write
file  The filename to write to
...  Arguments to pass to write.csv

Value

NULL

xtable.CrossTable  Add in methods to handle CrossTable objects in xtable

Description

Add in methods to handle CrossTable objects in xtable

Usage

## S3 method for class 'CrossTable'
xtable(x, caption = NULL,
  label = NULL, align = NULL, digits = NULL,
  display = NULL, beta.names = NULL, ...)
Arguments

- **x**: Model object
- **caption**: Caption for table
- **label**: See ?xtable
- **align**: See ?xtable
- **digits**: See ?xtable
- **display**: See ?xtable
- **beta.names**: See ?xtable
- ... Arguments to pass to xtable

Value

xtable object

See Also

xtable

---

**xtable.lme**

*Add in methods to handle LME objects in xtable*

Description

Add in methods to handle LME objects in xtable

Usage

```r
xtable.lme(x, caption = NULL, label = NULL, align = NULL,
           digits = NULL, display = NULL, beta.names = NULL, ...)
```

Arguments

- **x**: Model object
- **caption**: Caption for table
- **label**: See ?xtable
- **align**: See ?xtable
- **digits**: See ?xtable
- **display**: See ?xtable
- **beta.names**: See ?xtable
- ... Arguments to pass to xtable
xtablelm

Value
xtable object

See Also
xtable

---

**xtablelm**

*Produces the output of an *lm* object as it appears in the R console when you type* `summary(lmobject)`

Description

Produces the output of an *lm* object as it appears in the R console when you type `summary(lmobject)`

Usage

```r
xtablelm(lm.object, titref, labname, extracaption = NULL)
```

Arguments

- `lm.object` the name of your linear model object that you want to make a summary table for.
- `titref` the label name of the equation you made in Latex to cross reference
- `labname` the label name you want for this table
- `extracaption` adds whatever text string you pass to the title of the table.

Value

xtable object

See Also

xtable

Examples

```r
##
```
Index

*Topic **TRUE**
- trues, 30

*Topic **boolean**
- trues, 30

*Topic **manip**
- splitc, 26

aggregate, 27
- all, 13
- any, 13
- apply, 27

arrange, 26
- as.data.frame.by, 2
- as.matrix.by, 3
- autoplot.microbenchmark, 4

between, 4
- bin (between), 4
- by, 27
- bytable, 5

categorize, 6
- compareplot, 7

daysofweek, 8

destring, 9
- distinct, 9

evens, 10
- expandDF, 10

first(last), 14
- fpart, 11

hist_horiz, 12
- homogenous, 13

iapply, 13

japply, 14

last, 14
latex.table.by, 15
merge.list, 16
middle.group, 17
munch, 17
odds (evens), 10
panel.densityplot.enhanced
- (panel.ecdf), 18
panel.ecdf, 18
panel.xplot_rug (panel.ecdf), 18
prettify, 19
quantile.cutpoints, 6
quantile.cutpoints (between), 4
readdir, 19
remove.factors, 20
rep_along, 21
reshapeasy, 21
round_sigfig (roundnear), 22
roundnear, 22
searchPattern, 23
shift, 24
sides, 25
sort.data.frame, 25
split, 27
splitc, 26
splitc-package (splitc), 26
splitDF (expandDF), 10
stack.list, 28

tab, 28

tapply, 27
title.page.new, 29
trues, 30
unfactor.data.frame, 30
unsplitDF (expandDF), 10
write.sanitized.csv, 31
xtable, 32, 33
xtable.CrossTable, 31
xtable.lme, 32
xtable.summary.lme (xtable.lme), 32
xtablelm, 33