Package 'JBTools'

July 2, 2014

Title JBs tools and helper functions.

Version 0.7.2.6

Date 2014-01-01

Author Jannis v. Buttlar

Maintainer Jannis v. Buttlar <jbuttlar@bgc-jena.mpg.de>

Description Collection of several tools and helper functions used across the other packages of J. Buttlar (ncdf.tools and spectral.methods).

Imports foreach, parallel, gplots, colorspace, plotrix

Suggests doMC

License GPL-2

LazyLoad yes

Depends R (>= 2.14.0)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-04-03 14:00:41

R topics documented:

`JBTools-package` *JBs tools and helper functions.*

Description

Collection of several tools and helper functions used across the other packages of J. Buttlar (ncdf.tools and spectral.methods).

Details

Package:	JBTools
Title:	JBs tools and helper functions.
Version:	0.7.2.5
Date:	2014-01-01
Author:	Jannis v. Buttlar
Maintainer:	Jannis v. Buttlar <jbuttlar@bgc-jena.mpg.de>
Imports:	foreach, parallel, gplots, colorspace, plotrix
Suggests:	doMC
License:	GPL-2
LazyLoad:	yes
Depends:	R (>= 2.14.0)

Author(s)

Jannis v. Buttlar

checkInstalled *Check whether a command can be invoked via the command line*

Description

checkInstalled checks whether an external command can be run on the command line.

Usage

```
checkInstalled(commandName)
```

Arguments

commandName	character string: name of the program/command to check

Details

The test is a simple wrapper around Sys.which which returns TRUE if which returns a character string and FALSE if not.

Value

logical: whether the program is installed.

Author(s)

Jannis v. Buttlar

colorChangeDarkness *Change the dark/brightness of a color*

Description

Function to yield darker or brighter variants of the same color by mixing this color with white or black

Usage

```
colorChangeDarkness(col, factor)
```

Arguments

col	color to change. Can be a R color name or a hexadecimal color specification
factor	decimal factor to darken/brighten the color with. Values < 1 lead to darker colors, values > 1 to brighter colors.

Value

hexadecimal code for the new color.

Author(s)

Jannis v. Buttlar

See Also

mixcolor

`convertArgs2String` *Save function argument values to a character string*

Description

convertArgs2Strings saves function argument settings to a character string. details « convertArgs2Strings saves function argument settings to a character string. This is used in the ncdf routines that save these argument values as attributes of the target ncdf files.

Usage

```
convertArgs2String()
```

Value

character string: arguments given to function converted into a string.

Author(s)

Jannis v. Buttlar

`getSysinfo` *Compile system information*

Description

getSysinfo is a convenience function to compile some information about the computing system and environment used.

Usage

```
getSysinfo()
```

Details

The function is mainly used to save the system environment information in ncdf files containing the results of some calculations.

Value

character string with all version and system information of the current R system

Author(s)

Jannis v. Buttlar

`getVecInfo` *Compute vector summary statistics*

Description

This function computes several summary statistics of a vector.

Usage

```
getVecInfo(x)
```

Arguments

`x`

Value

vector with the vector statistics (min, max, mean, sdev, range, ratio na, ratio na inner, ratio INF)

Author(s)

Jannis v. Buttlar

indexDimVecs2Matrix *Transform integer indices to an index matrix*

Description

Transform integer indices to an index matrix. The input have to be index vector for each dimension of the target array. Extract, indexVec2Matrix

Usage

```
indexDimVecs2Matrix(...)
```

Arguments

...	integer vectors: indices to use for the different dimensions

Value

Index matrix

Author(s)

Jannis v. Buttlar

indexVec2Matrix *Transform a vector index to an index matrix*

Description

Transform an index vector into a index matrix. Extract

Usage

```
indexVec2Matrix(index, dim)
```

Arguments

index	vector index
dim	length of the two dimensions of the array to index. Identical to the result of dim(array).

Value

index array

Author(s)

Jannis v. Buttlar

isSeriesConstant *Check whether a vector is constant*

Description

isSeriesConstant checks whether a series is constant (up to a certain degree).

Usage

```
isSeriesConstant(x, tresh.const = 1e-12, ratio.const = 0.05)
```

Arguments

x	numeric vector: series to test.
tresh.const	numeric: maximum deviation allowed which is still considered to be constant.
ratio.const	numeric: ratio of the series which is allowed to be not constant for the whole series to be still considered to be constant.

Details

isSeriesConstant checks whether the amount of values deviating from the median of x for a higher value than tresh.const is bigger than ratio.const.

Value

logical: TRUE if series is constant, FALSE otherwise.

Author(s)

Jannis v. Buttlar

isSeriesContinuous *Test for continuous (non NA interrupted) series*

Description

isSeriesContinuous test whether a vector contains one non interrupted sequence of values (i.e. no NaN in between).

Usage

```
isSeriesContinuous(x)
```

Arguments

x	numeric vector: series to test

Details

The function returns TRUE when the vector contains one (and only one) non NA interupted sequence of values. E.g. it would also return TRUE for a vector that contains a sequence of NAs at the beginning and/or end of the vector.

Value

logical: whether the series contains gaps or not.

Author(s)

Jannis v. Buttlar

---

labelMargins *Label the margins of a plot*

Description

Writes equidistant labels to the margins of a plot.

Usage

```
labelMargins(labels, side = 1, ...)
```

Arguments

labels	character vector: labels to use
side	integer: side of the plot to label
...	further arguments passed to the plotting routines

Value

nothing is returned

Author(s)

Jannis v. Buttlar

MEF *Compute the modelling efficiency (MEF)*

Description

Calculates the modelling efficiency (MEF) of two arrays

Usage

```
MEF(prediction, observation)
```

Arguments

prediction	numeric: array 1
observation	numeric: array 2

Value

MEF of the two vectors

Author(s)

Jannis v. Buttlar

plotBG *Plot a colored plot background*

Description

plotBG colors the plot background of the plot region (not the device region!).

Usage

```
plotBG(color, exp.factor = 1, xlim = c(), ylim = c(), ...)
```

Arguments

color	color: which color to use
exp.factor	integer:possible expansion factor to use if very high values are plotted
xlim	
ylim	
...	further arguments passed to plot()

Details

The function opens a plot with no content and plots a very large polygon of the given color. Afterwards a new high level plot can be plotted over this background.

Value

Nothing is returned.

Author(s)

Jannis v. Buttlar

`plotColorScale` *Add a color scale to plots*

Description

plotColorScale is a wrapper function around color.legend to ease its usage.

Usage

```
plotColorScale(col, zlim = c(), pos = list(x = c(1.02, 1.08),
    y = c(0.1, 0.9)), align = "rb", gradient = "y", cex = 1,
    cex.title = 1, title = "cts/px", outer.range = c(FALSE, FALSE),
    legend = c())
```

Arguments

`col`	vector of color strings defining the palette to use
`zlim`	numeric vector (of length 2) defining the upper and lower limit of the values mapped to the color scale.
`pos`	
`align`	character: alignment option passed to color.legend
`gradient`	character: orientation option passed to color.legend
`cex`	numeric: character expansion factor for the text labels
`cex.title`	
`title`	character: the title of the color scale
`outer.range`	logical: whether to extend the scale over the zlim borders at its bottom and top.
`legend`	

Value

Nothing is returned.

Author(s)

Jannis v. Buttlar

See Also

color.legend

---

plotImageRotated *Plot a rotated image plot*

Description

plotImageRotated plots a raster/matrix using image() but supplying a rotated version of the matrix, i.e. it plots the matrix as it printed on the screen.

Usage

```
plotImageRotated(data, col.vals = c(), row.vals = c(), scale = TRUE,
    col = heat.colors(20), zlim = range(data, na.rm = TRUE),
    title = "", useRaster = TRUE, xlab = "", ylab = "", ...)
```

Arguments

data	matrix: data to be plotted
col.vals	numeric vector: coordinate values for the columns of data
row.vals	numeric vector: coordinate values for the rows of data
scale	logical: whether to plot a color scale besides the plot
col	color vector: colors to create the color scale.
zlim	numeric vector (length two): outer limits of the color-scale. Values above or below this range are colored brighter/darker than the maximum/minimum color (see ?plotColorScale).
title	character string: title of the color legend
useRaster	argument passed to image() to decide whether to draw polygons (FALSE) or use a bitmap raster (TRUE).
xlab	
ylab	
...	further arguments passed to image

Details

The normal image() function always plots a rotated version of a matrix. This function rotates the input to image in a way that the first entry of the matrix (col=1, row=1) shows up at the top-left corner of the plot. In other words the way a matrix would be displayed by printing it directly mapped to the plot.

Value

Nothing is returned.

Author(s)

Jannis v. Buttlar

See Also

image, plotColorScale, the plotting routines of the raster package

---

`plotNLines` *Plot many lines in one plot*

Description

plotNLines function uses different techniques to visualize many line plots on one display.

Usage

```
plotNLines(x.data = matrix(1:dim(y.data)[2], ncol = dim(y.data)[2],
    nrow = dim(y.data)[1], byrow = TRUE), y.data, option = c("normal",
    "diff.scales", "stacked")[1], n.lines.max = 30, grid = FALSE,
    scale = 1, plot.scale = TRUE, function.add = function() (1),
    bgc = "white", pch.axlink = 1, type = "l", colors = c(),
    xlim = c(), ylim = c(), xlab = c(), ylab = c(), labels = c(),
    lty = 1, title, yaxt = "s", ...)
```

Arguments

`x.data`	
`y.data`	numeric matrix/data frame: y-values with one series of values per row
`option`	character: which type of plot to use (see details)
`n.lines.max`	integer: only for 'stacked plots': how many lines to draw per panel
`grid`	logical: only for 'stacked plots': whether to draw a primitive grid
`scale`	numeric: only for 'stacked plots': scale factor to scale the y scale of the stacked plots
`plot.scale`	TRUE: only for 'stacked plots': whether to add a small scale showing the y axis scale
`function.add`	function: only for 'stacked plots': function to call after plotting the individual panels
`bgc`	color: color of the plot background
`pch.axlink`	integer: only for 'stacked plots': pch value for the symbol that links each line to its x-axis

type	standard plotting parameter
colors	colors to use for the different series
xlim	standard plotting parameter
ylim	standard plotting parameter
xlab	standard plotting parameter
ylab	standard plotting parameter
labels	standard plotting parameter
lty	standard plotting parameter
title	standard plotting parameter
yaxt	standard plotting parameter
...	further arguments passed to the plot() calls

Details

Many parameters are identical to standard plotting parameters (see ?par,?plot) and are not explained here. The function offers three options: 'normal': plots all plots in one coordinate system colored according to colors 'diff scales': plots all plots in the same region but uses different y axis scales. The visibility may be limited to ~7 plots. 'stacked': plots many plots in one region, all shifted vertically a bit to increase visibility. This allows for the easy comparison of many similar plots (e.g. time series) but reduces the details that are visible.

Author(s)

Jannis v. Buttlar

---

printStatus *Print a status message*

Description

This function prints a given string as a status report to the screen together with the current time.

Usage

```
printStatus(string.in)
```

Arguments

string.in	character string: status message to print

Value

Nothing is returned but a message is printed to the screen.

Author(s)

Jannis v. Buttlar

rangeZeroEqui *Compute a zero centered equi-sided range*

Description

rangeZeroEqui computes a zero centered equi-sided range, i.e. it the maximum absolute value and returns this as a positive and a negative value.

Usage

```
rangeZeroEqui(x)
```

Arguments

x	input vector

Value

vector

Author(s)

Jannis v. Buttlar

registerParallel *Set up a parallel computing front end*

Description

This function automatically sets up a cluster in a consistent way way for different parallel computing packages.

Usage

```
registerParallel(pckg.parallel = "doMC", max.cores = 0)
```

Arguments

pckg.parallel	character string: Package to use for parallel computing. Has to be (for the time beeing) one of doMC or doParallel.
max.cores	integer: amount of cores to use

Details

registerParallel sets up a cluster object of the selected package. In principle, this is a simple wrapper around the cluster creating functions of these packages that provides a unified usage.

Value

For "doMC": the amount of cores and for doParallel: the cluster object created.

Author(s)

Jannis v. Buttlar

See Also

foreach, registerDoMC

RMSE *Compute the residual mean square error (RMSE)*

Description

Calculates the residual mean square error (RMSE) of two arrays

Usage

```
RMSE(prediction, observation)
```

Arguments

prediction	numeric: array 1
observation	numeric: array 2

Value

RMSE of the two vectors

Author(s)

Jannis v. Buttlar

`userCoords` *Transfer relative to actual plot coordinate values*

Description

userCoords transfers realtive coordinate values (i.e. values between 0 and 1) to the actual coordinate system of the plot.

Usage

```
userCoords(x = c(), y = c())
```

Arguments

x	numeric vector(0-1): relative coordinates on the x axis
y	numeric vector(0-1): relative coordinates on the y axis

Details

x and y need to be values between 0 and 1. These values are then mapped to the coordinates used in the current plot.

Value

list with x and/or y component with values in the current coordinate system

Author(s)

Jannis v. Buttlar

Examples

```
plot(1:10)
text.coords <- userCoords(x=c(0.1,0.5),y=c(0.9,0.5))
text(text.coords,labels=c('1st Text','2nd Text'))
```

`whichClosest` *Find closest matches in two vectors*

Description

Function to find closest matches of vector A in vector B and return the respective indices.

Usage

```
whichClosest(x, x.match, arr.ind = FALSE)
```

Arguments

`x`	numeric vector: values that should be found in x.match
`x.match`	numeric vector: values the values of x should be matched against
`arr.ind`	

Value

integer vector: indices of the closest element of x.match to each entry in x

Author(s)

Jannis v. Buttlar

Index