Package 'JuicedGeneral'

August 6, 2021

Title General Juicy Functions

Version 0.6.2

Date 2021-08-06

Description These are some general useful functions that have not been included at any other location. Quite varied from calculations to plotting.

License Undecided

Depends data.table, devtools, haven, plotly, R (>= 4.1.0), readxl, scales, shiny, stats, stringr, svDialogs

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Author Joost Kuckartz [aut, cph, cre]

Maintainer Joost Kuckartz <jkuckartz1984@hotmail.com>

R topics documented:

`JuicedGeneral-package` *JuicedGeneral*

Description

These are some general useful functions that have not been included at any other location. Quite varied from calculations to plotting.

Author(s)

Maintainer: Joost Kuckartz <jkuckartz1984@hotmail.com> [copyright holder]

addcolumn *Add a column*

Description

Add a column to a data.table with a given name or use the variable as name (modification by reference).

Usage

```
addcolumn(data, colvalues, colname = NULL)
```

Arguments

data	The data.table to which to add a column.
colvalues	The values of the column to add (preferrably having the same length as the amount of rows in the data.table).
colname	The name of the new column. Default=NULL, which will read the original colvalues variable name and uses that as column name.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

addisoinfo *Add ISO information*

Description

Add ISO date information to a data.table

Usage

```
addisoinfo(datatable, datecolumn = "DATE", timezone = "")
```

Arguments

datatable	The data.table to add ISO weeknumber, year, and month to.
datecolumn	String identifying the column name in which a date is present. The column itself should be of the POSIXct class. Default="DATE".
timezone	String determining the timezone (from strptime). Default="", which uses the local time zone.

Value

Returns the modified data.table invisibly (modified by reference) with the added columns ISOWEEKNR, ISOYEAR and ISOMONTH.

See Also

Other data.table functions: addcolumn(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

Other date and time functions: makeconsistentdateformat(), makeconsistentdatetimeformat(), makedatecolumns(), makedatetimecolumns(), removeisoinfo(), seqdays()

---

bitsinbinary *Find bits in binary*

Description

Find which bits in a binary string are set to 1.

Usage

```
bitsinbinary(binaryvalues)
```

Arguments

binaryvalues	String vector containing binary bits.

Value

List with the locations which bits are set to 1. If no bits are set, or the provided binary value is invalid, the list value contains integer(0).

See Also

Other conversion functions: convertcoords_degrees2decimal(), hextobinary()

Examples

```
bitsinbinary(c("0010","0110"))
bitsinbinary(c("1010","string","0110"))
```

convertcoords_degrees2decimal *Converts degrees-minutes*

Description

Converts longitude or latitude in degrees-minutes to degrees-decimals.

Usage

```
convertcoords_degrees2decimal(degrees)
```

Arguments

degrees	The input longitude or latitude in degrees-minutes.

Value

The longitude or latitude in degrees-decimals.

See Also

Other conversion functions: `bitsinbinary()`, `hextobinary()`

copytoclipboard *Copy to Windows clipboard*

Description

Copy content (for example a data.table) to the Windows clipboard so it becomes available for paste-actions in Windows applications.

Usage

```
copytoclipboard(x)
```

Arguments

x	Content to copy.

Value

Nothing (but copies `x` to clipboard).

See Also

Other data.table functions: addcolumn(), addisoinfo(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

`createbins` *Create time series bins*

Description

Create time series bin by calculating statistical information for each bin containing data. It will use the rounded times as start and end times, even if data is starting at the end of the bin. There must be both time and data provided.

Usage

```
createbins(thetimes, thedata = NULL, binsize = "min")
```

Arguments

`thetimes`	Data.table with the first column being the DateTime and the second column being the values, or, a vector containing the DateTime only.
`thedata`	Vector containing the values correlated to the DateTime. Must be of equal length to `thetimes`. Default=`NULL`, which only uses the first parameter.
`binsize`	The bin size to split by. Can be `sec`, `min`, `hour`, `mday`, `mon`, `year`, `wday` or `yday`.

Value

Data.table with the starting times of the bins and each column with statistical information about that bin.

See Also

Other data functions: localmaxima(), localminimamaxima(), localminima(), removeoffset(), timeseriesstats()

Other time series functions: removeoffset(), timeseriesstats()

deg2rad *Degrees to radians*

Description

Degrees to radians

Usage

```
deg2rad(deg)
```

Arguments

deg	Degrees.

Value

Radians.

See Also

Other math functions: `rad2deg()`, `runavgweighed()`, `runavg()`, `runsd()`

dojoin *Join data.tables*

Description

This function executes all possible join options for data.tables. It has specific settings for the base and the additional data.table.

Usage

```
dojoin(
  basetable,
  additionaltable,
  basetablejoincolumns,
  additionaltablejoincolumns = NULL,
  additionaltableextractcolumns = NULL,
  removejoincols = NA,
  keepbaserows = TRUE,
  keepadditionalrows = FALSE
)
```

Arguments

basetable	The base data.table.
additionaltable	The additional data.table.
basetablejoincolumns	The column name of the base data.table to join on.
additionaltablejoincolumns	The column name of the additional data.table to join on. Default=NULL, which assumes the same column name as basetablejoincolumns.
additionaltableextractcolumns	The columns of the additional data.table to add to the base data.table. Default=NULL, which assumes that all columns need to be joined.
removejoincols	If TRUE, it will remove the provided basetablejoincolumns from the resulting table. If FALSE, it will keep the provided basetablejoincolumns in the result. If NA (default), it will check if basetablejoincolumns and additionaltablejoincolumns are the same. The column names that are different will result in the respective basetablejoincolumns to be removed. For example, let's take basetablejoincolumns=c("ID","SECID") and additionaltablejoincolu With removejoincols=TRUE, the column "ID2" will return in the result, but columns "ID" and "SECID" will not. With removejoincols=FALSE, the columns "ID.x", "ID.y", "ID2" and "SECID" will return. Note that column ID exists in both the base as well as additional table, which results in the "ID.x" and "ID.y" columns. With removejoincols=NA the column "ID" and "ID2" will return in the result, but column "SECID" will not.
keepbaserows	If TRUE (default), it keeps al rows of the base data.table and adds NA data for columns in the additional data.table where there is no match. If FALSE, rows in the base data.table for which there are no matching rows in the additional data.table are not returned.
keepadditionalrows	If TRUE, it keeps al rows of the additional data.table and adds NA data for columns in the base data.table where there is no match. If FALSE (default), rows in the additional data.table for which there are no matching rows in the base data.table are not returned.

Value

The joined data.table or, when join column(s) or extracted column(s) don't exist, the original base data.table.

See Also

The functions leftjoin, rightjoin, and fulljoin.

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

dorollapply *Rolling apply*

Description

Apply a function rolling over vector data.

Usage

```
dorollapply(vec, width, FUN)
```

Arguments

vec	The vector to apply the function on.
width	Width (or length) of the window.
FUN	The function to roll over the vector. Calls the function for all length pieces.

See Also

Other vector functions: extendvector(), findconstantchunks(), findconstant(), leaveonlychunks(), localmaxima(), localminimamaxima(), localminima(), makelength(), replaceNAelements(), replaceelements(), runavgweighed(), runavg(), runsd(), shift()

extendvector *Extend vector*

Description

Ensure a certain length of the vector is maintained, by cutting it short or extending it to the required length with a specific single value.

Usage

```
extendvector(vec, len, extval = NA)
```

Arguments

vec	The vector to set the length for.
len	The length the vector needs to be.
extval	The value to use in case the provided vec is shorter than the required len. Default=NA.

Value

The vector having the required lenght, extended if needed.

See Also

Other vector functions: dorollapply(), findconstantchunks(), findconstant(), leaveonlychunks(), localmaxima(), localminimamaxima(), localminima(), makelength(), replaceNAelements(), replaceelements(), runavgweighed(), runavg(), runsd(), shift()

extractcategories *Extract unique categories*

Description

Extract the unique categories/elements found in the dataset for a specific column.

Usage

```
extractcategories(data, datacolumn)
```

Arguments

data	The data.table in which all approaches are listed.
datacolumn	String containing the column from the data.table to extract from.

Value

Vector with all unique categories/elements.

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

Examples

```
extractcategories(data,"TREINNR") #returns all unique train numbers
```

findconstant *Find constant value*

Description

Within a numeric vector, find the locations of a window for which the values can be considered constant according to the provided standard deviation level settings.

Usage

```
findconstant(
  thevector,
  minpoints = 20,
  stdlevel = 0.05,
  desiredvalue = NULL,
  desiredvaluecloseness = 0.05
)
```

Arguments

thevector	Numeric vector to search in.
minpoints	Length of the window that requires to be constant. Default=20.
stdlevel	The maximum standard deviation of the window. Any standard deviation above this level will be considered not constant. Default=0.05.
desiredvalue	If provided, it will search for the vector windows for the closest desired average value with a standard deviation below the provided stdlevel. Default=NULL, which omits the search for a desired value.
desiredvaluecloseness	Value to determine if the provided desiredvalue is close enough to the value found. It will check if the window average is between desiredvalue-desiredvaluecloseness and desiredvalue+desiredvaluecloseness to be a match. Default=0.05.

Value

A list with the following items:

locations If desiredvalue is provided, the vector locations which result in the closest average desiredvalue with at least the standard deviation below stdlevel. Otherwise, the vector locations that provide the lowest standard deviation in the window, at least below the provided stdlevel.

average The calculated average for the vector locations.

standarddeviation The calculated standard deviation for the vector locations.

If no window below stdlevel is found, or if the desired average value is not found, or the closest desired average value is not below stdlevel, NULL returns.

See Also

Other vector functions: dorollapply(), extendvector(), findconstantchunks(), leaveonlychunks(), localmaxima(), localminimamaxima(), localminima(), makelength(), replaceNAelements(), replaceelements(), runavgweighed(), runavg(), runsd(), shift()

---

findconstantchunks *Find constant value chunks*

Description

Within a numeric vector, find all locations of a window for which the values can be considered constant according to the provided standard deviation level settings.

Usage

```
findconstantchunks(thevector, minpoints = 20, stdlevel = 0.05)
```

Arguments

thevector	Numeric vector to search in.
minpoints	Length of the window that requires to be constant. Default=20.
stdlevel	The maximum standard deviation of the window. Any standard deviation above this level will be considered not constant. Default=0.05.

Details

In contrast to the findconstant function, which either returns a constant window chunk for a provided desired value or the 'most constant' chunk (based on the lowest standard deviation), this function returns all possible chunks in a vector that can be considered constant.

Value

A data.table with one row for each chunk, with the following column names:

chunksize The amount of points in thevector that refers to this chunk.

chunkstartloc The very first location of thevector that refers to the start of this window chunk.

chunkendloc The very last location of thevector that refers to the end of this window chunk.

chunkaverage The average value of this chunk.

chunkstandarddeviation The standard deviation of this chunk (denominator *n*).

minstdstartloc The very first location of thevector that provide the lowest standard deviation in the chunk calculated for a window of size minpoints.

minstdendloc The very last location of thevector that provide the lowest standard deviation in the chunk calculated for a window of size minpoints.

minstdaverage The calculated average value for the vector locations that provide the lowest standard deviation in the chunk.

minstdstandarddeviation The standard deviation for the vector locations that provide the lowest standard deviation in the chunk (denominator *n*) calculated for a window of size minpoints.

If no window below stdlevel is found, NULL returns.

See Also

Other vector functions: dorollapply(), extendvector(), findconstant(), leaveonlychunks(), localmaxima(), localminimamaxima(), localminima(), makelength(), replaceNAelements(), replaceelements(), runavgweighed(), runavg(), runsd(), shift()

---

forcelisttype *Force all values to be of list type*

Description

Recursively go over the list and transform all values to be of list type.

Usage

```
forcelisttype(thelist)
```

Arguments

thelist	The list to traverse.

Value

List in which all values have been put into a list.

See Also

Other list functions: `leavenames()`, `listdepth()`, `stripnames()`

`fulljoin` *Full join data.tables*

Description

Execute a full join on two data.tables, meaning that all data of both data.tables will be kept.

Usage

```
fulljoin(
  basetable,
  additionaltable,
  basetablejoincolumns,
  additionaltablejoincolumns = NULL,
  additionaltableextractcolumns = NULL,
  removejoincols = NA
)
```

Arguments

`basetable`	The base data.table (if unmatched in `additionaltable`, `NA` values are added for this data.table).
`additionaltable`	The additional data.table (if unmatched in `basetable`, `NA` values are added for this data.table).
`basetablejoincolumns`	The column name string of the base data.table to join on.
`additionaltablejoincolumns`	The column name string of the additional data.table to join on. Default=`NULL`, which assumes the same column name as `basetablejoincolumns`.
`additionaltableextractcolumns`	The column name strings of the additional data.table to add to the base data.table. Default=`NULL`, which assumes that all columns need to be joined.
`removejoincols`	If `TRUE`, it will remove the provided `basetablejoincolumns` from the resulting table. If `FALSE`, it will keep the provided `basetablejoincolumns` in the result. If `NA` (default), it will check if `basetablejoincolumns` and `additionaltablejoincolumns` are the same. The column names that are different will result in the respective `basetablejoincolumns` to be removed. For example, let's take `basetablejoincolumns=c("ID","SECID")` and `additionaltablejoincolu` With `removejoincols=TRUE`, the column `"ID2"` will return in the result, but columns `"ID"` and `"SECID"` will not. With `removejoincols=FALSE`, the columns `"ID.x"`, `"ID.y"`, `"ID2"` and `"SECID"` will return. Note that column `ID` exists in both the base as well as additional table, which results in the `"ID.x"` and `"ID.y"` columns. With `removejoincols=NA` the column `"ID"` and `"ID2"` will return in the result, but column `"SECID"` will not.

Value

The joined data.table or, when join column(s) or extracted column(s) don't exist, the original base data.table.

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

---

getcolumnclasses *Get all column classes*

Description

Get all column classes

Usage

```
getcolumnclasses(thedata)
```

Arguments

thedata	The data.table to get column classes for

Value

Data.table with one row containing the column classes of each column.

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

getdefaulthovertext *Get hover text*

Description

Make the hover text for each data point in a graph. This is the default function, other specialized functions can be written for specific data. This function is provided as a variable in the plotsettings used in plotting.

Usage

```
getdefaulthovertext(plotdata, plotsettings, whichaxis = 0)
```

Arguments

plotdata	The data to use in the plot.
plotsettings	The settings to use in the plot (see getdefaultplotsettings).
whichaxis	The integer value that indicates the y axis from the plotsettings is used. Default=0, which does not specify a specific y axis.

Details

The function has two purposes. First, it provides the hover text in the graph for a specific axis. Second, it provides the text to display in an information field (if available).

Value

The hover text for each data point.

See Also

Other plot functions: getdefaultlabeltext(), getdefaultplotsettings(), makeplotsettings(), plot_makecolors(), plot_makemarkers(), plotcandlestick(), plotdensityfunctions(), plotecdffunctions(), plothistogram(), plotpiechart(), plotscatterwithlink(), plotscatter(), sampledashboard(), updateplotsettings()

getdefaultlabeltext *Get label text*

Description

Make the label text for each data point in a graph. This is the default function, other specialized functions can be written for specific data. This function is provided as a variable in the plotsettings used in plotting.

Usage

```
getdefaultlabeltext(plotdata, plotsettings, whichaxis = 0)
```

Arguments

plotdata	The data to use in the plot.
plotsettings	The settings to use in the plot (see getdefaultplotsettings).
whichaxis	The integer value that indicates the y axis from the plotsettings is used. Default=0, which does not specify a specific y axis.

Value

The label text for each data point.

See Also

Other plot functions: getdefaulthovertext(), getdefaultplotsettings(), makeplotsettings(), plot_makecolors(), plot_makemarkers(), plotcandlestick(), plotdensityfunctions(), plotecdffunctions(), plothistogram(), plotpiechart(), plotscatterwithlink(), plotscatter(), sampledashboard(), updateplotsettings()

---

getdefaultplotsettings *Plot settings*

Description

Get the default plotsettings for use in plotting functions. Settings contain the following elements:

xaxis This sets the column from the data.table to be used for the x axis. A string containing only one column name. Default="x".

xaxisrangemode This sets the x axis range setting. If "normal", all data is displayed. If "tozero", the zero axis is always shown. If "nonnegative", the x axis range never goes below zero. Default="normal"

yaxes This sets the columns from the data.table to display on the y axis. Can be more than one string containing the column names. If graphtype="candlestick" this needs to be of at least length 4, containing the column information for 'open', 'close', 'high', 'low', in that order. Any additional columns will be added as a regular scatter trace. If graphtype="piechart" only the first provided column is used. If graphtype="treemap" only the first provided column is used. Default=c("y").

yaxesvisible This sets the visibility settings for the y axes. Must be of same length as plotsettings$yaxes, and must be strings containing either YES, NO, or legendonly. Default=c("YES").

yaxeslinecolors This sets the main display colors for the y axes. Must be of same length as plotsettings$yaxes, and must be strings containing a color indication. If graphtype="scatter" this setting is used for line colors. If graphtype="scatterstack" this setting is used for line colors. Fill colors will be automatically made based on this color. If graphtype="histogram" this setting is used for bar colors. If graphtype="candlestick" this setting is omitted. If graphtype="piechart" this setting is omitted. If graphtype="treemap" this setting is omitted. Default=c("#6C9DDC").

yaxesmarkercolors This sets the columns from the data.table to use as data source for the color of the markers on the representative y axis. Must be of same length as plotsettings$yaxes and be more than one string containing the column names, NA where no markers are to be displayed, or "linecolor" if the marker needs to take the same color as the line. Works together with plotsettings$markercolorsettings, where the data in those columns are associated with colors. If graphtype="scatter" this setting is used for marker colors. If graphtype="scatterstack" this setting is used for marker colors. If graphtype="histogram" this setting is used omitted. If graphtype="candlestick" this setting is omitted. If graphtype="piechart" this setting is used for pie colors. If graphtype="treemap" this setting is used for block colors. Default=c("linecolor").

yaxesmarkersymbols This sets the columns from the data.table to use as data source for the symbols of the markers on the representative y axis. Must be of same length as plotsettings$yaxes. Can be more than one string containing the column names, or NA where no symbols are to be displayed. Works together with plotsettings$markersymbolsettings, where the specific symbols for columns are associated. If graphtype="scatter" this setting is used for marker colors. If graphtype="scatterstack" this setting is used for marker colors. If graphtype="histogram" this setting is used omitted. If graphtype="candlestick" this setting is omitted. If graphtype="piechart" this setting is omitted. If graphtype="treemap" this setting is omitted. Default=c(NA).

yaxesmarkersizes This sets the size of the marker (if displayed). Default=8. Can be a single value, making all markers equal in size, or the same length as plotsettings$yaxes allowing each y axis to have a different size marker. Missing values will use the first provided marker size.

yaxesnames This sets the name of the y axes, as to be displayed in the legend. Also allows legend grouping by name. Must be of same length as plotsettings$yaxes, and must be strings containing the name or NA to use the same name as provided in plotsettings$yaxes. Default=c(NA).

yaxisrangemode This sets the y axis range setting. If "normal", all data is displayed. If "tozero", the zero axis is always shown. If "nonnegative", the y axis range never goes below zero. Default="normal". This setting works globally, and not for each y axis individually.

displaytitle This sets the graph title to a specified text. Must be a string or NULL to display no title. Default=NULL.

displaylegend This sets the visibility of a legend in the graph (either TRUE or FALSE). Default=TRUE.

graphtype This sets the graph type. Currently supported are "scatter" (will call plotscatter), "scatterstack" (will call plotscatter), "histogram" (will call plothistogram), "candlestick" (will call plotcandlestick), "piechart" (will call plotpiechart), and "treemap" (will call plottreemap). Checks for graphtype happen in the dashboard, not in the plot- functions. Default="scatter".

graphmode This sets the mode of the graph, and is dependent on what graphtype is chosen. Default=NA. If graphtype="scatter" this setting can be a vector with the same length as yaxes and will be either "lines", "lines+markers" or "markers" for the specific y axis data to be plotted in that mode. If the length is insufficient, it will default to the NA setting for that y axis. When NA is provided, it will use the "lines+markers" setting. If graphtype="scatterstack" this setting is the same as graphtype="scatter" but "none" can be added to hide the lines. When NA is provided, it will use the "none" setting. If graphtype="histogram" this setting is used for the barmode setting of the plot layout. Can be either "group", "stack" or "overlay". When NA is provided, it will use the "group" setting. If graphtype="candlestick" this setting can be a vector with the same length as yaxes and will be either "lines", "lines+markers" or "markers" for the specific y axis data to be plotted in that mode. If the length is insufficient, it will default to the NA setting for that y axis.

When NA is provided, it will use the "lines+markers" setting. If graphtype="piechart" this setting is omitted. If graphtype="treemap" this setting is omitted.

markercolorsettings This associates the column names of the data.table with the content in that column, and the color to use. The association of data with colors only happens when yaxesmarkercolors is provided. Consists of a list of list, in which the interior list contains:

columnnames String vector of data.table column name that contains the data to be mapped.

columncontent Vector of discrete data points to be found in this column of the data.table. If only two values are provided and they are numeric, then they are assumed to be the minimum and maximum value for a color ramp to be generated. Leaving this list item empty (NULL or NA) then a color ramp is automatically assumed, mapping the minimum and maximum value of the data.

columncolors The color to use for each specific data point, or the various colors (minimum 2) of the color ramp to generate. To map discrete data points this list item must be of the same length as columncontent and can be a color name, a hex color, or "linecolor" (which takes the same color as the line, specified in plotsettings$yaxeslinecolors).

The default is a list too long to report here, but can be found by simply executing this function.

markersymbolsettings This associates the column names of the data.table with the content in that column, and the symbol to use. The association of data with colors only happens when yaxesmarkersymbols is provided. Consists of a list of list, in which the interior list contains:

columnnames String vector of data.table column name that contains the data to be mapped.

columncontent Vector of discrete data points to be found in this column of the data.table.

columnsymbols The symbol to use for each specific data point. Must be of the same length as columncontent.

datafix_keepmissingvalues This informs the plotting functions what to do with missing values. If TRUE, missing values are kept and can cause gaps in the display. If FALSE, missing values are removed which will create a continuous display. Default=TRUE.

function_hovertext This informs the plotting functions which function to use to create the hover text format. Set to NULL or to a function that returns NULL to not display any hover text. Default=getdefaulthovertext.

function_labeltext This informs the plotting functions which function to use to create the label text format. Set to NULL or to a function that returns NULL to not display any label text. Default=getdefaultlabeltext, which simply returns NULL to avoid default label display.

labelposition This sets the positioning method of the label text. If graphtype="scatter" or graphtype="scatterstack" or graphtype="treemap" this setting can be a combination of "top", "middle" or "bottom" with "left", "center" or "right". If graphtype="histogram" or graphtype="piechart" this setting can be either of "inside", "outside", "auto" or "none". If graphtype="candlestick" this setting is only used for the additional scatter traces that follow the candlestick traces. Default=NA, which takes either "middle center" or "auto" depending on graphtype.

Usage

```
getdefaultplotsettings()
```

Value

List with default plot settings.

See Also

Other plot functions: getdefaulthovertext(), getdefaultlabeltext(), makeplotsettings(), plot_makecolors(), plot_makemarkers(), plotcandlestick(), plotdensityfunctions(), plotecdffunctions(), plothistogram(), plotpiechart(), plotscatterwithlink(), plotscatter(), sampledashboard(), updateplotsettings()

---

getfileextensions *Get file extension*

Description

Get the file extension from a full path or filename.

Usage

```
getfileextensions(fullpaths)
```

Arguments

fullpaths	The full paths or file name to get extensions from.

Value

List of extensions.

See Also

Other file functions: getfilenames(), getfiletable()

---

getfilenames *Get file name*

Description

Get the file name from a full path.

Usage

```
getfilenames(fullpaths, excludeextension = FALSE)
```

Arguments

fullpaths	The full paths to get file names from.
excludeextension	Set to TRUE to exclude the extension from the returned filenames.. Default=FALSE, which includes the extension.

Value

List of file names.

See Also

Other file functions: `getfileextensions()`, `getfiletable()`

`getfiletable` *Get directory file list*

Description

Get a list of files in a specific directory.

Usage

```
getfiletable(
  dirtocheck,
  fileextension = NULL,
  includesubdirs = FALSE,
  printmessage = FALSE
)
```

Arguments

`dirtocheck`	The directory to obtain the files from.
`fileextension`	One of more strings specifying the extensions (without dot, such as `"zip"`) of the files to return. Default=`NULL`, which returns all files.
`includesubdirs`	Set to `TRUE` to also include the files in all subdirectories. Default=`FALSE`, which only checks the cirrent directory.
`printmessage`	Set to `TRUE` to display the amount of files found in the directory in a message. Default=`FALSE`, which does not print a message.

Value

Data.table with the files, containing the columns FULLPATH (full non-relative path), DIRONLY (directory of the file), FILENAME (file name without extension), and EXTENSION (file extension).

See Also

Other file functions: `getfileextensions()`, `getfilenames()`

gethuecolors *Generate hue colors*

Description

Generate a range of hues (colors) for graphs

Usage

```
gethuecolors(n)
```

Arguments

n	The amount of colors to generate

Value

Vector containing n amount of colors

See Also

Other plot assists: `getpalettecolors()`

getpalettecolors *Generate palette colors*

Description

Generate a range of palette based colors. Uses the 70 unique colors from the RColorBrewer library and recycles them to match the number of observations.

Usage

```
getpalettecolors(observations)
```

Arguments

observations	character vector or number vector

Value

Data.table with the columns `OBSERVATIONS` and `COLOR`.

See Also

Other plot assists: `gethuecolors()`

Examples

```
observations =  1:100
DT_test = merge(
    x=data.table(BLABLA = observations),
    y=ut_create_colors(observations),
    by.x="BLABLA",
    by.y="OBSERVATIONS"
  )

  table(
    data.table(
      table(DT_test$COLOR)
    )[order(-N)]$N
  )
```

`hextobinary` *Hex to binary*

Description

Transform hexadecimal values (in string vector) to binary bits (also in string vector)

Usage

```
hextobinary(hexvalues)
```

Arguments

`hexvalues`	String vector containing hexadecimal values.

Value

String vector containing the related binary bits, or NA where the provided string is invalid (not a hex value representation).

See Also

Other conversion functions: `bitsinbinary()`, `convertcoords_degrees2decimal()`

Examples

```
hexdat <- replicate(10, paste(format.hexmode(sample(16,4)-1),collapse=''))
hextobinary(hexdat)
```

inbetween *In between*

Description

Check if a value is in between minimum and maximum value

Usage

```
inbetween(value, minval, maxval)
```

Arguments

value	The value to check
minval	The minimum value of the range
maxval	The maximum value of the range

Value

TRUE if the value is in between the range (minimum and maximum values included), FALSE if this is not the case.

innerjoin *Inner join data.tables*

Description

Execute an inner join on two data.tables, meaning that only data present in both data.tables will be kept.

Usage

```
innerjoin(
  basetable,
  additionaltable,
  basetablejoincolumns,
  additionaltablejoincolumns = NULL,
  additionaltableextractcolumns = NULL,
  removejoincols = NA
)
```

Arguments

basetable	The base data.table.
additionaltable	The additional data.table.
basetablejoincolumns	The column name string of the base data.table to join on.

additionaltablejoincolumns
: The column name string of the additional data.table to join on. Default=NULL, which assumes the same column name as basetablejoincolumns.

additionaltableextractcolumns
: The column name strings of the additional data.table to add to the base data.table. Default=NULL, which assumes that all columns need to be joined.

removejoincols
: If TRUE, it will remove the provided basetablejoincolumns from the resulting table. If FALSE, it will keep the provided basetablejoincolumns in the result. If NA (default), it will check if basetablejoincolumns and additionaltablejoincolumns are the same. The column names that are different will result in the respective basetablejoincolumns to be removed.

  For example, let's take basetablejoincolumns=c("ID","SECID") and additionaltablejoincolu
  With removejoincols=TRUE, the column "ID2" will return in the result, but columns "ID" and "SECID" will not. With removejoincols=FALSE, the columns "ID.x", "ID.y", "ID2" and "SECID" will return. Note that column ID exists in both the base as well as additional table, which results in the "ID.x" and "ID.y" columns. With removejoincols=NA the column "ID" and "ID2" will return in the result, but column "SECID" will not.

Value

The joined data.table or, when join column(s) or extracted column(s) don't exist, the original base data.table.

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

---

leavenames *Leave list with only names*

Description

Removes all named elements except the provided names from a list. If none of the provided names are found within the list, the full list is kept (therefore keeping the list structure). Executes recursively so that it also works within lists in lists.

Usage

```
leavenames(thelist, thenames)
```

Arguments

thelist	List to only leave the provided names in.
thenames	Vector of strings containing the names of the elements to leave in the list.

Value

List with only the named elements remaining.

See Also

Other list functions: `forcelisttype()`, `listdepth()`, `stripnames()`

`leaveonlychunks` *Leave chunks*

Description

For a vector, leave only those elements where a certain value occurs for at least a certain amount of time in a chunk.

Usage

```
leaveonlychunks(vals, chunksize, replaceval = NA)
```

Arguments

`vals`	Vector to check for chunks.
`chunksize`	Minimum size of a chunk.
`replaceval`	Replace values of too small chunks in the vector with this value. Default=`NA`, which replaces all chunks that are too small with `NA`. If `NULL`, it will automatically check the class of the vector, and uses `FALSE` for booleans and `0` for numeric or integer vectors.

Value

A vector where the length of chunks having the same value is larger than the provided `chunksize`. Too small chunks have been replaced with the `replaceval`.

See Also

Other vector functions: `dorollapply()`, `extendvector()`, `findconstantchunks()`, `findconstant()`, `localmaxima()`, `localminimamaxima()`, `localminima()`, `makelength()`, `replaceNAelements()`, `replaceelements()`, `runavgweighed()`, `runavg()`, `runsd()`, `shift()`

leftjoin *Left join data.tables*

Description

Execute a left join on two data.tables, meaning that all data of the left data.table (basetable) will be kept, and only the data from the right data.table (additionaltable) is linked to where it can be found in the left data.table.

Usage

```
leftjoin(
  basetable,
  additionaltable,
  basetablejoincolumns,
  additionaltablejoincolumns = NULL,
  additionaltableextractcolumns = NULL,
  removejoincols = NA
)
```

Arguments

basetable	The base data.table
additionaltable	The additional data.table (if unmatched in basetable, NA values are added for this data.table).
basetablejoincolumns	The column name string of the base data.table to join on.
additionaltablejoincolumns	The column name string of the additional data.table to join on. Default=NULL, which assumes the same column name as basetablejoincolumns.
additionaltableextractcolumns	The column name strings of the additional data.table to add to the base data.table. Default=NULL, which assumes that all columns need to be joined.
removejoincols	If TRUE, it will remove the provided basetablejoincolumns from the resulting table. If FALSE, it will keep the provided basetablejoincolumns in the result. If NA (default), it will check if basetablejoincolumns and additionaltablejoincolumns are the same. The column names that are different will result in the respective basetablejoincolumns to be removed. For example, let's take basetablejoincolumns=c("ID","SECID") and additionaltablejoincolu With removejoincols=TRUE, the column "ID2" will return in the result, but columns "ID" and "SECID" will not. With removejoincols=FALSE, the columns "ID.x", "ID.y", "ID2" and "SECID" will return. Note that column ID exists in both the base as well as additional table, which results in the "ID.x" and "ID.y" columns. With removejoincols=NA the column "ID" and "ID2" will return in the result, but column "SECID" will not.

Value

The joined data.table or, when join column(s) or extracted column(s) don't exist, the original base data.table.

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

---

linearcalc *Linear interpolation with normalization*

Description

Perform linear interpolation and normalization in between a minimum and maximum value. The minimum value and below represent 0, the maximum value and above represent 1, unless inverted=TRUE, which reverses this logic.

Usage

```
linearcalc(inputvector, minval, maxval, inverted = FALSE)
```

Arguments

inputvector	The values to calculate on.
minval	The minimum value.
maxval	The maximum value.
inverted	Set to TRUE to invert the result, so that the maximum value and above represent 0 and the minimum value and below represent 1 (default=FALSE, which uses the minimum value and below to represent 0 and the maximum value and above to represent 1).

See Also

Other predictors: linearextrapolate()

---

linearextrapolate *Linear extrapolation*

Description

Linear extrapolation

Usage

```
linearextrapolate(x, y, xi)
```

Arguments

x	The values on the x axis.
y	Corresponding values on the y axis.
xi	The extrapolated x axis value to calculate a new y value for.

Value

The calculated extrapolated y value.

See Also

Other predictors: `linearcalc()`

---

listdepth *Find list depth*

Description

Find the depth of a list, which is an integer value of the furthest recursion into the lists in lists.

Usage

```
listdepth(thelist, thisdepth = 0)
```

Arguments

thelist	List to find the depth for.
thisdepth	Used for recursion to find the depth but can be modified if you provide a list within list already. Default=0.

Value

Integer that identifies the depth of a list.

See Also

Other list functions: `forcelisttype()`, `leavenames()`, `stripnames()`

loaddependentpackages *Load dependencies*

Description

For a development package, only load the dependencies.

Usage

```
loaddependentpackages(dir)
```

Arguments

dir	The directory of the development package.

See Also

Other loading functions: loaddevelopmentlibrary(), loadlargetable(), loadlibrary(), loadsastable(), loadsource(), loadspecifictable(), loadtable()

loaddevelopmentlibrary *Load a development library*

Description

Load a specific named development library.

Usage

```
loaddevelopmentlibrary(x, musthaveversion = "0.0.0")
```

Arguments

x	Character string of the library to load.
musthaveversion	The version that the package must have (default="0.0.0")

See Also

Other loading functions: loaddependentpackages(), loadlargetable(), loadlibrary(), loadsastable(), loadsource(), loadspecifictable(), loadtable()

`loadlargetable` *Load a large table*

Description

Load a large SAS or RDS data table from file, with an automatic RDS existance check, and return it as a data.table. If a SAS table is loaded, it will automatically save it as RDS in the same directory for future faster loading.

Usage

```
loadlargetable(dir, fname, forcesas = FALSE)
```

Arguments

`dir`	Directory in which the file can be found. Must end with '/'.
`fname`	File name of the file (without extension).
`forcesas`	Set to TRUE if the original SAS table needs to be loaded (it otherwise uses the quicker rds file).

Details

Depending on the size of the SAS table, this function can take a while to finish. It does not have a progress indicator.

Value

Data.table on success, NULL on failure.

See Also

Other loading functions: `loaddependentpackages()`, `loaddevelopmentlibrary()`, `loadlibrary()`, `loadsastable()`, `loadsource()`, `loadspecifictable()`, `loadtable()`

Other data.table functions: `addcolumn()`, `addisoinfo()`, `copytoclipboard()`, `dojoin()`, `extractcategories()`, `fulljoin()`, `getcolumnclasses()`, `innerjoin()`, `leftjoin()`, `loadsastable()`, `loadspecifictable()`, `loadtable()`, `makedatecolumns()`, `makedatetimecolumns()`, `makeintegercolumns()`, `makenumericcolumns()`, `mapcolumnclasses()`, `prefixcolnames()`, `removeNAfromtable()`, `removeallNAcolumns()`, `removeallNArows()`, `removecolumns()`, `removeisoinfo()`, `removerows()`, `renamecolumns()`, `rightjoin()`, `runcolfun()`, `selectuniques()`, `selectwherein()`, `selectwhere()`, `setcolclass()`, `setcolnames()`, `setcoltofront()`, `setcolumns()`, `unlistcolumns()`, `updatetablevalues()`

loadlibrary *Load a library*

Description

Load a specific named library. If it is not yet installed, it will install it.

Usage

```
loadlibrary(x)
```

Arguments

x	Character string of the library to load.

See Also

Other loading functions: loaddependentpackages(), loaddevelopmentlibrary(), loadlargetable(), loadsastable(), loadsource(), loadspecifictable(), loadtable()

loadsastable *Load a SAS data table*

Description

Load a SAS data table from file and return it as a data.table.

Usage

```
loadsastable(dir, fname)
```

Arguments

dir	Directory in which the file can be found. Must end with '/'.
fname	File name of the file (without extension).

Details

Depending on the size of the SAS table, this function can take a while to finish. It does not have a progress indicator.

Value

Data.table on success, NULL on failure.

See Also

Other loading functions: loaddependentpackages(), loaddevelopmentlibrary(), loadlargetable(), loadlibrary(), loadsource(), loadspecifictable(), loadtable()

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

---

loadsource *Load source files*

Description

Load all source files in a specific directory. Can be used to source a package, allowing direct package editing while running it. Of course, after code modification the package needs to be built and installed again.

Usage

```
loadsource(dir, openeditor = FALSE, loaddependencies = FALSE)
```

Arguments

dir	The directory to load the source files from.
openeditor	Set to TRUE to open the source files in the editor. In RStudio, this allows debugging of the functions through automatic error breakpoints or using the F2 function lookup shortcut followed by direct editing. Default=FALSE, which only sources the files without opening an editor.
loaddependencies	Set to TRUE to load all libraries that the source package depends on. This allows for the loading of the source plus all dependent libraries, without actually calling library of the actual package. This allows for setting breakpoints and debugging the opened source code without RStudio complaining that an updated package is needed. Default=FALSE, which does not call dependent libraries.

See Also

Other loading functions: loaddependentpackages(), loaddevelopmentlibrary(), loadlargetable(), loadlibrary(), loadsastable(), loadspecifictable(), loadtable()

`loadspecifictable` *Load a RDS, CSV, XLSX or SAS file*

Description

Loads the file and returns it as a data.table. The loading priority follows RDS, then CSV, then XLSX, then SAS.

Usage

```
loadspecifictable(dir, fname, filetype = "RDS", largetable = FALSE)
```

Arguments

`dir`	Directory in which the file can be found. If it does not end with '/', this directory separator will be added.
`fname`	File name of the file (without extension).
`filetype`	The file type of the file to force load ("RDS", "CSV", "XLSX", or "SAS"). Default = "RDS". Setting this to a either "CSV", "XLSX" or "SAS" forces a reload.
`largetable`	Set to `TRUE` if this is a large table, which stores the table to RDS after loading. Default=`FALSE`, which does not store the result in a RDS file.

Details

Depending on the size of the file, this function can take a while to finish. It does not have a progress indicator.

Value

Data.table on success, NULL on failure.

See Also

Other loading functions: `loaddependentpackages()`, `loaddevelopmentlibrary()`, `loadlargetable()`, `loadlibrary()`, `loadsastable()`, `loadsource()`, `loadtable()`

Other data.table functions: `addcolumn()`, `addisoinfo()`, `copytoclipboard()`, `dojoin()`, `extractcategories()`, `fulljoin()`, `getcolumnclasses()`, `innerjoin()`, `leftjoin()`, `loadlargetable()`, `loadsastable()`, `loadtable()`, `makedatecolumns()`, `makedatetimecolumns()`, `makeintegercolumns()`, `makenumericcolumns()`, `mapcolumnclasses()`, `prefixcolnames()`, `removeNAfromtable()`, `removeallNAcolumns()`, `removeallNArows()`, `removecolumns()`, `removeisoinfo()`, `removerows()`, `renamecolumns()`, `rightjoin()`, `runcolfun()`, `selectuniques()`, `selectwherein()`, `selectwhere()`, `setcolclass()`, `setcolnames()`, `setcoltofront()`, `setcolumns()`, `unlistcolumns()`, `updatetablevalues()`

`loadtable` *Load a tabular file*

Description

Loads a tabular file and returns it as a data.table. Automatically loads the most efficient type first if available. It can read the following file types: RDS, CSV, XLSX, SAS

Usage

```
loadtable(
  dir,
  fname,
  filetype = "RDS",
  quickstore = TRUE,
  forceallsheets = FALSE,
  ...
)
```

Arguments

`dir`	Directory in which the file can be found. If it does not end with '/', this directory separator will be added.
`fname`	File name of the file (without extension).
`quickstore`	Set to `FALSE` to NOT store the data to .RDS after loading. Default=`TRUE`, which does store an RDS file that makes future loading faster.
`forceallsheets`	Set to `TRUE` if all sheets of an .XLSX file need to be loaded, without asking for user input.

Value

Data.table on success, `NULL` on failure. It will return a list of data.tables in case an .xlsx with multiple sheets is selected to read.

See Also

Other loading functions: `loaddependentpackages()`, `loaddevelopmentlibrary()`, `loadlargetable()`, `loadlibrary()`, `loadsastable()`, `loadsource()`, `loadspecifictable()`

Other data.table functions: `addcolumn()`, `addisoinfo()`, `copytoclipboard()`, `dojoin()`, `extractcategories()`, `fulljoin()`, `getcolumnclasses()`, `innerjoin()`, `leftjoin()`, `loadlargetable()`, `loadsastable()`, `loadspecifictable()`, `makedatecolumns()`, `makedatetimecolumns()`, `makeintegercolumns()`, `makenumericcolumns()`, `mapcolumnclasses()`, `prefixcolnames()`, `removeNAfromtable()`, `removeallNAcolumns()`, `removeallNArows()`, `removecolumns()`, `removeisoinfo()`, `removerows()`, `renamecolumns()`, `rightjoin()`, `runcolfun()`, `selectuniques()`, `selectwherein()`, `selectwhere()`, `setcolclass()`, `setcolnames()`, `setcoltofront()`, `setcolumns()`, `unlistcolumns()`, `updatetablevalues()`

localmaxima *Get local maxima*

Description

Get local maxima

Usage

```
localmaxima(x)
```

Arguments

x	Vector of values to get the local maxima locations of.

Value

The vector locations (like the which function) where local maxima occur.

See Also

Other vector functions: dorollapply(), extendvector(), findconstantchunks(), findconstant(), leaveonlychunks(), localminimamaxima(), localminima(), makelength(), replaceNAelements(), replaceelements(), runavgweighed(), runavg(), runsd(), shift()

Other data functions: createbins(), localminimamaxima(), localminima(), removeoffset(), timeseriesstats()

localminima *Get local minima*

Description

Get local minima

Usage

```
localminima(x)
```

Arguments

x	Vector of values to get the local minima locations of.

Value

The vector locations (like the which function) where local minima occur.

See Also

Other vector functions: dorollapply(), extendvector(), findconstantchunks(), findconstant(), leaveonlychunks(), localmaxima(), localminimamaxima(), makelength(), replaceNAelements(), replaceelements(), runavgweighed(), runavg(), runsd(), shift()

Other data functions: createbins(), localmaxima(), localminimamaxima(), removeoffset(), timeseriesstats()

`localminimamaxima` *Get local minima and maxima*

Description

Get local minima and maxima

Usage

```
localminimamaxima(x, threshold = 1)
```

Arguments

x	Vector of values to get the local minima and maxima locations of.
threshold	Integer threshold value to get wider spaced minima and maxima the higher threshold is. Default = 1.

Value

Named list with vector locations (like the `which` function) where (local) minima and maxima occur.

See Also

Other vector functions: dorollapply(), extendvector(), findconstantchunks(), findconstant(), leaveonlychunks(), localmaxima(), localminima(), makelength(), replaceNAelements(), replaceelements(), runavgweighed(), runavg(), runsd(), shift()

Other data functions: createbins(), localmaxima(), localminima(), removeoffset(), timeseriesstats()

`makeconsistentdateformat` *Fix dates*

Description

Make a consistent date format from a provided variable. Has internal class detection so can be called on already properly formatted dates (POSIXt) too. For date and time use makeconsistentdatetimeformat function.

Usage

```
makeconsistentdateformat(thedates, asstring = NULL, timezone = "")
```

Arguments

thedates	Vector with dates to convert. Can work with POSIXt, Date, numeric and string. If numeric is provided, it assumes Unix timestamp in seconds. If NULL is provided it returns NULL too.
asstring	A string format (following strptime) to convert the date to. Default=NULL, which returns the date in POSIXct format.
timezone	String determining the timezone (from as.POSIXct). Default="", which uses the local time zone.

Value

The dates in POSIXct format with the provided timezone, or as a string through the provided format, or NULL if the provided input is incorrect.

See Also

Other date and time functions: addisoinfo(), makeconsistentdatetimeformat(), makedatecolumns(), makedatetimecolumns(), removeisoinfo(), seqdays()

---

makeconsistentdatetimeformat

Fix datetimes

Description

Make a consistent date format from a provided variable. Has internal class detection so can be called on already properly formatted dates (POSIXt) too. For date only use makeconsistentdateformat function.

Usage

```
makeconsistentdatetimeformat(thedates, asstring = NULL, timezone = "")
```

Arguments

thedates	Vector with string datetimes to convert. Can work with POSIXt, Date, numeric and string. If numeric is provided, it assumes Unix timestamp in seconds. If NULL is provided it returns NULL too.
asstring	A string format (uses strftime) to convert the datetime to (default=NULL, which returns the date in POSIX format).
timezone	String determining the timezone (from as.POSIXct). Default="", which uses the local time zone.

Value

The datetimes in POSIXct format with the provided timezone, or as a string through the provided format, or NULL if the provided input is incorrect.

See Also

Other date and time functions: addisoinfo(), makeconsistentdateformat(), makedatecolumns(), makedatetimecolumns(), removeisoinfo(), seqdays()

makecoordinatecolumns *Make columns GPS coordinates*

Description

Make specific columns from a data.table numeric GPS coordinates, if they are of character class.

Usage

```
makecoordinatecolumns(
  thedata,
  thecolnames = NULL,
  degreestodecimal = FALSE,
  divisionfactor = 1
)
```

Arguments

thedata	The data.table to modify.
thecolnames	Column names to modify to numeric GPS coordinates. If commas are present, they are replaced by dots (to represent decimals). Default=NULL, which transforms all columns that it detects to be of class character.
degreestodecimal	Set to TRUE to assume the existing coordinates are in degrees, and need to be converted to decimals. Default=FALSE, which assumes the existing coordinates to be decimals already.
divisionfactor	Provide a value to divide the coordinates by. In certain situations, the provided coordinates have been multiplied by this factor to be stored more efficiently. Default=1, which assumes the coordinates have been stored without multiplication.

Value

Returns the modified data.table invisibly (modified by reference).

makedatecolumns *Make columns date*

Description

Make specific columns from a data.table into date format, either POSIXct with timezone Europe/Berlin, or a string format, when the current column class is of character.

Usage

```
makedatecolumns(thedata, thecolnames, stringformat = NULL, timezone = "")
```

Arguments

thedata	The data.table to modify.
thecolnames	Column names to modify to date.
stringformat	A string format (following strptime) to convert the date to. Default=NULL, which sets the column to POSIXct.
timezone	String determining the timezone (from as.POSIXct). Default="", which uses the local time zone.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

Other date and time functions: addisoinfo(), makeconsistentdateformat(), makeconsistentdatetimeformat(), makedatetimecolumns(), removeisoinfo(), seqdays()

---

makedatetimecolumns *Make columns datetime*

Description

Make specific columns from a data.table into datetime format, either POSIXct with timezone Europe/Berlin, or a string format.

Usage

```
makedatetimecolumns(thedata, thecolnames, stringformat = NULL, timezone = "")
```

Arguments

thedata	The data.table to modify.
thecolnames	Column names to modify to date.
stringformat	A string format (following strptime) to convert the date to. Default=NULL, which sets the column to POSIXct.
timezone	String determining the timezone (from as.POSIXct). Default="", which uses the local time zone.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

Other date and time functions: addisoinfo(), makeconsistentdateformat(), makeconsistentdatetimeformat(), makedatecolumns(), removeisoinfo(), seqdays()

`makeintegercolumns` *Make columns integer*

Description

Make specific columns from a data.table integer, if they are of character class.

Usage

```
makeintegercolumns(thedata, thecolnames = NULL)
```

Arguments

`thedata`	The data.table to modify.
`thecolnames`	Column names to modify to integer. Default=`NULL`, which transforms all columns that it detects to be of class character.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

makelength *Extend vector*

Description

Ensure a certain length of the vector is maintained, by cutting it short or extending it to the required length with a specific single value.

Usage

```
makelength(vec, len, extval = NA)
```

Arguments

vec	The vector to set the length for.
len	The length the vector needs to be.
extval	The value to use in case the provided vec is shorter than the required len. Default=NA.

Value

The vector having the required lenght, extended if needed.

See Also

Other vector functions: dorollapply(), extendvector(), findconstantchunks(), findconstant(), leaveonlychunks(), localmaxima(), localminimamaxima(), localminima(), replaceNAelements(), replaceelements(), runavgweighed(), runavg(), runsd(), shift()

makenumericcolumns *Make columns numeric*

Description

Make specific columns from a data.table numeric, if they are of character class.

Usage

```
makenumericcolumns(thedata, thecolnames = NULL)
```

Arguments

thedata	The data.table to modify.
thecolnames	Column names to modify to numeric. If commas are present, they are replaced by dots (to represent decimals). Default=NULL, which transforms all columns that it detects to be of class character.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

---

makeplotsettings *Make plotsettings*

Description

Simplified function to create automatic plotsettings for a range of yaxes names.

Usage

```
makeplotsettings(yaxesnames, sethuecolors = TRUE)
```

Arguments

yaxesnames	Vector with strings representing columns to display on the y axes.
sethuecolors	Set to FALSE to keep each y axis in the default color. Default=TRUE, which makes a unique line color for each y axis.

Value

List with default plot settings as returned from getdefaultplotsettings, but with the default settings applied to more than one y axis.

See Also

Other plot functions: getdefaulthovertext(), getdefaultlabeltext(), getdefaultplotsettings(), plot_makecolors(), plot_makemarkers(), plotcandlestick(), plotdensityfunctions(), plotecdffunctions(), plothistogram(), plotpiechart(), plotscatterwithlink(), plotscatter(), sampledashboard(), updateplotsettings()

---

mapcolumnclasses *Map column classes*

Description

Ensure columns of a data.table have the right class and if not, transform those columns.

Usage

```
mapcolumnclasses(thedata, thecolumns, theclasses, ...)
```

Arguments

thedata	The data.table to modify.
thecolumns	Vector string of column names of the data.table that need to have a specific class.
theclasses	Vector string of column classes that they need to be. Need to be of same length as thecolumns. If the strings "date" or "datetime" are provided but the columns are not yet in POSIXct, the functions makedatecolumns and makedatetimecolumns are called respectively.
...	Additional parameters of the makedatecolumns and makedatetimecolumns functions, if needed.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

---

paste0collapse *String collapse*

Description

Collapse a string with a provided collapse string. Different from paste in that the collapse string character is first.

Usage

```
paste0collapse(collapsestring, ...)
```

Arguments

collapsestring	The string to connect all items.

See Also

Other string functions: str_pad_month()

plotcandlestick *Plot a candlestick chart*

Description

Plots a candlestick chart with columns from a data.table. For candlesticks to display, four yaxes columns are needed. The first four in `plotsettings$yaxes` represent the open, close, high, and low data (in that order). Any further yaxes will be displayed based on the representative `plotsettings$graphmode`, or its default as lines.

Usage

```
plotcandlestick(
  datatable,
  plotsettings = NULL,
  returnplotly = FALSE,
  plotlysource = "A"
)
```

Arguments

`datatable`	The data.table in which the data is stored to plot.
`plotsettings`	The settings to use in the plot. Default=`NULL`, which loads default plotsettings from `getdefaultplotsettings`.
`returnplotly`	Set to `TRUE` to return the plotly variable to be used or modified in another setting (such as in a dashboard). Default=`FALSE`, which displays the plot.
`plotlysource`	Source attribute as defined in `plot_ly`). Default=`"A"`.

Value

The plotly variable if `returnplotly=TRUE` or an invisible `NULL`. It will always return `NULL` and print a message if the plotsettings do not match the data.

See Also

Other plot functions: `getdefaulthovertext()`, `getdefaultlabeltext()`, `getdefaultplotsettings()`, `makeplotsettings()`, `plot_makecolors()`, `plot_makemarkers()`, `plotdensityfunctions()`, `plotecdffunctions()`, `plothistogram()`, `plotpiechart()`, `plotscatterwithlink()`, `plotscatter()`, `sampledashboard()`, `updateplotsettings()`

plotdensityfunctions *Plot density functions*

Description

Plot density functions based on data from a data.table. Simplified plotting function, without the use of any plotsettings from `getdefaultplotsettings`. Uses the `density` function.

Usage

```
plotdensityfunctions(
  datatable,
  columns = NULL,
  returnplotly = FALSE,
  withlegend = TRUE
)
```

Arguments

datatable	The data.table with the data to plot density functions from.
columns	The columns to plot density functions for. Default=NULL, which plots the density functions for all columns.

Value

Nothing, just plots the density functions.

See Also

Other plot functions: getdefaulthovertext(), getdefaultlabeltext(), getdefaultplotsettings(), makeplotsettings(), plot_makecolors(), plot_makemarkers(), plotcandlestick(), plotecdffunctions(), plothistogram(), plotpiechart(), plotscatterwithlink(), plotscatter(), sampledashboard(), updateplotsettings()

---

plotecdffunctions *Plot ecdf functions*

Description

Plot empirical cumulative distribution functions based on data from a data.table. Uses the ecdf function. Simplified plotting function, without the use of any plotsettings from getdefaultplotsettings.

Usage

```
plotecdffunctions(
  datatable,
  columns = NULL,
  quantilerange = c(0, 1),
  inpercentage = FALSE,
  returnplotly = FALSE,
  withlegend = TRUE
)
```

Arguments

datatable	The data.table with the data to plot ecdf functions from.
columns	The columns to plot ecdf functions for. Default=NULL, which plots the density functions for all columns.
quantilerange	The range to plot, between 0 and 1, at least two values. Default=c(0,1), which plots the full range.

inpercentage	Set to TRUE to display percentages between 0 and 100. Default=FALSE, which displays factors between 0 and 1.

Value

Nothing, just plots the ecdf functions.

See Also

Other plot functions: getdefaulthovertext(), getdefaultlabeltext(), getdefaultplotsettings(), makeplotsettings(), plot_makecolors(), plot_makemarkers(), plotcandlestick(), plotdensityfunctions(), plothistogram(), plotpiechart(), plotscatterwithlink(), plotscatter(), sampledashboard(), updateplotsettings()

---

plothistogram *Plot a histogram*

Description

Plots a histogram with columns from a data.table.

Usage

```
plothistogram(
  datatable,
  plotsettings = NULL,
  returnplotly = FALSE,
  plotlysource = "A"
)
```

Arguments

datatable	The datatable in which the data is stored to plot.
plotsettings	The settings to use in the plot. Default=NULL, which loads default plotsettings from getdefaultplotsettings.
returnplotly	Set to TRUE to return the plotly variable to be used or modified in another setting (such as in a dashboard). Default=FALSE, which displays the plot.
plotlysource	Source attribute as defined in plot_ly). Default="A".

Value

The plotly variable or an invisible NULL.

See Also

Other plot functions: getdefaulthovertext(), getdefaultlabeltext(), getdefaultplotsettings(), makeplotsettings(), plot_makecolors(), plot_makemarkers(), plotcandlestick(), plotdensityfunctions(), plotecdffunctions(), plotpiechart(), plotscatterwithlink(), plotscatter(), sampledashboard(), updateplotsettings()

plotpiechart *Plot a pie chart*

Description

Plots a pie chart with columns from a data.table.

Usage

```
plotpiechart(
  datatable,
  plotsettings = NULL,
  returnplotly = FALSE,
  plotlysource = "A",
  donuthole = 0
)
```

Arguments

datatable	The datatable in which the data is stored to plot.
plotsettings	The settings to use in the plot. Default=NULL, which loads default plotsettings from getdefaultplotsettings.
returnplotly	Set to TRUE to return the plotly variable to be used or modified in another setting (such as in a dashboard). Default=FALSE, which displays the plot.
plotlysource	Source attribute as defined in plot_ly). Default="A".
donuthole	Set to a value between 0 and 1 to create a donut pie chart.

Value

The plotly variable or an invisible NULL.

See Also

Other plot functions: getdefaulthovertext(), getdefaultlabeltext(), getdefaultplotsettings(), makeplotsettings(), plot_makecolors(), plot_makemarkers(), plotcandlestick(), plotdensityfunctions(), plotecdffunctions(), plothistogram(), plotscatterwithlink(), plotscatter(), sampledashboard(), updateplotsettings()

plotscatter *Plot a scatterplot*

Description

Plots a scatterplot with columns from a data.table.

Usage

```
plotscatter(
  datatable,
  plotsettings = NULL,
  returnplotly = FALSE,
  plotlysource = "A"
)
```

Arguments

datatable	The data.table in which the data is stored to plot.
plotsettings	The settings to use in the plot. Default=NULL, which loads default plotsettings from getdefaultplotsettings.
returnplotly	Set to TRUE to return the plotly variable to be used or modified in another setting (such as in a dashboard). Default=FALSE, which displays the plot.
plotlysource	Source attribute as defined in plot_ly). Default="A".

Value

The plotly variable if returnplotly=TRUE or an invisible NULL. It will always return NULL and print a message if the plotsettings do not match the data.

See Also

Other plot functions: getdefaulthovertext(), getdefaultlabeltext(), getdefaultplotsettings(), makeplotsettings(), plot_makecolors(), plot_makemarkers(), plotcandlestick(), plotdensityfunctions(), plotecdffunctions(), plothistogram(), plotpiechart(), plotscatterwithlink(), sampledashboard(), updateplotsettings()

---

plotscatterwithlink *Plot a scatterplot*

Description

Plots a scatterplot with columns from a data.table.

Usage

```
plotscatterwithlink(
  datatable,
  xaxiscolumn = "x",
  yaxiscolumn = "y",
  namecolumn = NULL,
  linkxaxiscolumn = NULL,
  linkyaxiscolumn = NULL,
  linknamecolumn = NULL,
  plotsettings = NULL,
  returnplotly = FALSE,
  plotlysource = "A"
)
```

Arguments

datatable	The datatable in which the data is stored to plot.
xaxiscolumn	String with column name to use for the x axis (default="x").
yaxiscolumn	String with column name to use for the y axis (default="y").
namecolumn	String with column name to use to display name information, meaning that each data point has a name as specified in this column. Default=NULL, which does not use point names.
linkxaxiscolumn	String with column name to use for the x axis, drawing linkage lines from the representing xaxiscolumn and yaxiscolumn. Default=NULL, which draws no linkage lines between related points.
linkyaxiscolumn	String with column name to use for the y axis, drawing linkage lines from the representing xaxiscolumn and yaxiscolumn. Default=NULL, which draws no linkage lines between related points.
linknamecolumn	String with column name to use to display name information for linkage points. Default=NULL, which uses the settings as specified in namecolumn.
plotsettings	The settings to use in the plot. Default=NULL, which loads default plotsettings from getdefaultplotsettings.
returnplotly	Set to TRUE to return the plotly variable to be used or modified in another setting (such as in a dashboard). Default=FALSE, which displays the plot.
plotlysource	Source attribute as defined in plot_ly). Default="A".

Value

The plotly variable or an invisible NULL.

See Also

Other plot functions: getdefaulthovertext(), getdefaultlabeltext(), getdefaultplotsettings(), makeplotsettings(), plot_makecolors(), plot_makemarkers(), plotcandlestick(), plotdensityfunctions(), plotecdffunctions(), plothistogram(), plotpiechart(), plotscatter(), sampledashboard(), updateplotsettings()

---

plot_makecolors *Make colors*

Description

Make the color settings for each data point in a graph.

Usage

```
plot_makecolors(plotdata, plotsettings, whichaxis)
```

Arguments

plotdata	The data to use in the plot.
plotsettings	The settings to use in the plot.
whichaxis	The integer value that indicates the axis used.

Value

The color settings for each data point (list with a named vector `color`).

See Also

Other plot functions: `getdefaulthovertext()`, `getdefaultlabeltext()`, `getdefaultplotsettings()`, `makeplotsettings()`, `plot_makemarkers()`, `plotcandlestick()`, `plotdensityfunctions()`, `plotecdffunctions()`, `plothistogram()`, `plotpiechart()`, `plotscatterwithlink()`, `plotscatter()`, `sampledashboard()`, `updateplotsettings()`

---

plot_makemarkers *Make markers*

Description

Make the marker settings for each data point in a graph.

Usage

```
plot_makemarkers(plotdata, plotsettings, whichaxis)
```

Arguments

`plotdata`	The data to use in the plot.
`plotsettings`	The settings to use in the plot.
`whichaxis`	The integer value that indicates the axis used.

Value

The marker settings for each data point.

See Also

Other plot functions: `getdefaulthovertext()`, `getdefaultlabeltext()`, `getdefaultplotsettings()`, `makeplotsettings()`, `plot_makecolors()`, `plotcandlestick()`, `plotdensityfunctions()`, `plotecdffunctions()`, `plothistogram()`, `plotpiechart()`, `plotscatterwithlink()`, `plotscatter()`, `sampledashboard()`, `updateplotsettings()`

prefixcolnames *Prefix all column names*

Description

Add a prefix text to all column names of a data.table

Usage

```
prefixcolnames(thedata, prefix, omitcolumns = NULL)
```

Arguments

prefix	String to add as prefix to the column names
omitcolumns	Columns that do not need a prefix (default=NULL, which prefixes every column)
datatable	The data.table to update columns for

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

rad2deg *Radians to degrees*

Description

Radians to degrees

Usage

```
rad2deg(rad)
```

Arguments

rad	Radians.

Value

Degrees.

See Also

Other math functions: deg2rad(), runavgweighed(), runavg(), runsd()

`rd_to_wgs84` *Translate coordinates in rijksdriehoek (Dutch National Grid) system to WGS84*

Description

Translate coordinates in rijksdriehoek (Dutch National Grid) system to WGS84

Usage

```
rd_to_wgs84(x, y)
```

Arguments

x	vector of x-coordinates in rijksdriehoek system; or without y a vector of length 2 containing the x and y coordinates.
y	vector of y-coordinates in rijksdriehoek system

Value

A list with lamba (decimal degrees; longitude) and phi (decimal degrees; latitude), or if just x was given a numeric vector of length 2 with lambda and phi.

`recastPOSIXct` *Recast timezone*

Description

Transform a timezone to another but keep the exact time the same (no calculation). This one is slow.

Usage

```
recastPOSIXct(x, tz)
```

Arguments

x	The POSIXct time to change
tz	The timezone to transform to

Value

The POSIXct time in the new timezone

recasttimezone *Recast timezone (quick)*

Description

Transform a timezone to another but keep the time the same (no calculation). Uses a numeric difference calculation based on the timezone time difference of the first item instead of character transformations, so that transformation is faster. This might not always be the right way when dealing with timezone values that have a change in daylight saving time.

Usage

```
recasttimezone(x, tz)
```

Arguments

x	The POSIXct time to change
tz	The timezone to transform to

Value

The POSIXct time in the new timezone

removeallNAcolumns *Remove columns containing NA*

Description

Remove columns from a data.table where all data in that column is NA.

Usage

```
removeallNAcolumns(data)
```

Arguments

data	The data.table to remove the columns from.

Value

The data.table with the columns removed.

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

removeallNArows *Remove rows containing only NA*

Description

Remove columns from a data.table where all data in a row is NA. Different from na.omit in that the complete row needs to be NA.

Usage

```
removeallNArows(data)
```

Arguments

data	The data.table to remove the rows from.

Value

The data.table with the rows removed.

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

removecolumns *Remove columns*

Description

Remove specific named columns from a data.table (if they exist).

Usage

```
removecolumns(data, remcols)
```

Arguments

data	The data.table to remove the columns from.
remcols	Column names (character vector) to remove.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

---

removeisoinfo *Remove ISO information*

Description

Remove ISO date information from a data.table, as was added through the addisoinfo function.

Usage

```
removeisoinfo(datatable)
```

Arguments

datatable	The data.table to remove ISO weeknumber, year, and month from

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

Other date and time functions: addisoinfo(), makeconsistentdateformat(), makeconsistentdatetimeformat(), makedatecolumns(), makedatetimecolumns(), seqdays()

removeNAfromtable *Remove NA values*

Description

Update (by reference) all NA's in a data.table with zero or FALSE or an empty string (based on the column class).

Usage

```
removeNAfromtable(datatable)
```

Arguments

datatable	The data.table to update.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

removeoffset *Remove offset Remove offset value from time series data for a period that is considered non-moving.*

Description

Remove offset Remove offset value from time series data for a period that is considered non-moving.

Usage

```
removeoffset(dataset, statictime = 60, timecolumn = NULL, datacolumn = NULL)
```

Arguments

dataset	The data.table containing the data from which offset needs to be removed.
statictime	The amount of seconds of the beginning of the dataset to take as static non-moving. Default=60.
timecolumn	The name of the column that contains the time information in POSIXct. Default=NULL, which takes the first column.
datacolumn	The name of the column(s) that contain the data (numeric). Default=NULL, which takes all columns that are not the time column.

Value

The sensor data with the offset removed, or NULL on failure.

See Also

Other data functions: createbins(), localmaxima(), localminimamaxima(), localminima(), timeseriesstats()

Other time series functions: createbins(), timeseriesstats()

---

removerows *Remove rows by reference*

Description

Remove specific rows from a data.table (by providing indices). Note that removing a row by reference when the table only consists of one row does not work.

Usage

```
removerows(data, remrows)
```

Arguments

data	The data.table to remove the rows from.
remrows	Row indices (for example as returned by which=TRUE).

Value

Returns the modified data.table. It is modified by reference if not all rows are removed, but returns a 'copy' if all rows are removed.

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(removeallNArows(), removecolumns(), removeisoinfo(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

renamecolumns *Rename columns*

Description

Rename specific named columns from a data.table (if they exist).

Usage

```
renamecolumns(datatable, origcols, newcols)
```

Arguments

datatable	The data.table to rename the columns from.
origcols	Original column names (character vector).
newcols	New column names (character vector). Must be of same length as origcols.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

replaceelements *Replace elements in a vector*

Description

Replace elements in a vector by values from another (equal length) vector based on a vector condition.

Usage

```
replaceelements(input, condition, values)
```

Arguments

input	The input vector from which elements need to be replaced.
condition	A condition or TRUE/FALSE vector that specifies which elements need to be replaced (need to be of equal length as input).
values	A vector (need to be of equal length as input) for which values need to be taken where the condition is TRUE.

Value

New vector in which the elements of input are replaced with elements from values for which condition is TRUE.

See Also

Other vector functions: dorollapply(), extendvector(), findconstantchunks(), findconstant(), leaveonlychunks(), localmaxima(), localminimamaxima(), localminima(), makelength(), replaceNAelements(), runavgweighed(), runavg(), runsd(), shift()

`replaceNAelements` *Replace NA in a vector*

Description

Replace all NA values in a vector by a single value.

Usage

```
replaceNAelements(input, value = 0)
```

Arguments

input	The input vector from which NA elements need to be replaced.
value	The value to use instead of NA Default=0.

Value

New vector in which NA is replaced by the value.

See Also

Other vector functions: dorollapply(), extendvector(), findconstantchunks(), findconstant(), leaveonlychunks(), localmaxima(), localminimamaxima(), localminima(), makelength(), replaceelements(), runavgweighed(), runavg(), runsd(), shift()

`reproject_geojson` *Reproject a geoJSON file to an other coordinate system*

Description

Reproject a geoJSON file to an other coordinate system

Usage

```
reproject_geojson(
  filein,
  fileout,
  projection,
  inencoding = getOption("encoding")
)
```

Arguments

filein	the input file or connection.
fileout	the output file or connection.
projection	a function that projects coordinates from the original coordinate system to the new system. The function has to accept numeric vectors of length two as its input and return a vector of length two.
inencoding	when filein is a character string this encoding is used when reading from the file.

Value

A list containing the converted geoJSON. This output is usually not needed.

---

rightjoin *Right join data.tables*

Description

Execute a right join on two data.tables, meaning that all data of the right data.table (additionaltable) will be kept, and only the data from the left data.table (basetable) is linked to where it can be found in the right data.table.

Usage

```
rightjoin(
  basetable,
  additionaltable,
  basetablejoincolumns,
  additionaltablejoincolumns = NULL,
  additionaltableextractcolumns = NULL,
  removejoincols = NA
)
```

Arguments

basetable	The base data.table (if unmatched in additionaltable, NA values are added for this data.table).
additionaltable	The additional data.table.
basetablejoincolumns	The column name string of the base data.table to join on.
additionaltablejoincolumns	The column name string of the additional data.table to join on. Default=NULL, which assumes the same column name as basetablejoincolumns.
additionaltableextractcolumns	The column name strings of the additional data.table to add to the base data.table. Default=NULL, which assumes that all columns need to be joined.

removejoincols If TRUE, it will remove the provided basetablejoincolumns from the resulting table. If FALSE, it will keep the provided basetablejoincolumns in the result. If NA (default), it will check if basetablejoincolumns and additionaltablejoincolumns are the same. The column names that are different will result in the respective basetablejoincolumns to be removed.

For example, let's take basetablejoincolumns=c("ID","SECID") and additionaltablejoincolu With removejoincols=TRUE, the column "ID2" will return in the result, but columns "ID" and "SECID" will not. With removejoincols=FALSE, the columns "ID.x", "ID.y", "ID2" and "SECID" will return. Note that column ID exists in both the base as well as additional table, which results in the "ID.x" and "ID.y" columns. With removejoincols=NA the column "ID" and "ID2" will return in the result, but column "SECID" will not.

Value

The joined data.table or, when join column(s) or extracted column(s) don't exist, the original base data.table.

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

---

runavg *Running average*

Description

Calculate running average.

Usage

```
runavg(vec, width)
```

Arguments

vec	The vector to calculate the running average for.
width	Width of the window.

Value

Vector of the same length as vec with the running average.

See Also

Other math functions: deg2rad(), rad2deg(), runavgweighed(), runsd()

Other vector functions: dorollapply(), extendvector(), findconstantchunks(), findconstant(), leaveonlychunks(), localmaxima(), localminimamaxima(), localminima(), makelength(), replaceNAelements(), replaceelements(), runavgweighed(), runsd(), shift()

---

runavgweighed *Running average weigthed*

Description

Calculate running average with weighing factors.

Usage

```
runavgweighed(vec, weighvals)
```

Arguments

vec	The vector to calculate the running average for.
weighvals	Vector containing the weighing values (length of vector indicates the amount of previous points to use for calculation).

Value

Vector of the same length as vec with the running average.

See Also

Other math functions: deg2rad(), rad2deg(), runavg(), runsd()

Other vector functions: dorollapply(), extendvector(), findconstantchunks(), findconstant(), leaveonlychunks(), localmaxima(), localminimamaxima(), localminima(), makelength(), replaceNAelements(), replaceelements(), runavg(), runsd(), shift()

---

runcolfun *Run function on columns*

Description

Run a specific function on the provided columns, to update them.

Usage

```
runcolfun(thedata, FUN, thecolnames = NULL)
```

Arguments

thedata	The data.table to modify.
thecolnames	Column names to run the function on. Default=NULL, which runs the function on all columns.
thefun	The function to run on the columns.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

---

runsd *Running standard deviation*

Description

Calculate running standard deviation. In constract with sd, uses denominator *n*.

Usage

```
runsd(vec, width)
```

Arguments

vec	The vector to calculate the running standard deviation for.
width	Width of the window.

Value

Vector of the same length as vec with the running standard deviation.

See Also

Other math functions: deg2rad(), rad2deg(), runavgweighed(), runavg()

Other vector functions: dorollapply(), extendvector(), findconstantchunks(), findconstant(), leaveonlychunks(), localmaxima(), localminimamaxima(), localminima(), makelength(), replaceNAelements(), replaceelements(), runavgweighed(), runavg(), shift()

sampledashboard *Interactive dashboard*

Description

Display data from a data.table in an interactive graph. This function displays a sample dashboard with many buttons lacking exact functionality. It's intention is for the code to be modified to fit a dashboard designed for a specific application.

Usage

```
sampledashboard(
  alldata,
  plotsettings = NULL,
  categorycolumnname = "CATEGORY",
  stoponlastaction = TRUE
)
```

Arguments

alldata	The data.table to interactively display.
plotsettings	The settings for the plot. Has to be a modification of the returned list from getdefaultplotsettings default=(NULL, which plots according to the default settings as returned by getdefaultplotsettings).
categorycolumnname	String with column name to use for the different categories axis (default="CATEGORY").
stoponlastaction	Set to FALSE to keep the interactive plot running when the last item in the drop-down item has been handled. Default=TRUE, which closes the plot after the last item has been handled.

See Also

Other plot functions: getdefaulthovertext(), getdefaultlabeltext(), getdefaultplotsettings(), makeplotsettings(), plot_makecolors(), plot_makemarkers(), plotcandlestick(), plotdensityfunctions(), plotecdffunctions(), plothistogram(), plotpiechart(), plotscatterwithlink(), plotscatter(), updateplotsettings()

selectuniques *Extract unique categories*

Description

Extract the unique categories/elements found in the dataset for a specific column.

Usage

```
selectuniques(data, datacolumn)
```

Arguments

data	The data.table in which all approaches are listed.
datacolumn	String containing the column from the data.table to extract from.

Value

Vector with all unique categories/elements.

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

Examples

```
selectuniques(data,"TREINNR") #returns all unique train numbers
```

---

selectwhere *Extract from a dataset*

Description

Extract specified columns from the dataset where 1 specific column contains specific values. This follows the SQL principle "SELECT selectcols FROM data WHERE datacolumn IN datavalues".

Usage

```
selectwhere(data, whereclause, selectcols = NULL, uniques = FALSE)
```

Arguments

data	The data.table in which all approaches are listed
whereclause	String containing the condition certain columns must match
selectcols	Vector containing column name strings to export (default=NULL, which returns all columns).
uniques	Set to TRUE if only unique rows are to be returned (default=FALSE, which returns all matching rows).

Value

Data.table containing all rows for which the column matches the values. Returns a vector if uniques=TRUE and selectcols contains only one column.

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

Examples

```
selectwhere(data, "is_ATTENTIE==1&is_ALARM==1",c("DATUM","TREINNR"),TRUE) #returns all unique dates and train
```

`selectwherein` *Extract from a dataset*

Description

Extract specified columns from the dataset where 1 specific column contains specific values. This follows the SQL principle "SELECT selectcols FROM data WHERE datacolumn IN datavalues".

Usage

```
selectwherein(data, datacolumn, datavalues, selectcols = NULL, uniques = FALSE)
```

Arguments

`data`	The data.table in which all approaches are listed.
`datacolumn`	String containing the column which must match the values.
`datavalues`	The data values to search for.
`selectcols`	Vector containing column name strings to export (default=`NULL`, which returns all columns).
`uniques`	Set to `TRUE` if only unique rows are to be returned (default=`FALSE`, which returns all matching rows).

Value

Data.table containing all rows for which the column matches the values.

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

Examples

```
selectwherein(data, "SIGNAALREMCURVE",c("ATTENTIE","ALARM"),c("DATUM","TREINNR"),TRUE) #returns all unique d
```

seqdays *Sequence of days*

Description

Get a sequence of days.

Usage

```
seqdays(startdate = NULL, enddate = NULL, stepdate = 1, timezone = "")
```

Arguments

startdate	The start date. Default=NULL, which assumes today.
enddate	The end date. Default=NULL, which assumes today.
stepdate	The sequence increment (default=1).
timezone	String determining the timezone (from as.POSIXct). Default="", which uses the local time zone.

Value

The sequence in POSIXct format.

See Also

Other date and time functions: addisoinfo(), makeconsistentdateformat(), makeconsistentdatetimeformat(), makedatecolumns(), makedatetimecolumns(), removeisoinfo()

setcolclass *Set column class*

Description

Generic function to set columns of a data.table to a specific class.

Usage

```
setcolclass(thedata, theclassto, theclassfrom = NULL, thecolnames = NULL)
```

Arguments

thedata	The data.table to modify.
theclassto	Single string or function representing the class to transform columns to.
theclassfrom	Single string or function representing the class the column originally needs to be. Default=NULL, which transforms all column types to the provided class.
thecolnames	Vector string of column names to set the class for. Default=NULL, which sets the class for all columns.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

---

setcolnames *Rename columns*

Description

Rename specific named columns from a data.table (if they exist).

Usage

```
setcolnames(datatable, newnames, orignames = NULL)
```

Arguments

datatable	The data.table to rename the columns from.
newnames	New column names (character vector). Must be of same length as orignames.
orignames	Original column names (character vector). Default=NULL, which tries to rename all columns.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcoltofront(), setcolumns(), unlistcolumns(), updatetablevalues()

setcoltofront *Move specific columns to front*

Description

Move specific named columns to the front of a data.table, while keeping the other column names in the provided order.

Usage

```
setcoltofront(datatable, columnnames)
```

Arguments

datatable	The data.table to move columns.
columnnames	The column names (character vector) to move to the front, in the order that they should be.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcolumns(), unlistcolumns(), updatetablevalues()

setcolumns *Set columns*

Description

Set the columns that a data.table must have. If it does not exist, it will add the column with a default value.

Usage

```
setcolumns(data, cnames, cval = NA)
```

Arguments

data	The data.table which needs to have the columns.
cnames	The column names that need to exist.
cval	The default value for newly added columns. Default=NA.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), unlistcolumns(), updatetablevalues()

---

shift *Shift vector values*

Description

Shift vector values to the left or right.

Usage

```
shift(x, n, invert = FALSE, default = NA)
```

Arguments

x	Vector for which to shift values.
n	Number of places to be shifted. Positive numbers will shift to the right while negative numbers will shift to the left. This direction can be inverted by the invert parameter.
invert	Whether shift direction should be inverted.
default	The value that should be inserted by default (at the empty spots).

Value

The shifted vector.

See Also

Other vector functions: dorollapply(), extendvector(), findconstantchunks(), findconstant(), leaveonlychunks(), localmaxima(), localminimamaxima(), localminima(), makelength(), replaceNAelements(), replaceelements(), runavgweighed(), runavg(), runsd()

stripnames *Strip string of named elements*

Description

Remove all specific named elements from a list. Executes recursively so that it also works within lists in lists.

Usage

```
stripnames(thelist, thenames)
```

Arguments

thelist	List to remove names from.
thenames	Vector of strings containing the names of the elements to remove from the list.

Value

List with all the named elements removed.

See Also

Other list functions: forcelisttype(), leavenames(), listdepth()

str_pad_month *Pad month with 0*

Description

Simple function to pad zeros to a month

Usage

```
str_pad_month(themonth)
```

Arguments

themonth	Month value or string to pad

Value

String with month value padded (always zero padded left, length 2)

See Also

Other string functions: paste0collapse()

timeseriesstats *Get some time series statistics*

Description

Get some time series statistics

Usage

```
timeseriesstats(thetimes, thedata = NULL)
```

Arguments

thetimes	Data.table with the first column being the DateTime and the second column being the values, or, a vector containing the DateTime only.
thedata	Vector containing the values correlated to the DateTime. Must be of equal length to thetimes. Default=NULL, which only uses the first parameter.

Value

A named list with some time series statistics or NULL on failure. The "grouping" variables show how many points would be within a specific time period, and how many bins would occur for the full time period. This can be useful for calculating optimal statistics for data lasting a long time.

See Also

createbins

Other data functions: createbins(), localmaxima(), localminimamaxima(), localminima(), removeoffset()

Other time series functions: createbins(), removeoffset()

unlistcolumns *Unlist columns*

Description

Unlist all columns that are of the list type.

Usage

```
unlistcolumns(thedata, thecolnames = NULL)
```

Arguments

thedata	The data.table to modify.
thecolnames	Column names to unlist. Default=NULL, which unlists all columns that it detects to be of class list.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(), removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), updatetablevalues()

---

updateplotsettings *Update plotsettings*

Description

Update the plotsettings based on the column names in the data. It checks if the provided settings are possible to display. This function is always called from the plot functions.

Usage

```
updateplotsettings(plotsettings, datacolnames)
```

Arguments

plotsettings	The plotsettings to check.
datacolnames	The column names present in the data.

Value

Updated plotsettings with axes removed that are not present in the data. If none of the axes are present (either x-axis or y-axes), the function will return NULL.

See Also

Other plot functions: getdefaulthovertext(), getdefaultlabeltext(), getdefaultplotsettings(), makeplotsettings(), plot_makecolors(), plot_makemarkers(), plotcandlestick(), plotdensityfunctions(), plotecdffunctions(), plothistogram(), plotpiechart(), plotscatterwithlink(), plotscatter(), sampledashboard()

updatetablevalues *Update values*

Description

Update (by reference) specific values in a data.table.

Usage

```
updatetablevalues(datatable, from, to, columns = NULL)
```

Arguments

datatable	The data.table to update.
from	A vector of the items to replace.
to	A vector of replacement values.
columns	A vector of column names, or vector of integer column indices, where replacement needs to happen. Default=NULL, which replaces the values in all columns.

Value

Returns the modified data.table invisibly (modified by reference).

See Also

Other data.table functions: addcolumn(), addisoinfo(), copytoclipboard(), dojoin(), extractcategories(), fulljoin(), getcolumnclasses(), innerjoin(), leftjoin(), loadlargetable(), loadsastable(), loadspecifictable(), loadtable(), makedatecolumns(), makedatetimecolumns(), makeintegercolumns(), makenumericcolumns(), mapcolumnclasses(), prefixcolnames(), removeNAfromtable(), removeallNAcolumns(removeallNArows(), removecolumns(), removeisoinfo(), removerows(), renamecolumns(), rightjoin(), runcolfun(), selectuniques(), selectwherein(), selectwhere(), setcolclass(), setcolnames(), setcoltofront(), setcolumns(), unlistcolumns()

wgs84_to_rd *Translate coordinates in WGS84 to rijksdriehoek (Dutch National Grid)*

Description

Translate coordinates in WGS84 to rijksdriehoek (Dutch National Grid)

Usage

```
wgs84_to_rd(lambda, phi)
```

Arguments

lambda	numeric vector with longitutes (in decimal degrees), or without phi a vector of length 2 containing both phi and lambda.
phi	numeric vector with latitudes (in decimal degrees).

Value

A list with x and y coordinates in Rijksdriehoek system coordinates, of if just lambda was given a numeric vector of length 2 with both coordinates.

Index