Package ‘sparkTable’

December 13, 2016

Type Package
Title Sparklines and Graphical Tables for TeX and HTML
Version 1.3.0
Date 2016-12-13
Author Alexander Kowarik, Bernhard Meindl, Matthias Templ
Maintainer Alexander Kowarik <alexander.kowarik@statistik.gv.at>
Description Create sparklines and graphical tables for documents and websites.
URL https://github.com/alexkowa/sparkTable
BugReports https://github.com/alexkowa/sparkTable/issues
Depends utils, methods, Cairo, gridExtra, ggplot2, shiny, Rglpk
Imports xtable,StatMatch,boot,pixmap,RGraphics,grid
License GPL
NeedsCompilation no
Repository CRAN
Date/Publication 2016-12-13 19:55:10

R topics documented:

   brp .......................................................... 2
   checkerplot .................................................. 3
   customizeSparkTable ........................................ 5
   export-methods ............................................. 6
   geoTable-class ............................................ 9
   getParameter .............................................. 10
   newGeoTable .............................................. 11
   newSparkBar .............................................. 12
   newSparkTable ........................................... 14
   optimal_grid_allocation .................................. 16
   plot-methods ............................................... 17
   reshapeExt ............................................... 17


Description

brp...Gross Regional Product Austria
gdp...GDP for 41 different regions and from 1997 to 2008
gini...gini Index for different european countries from 2004 to 2007
dat...Production Index in Austria
pop...the number of people living in Austria in different subgroups and years
popEU ... population of the EU
debtEU ... debt of the European countries
coordsEU ... coordinates of the European capitals
alcohol ... alcohol consumption in the European Union from 1970 until 2008
EU_data ... Unemployment rates of females in the EU from 1997 until 2008
USdata1 ... Health insurance coverage from US states in 2009
USdata2 ... Health insurance by government and private insurance from US states in 2009
coordUS ... coordinates of the US states (centres)
AT_Soccer ... Results from the Austrian Bundesliga 2013/2014

Usage

data(brp)
data(EU_data)
data(UStdata1)
data(UStdata2)
data(coordUS)
data(alcohol)
data(gdp)
data(gini)
data(dat)
data(pop)
data(popEU)
data(debtEU)
data(coordsEU)
data(AT_Soccer)

Author(s)

Bernhard Meindl, Alexander Kowarik, Statistics Austria
checkerplot

Examples

data(brp)
data(gdp)
data(gini)
data(pi)
data(population)

checkerplot  Checkerplot

Description

Visualization of regional indicators with the checkerplot

Usage

checkerplot(data, cols=5, rows=5, geom="line", errorbar=FALSE, title=NULL, title.size=20, label.size=11, xbreaks=NULL, xlabels=NULL, ybreaks=NULL, ylabels=NULL, ymin=NULL, ymax=NULL, img=NULL, aes.geom=NULL, margin_yaxis=0, margin_yaxis2=0, margin_xaxis=0, margin_xaxis2=0, opts=NULL, ...)

Arguments

data  data frame
cols  number of columns in the grid
rows  number of rows in the grid
geom  defines the geom, the geometric object, of the plot ("line", "bar", "point" are possible at the moment)
errorbar  TRUE/FALSE plot additional errorbars (testing only)
title  title of the plot, displayed above the plot
title.size  font size of the plot title
label.size  font size of the labels
xbreaks  number of breaks for the x-axis, default it is calculated automatically
xlabels  labels of the breaks (optional)
ybreaks  number of breaks for the y-axis, default it is calculated automatically
ylabels  labels of the breaks (optional)
ymin  minimum value of y-axis displayed, if not provided it will be automatically calculated
ymax  maximum value of x-axis displayed, if not provided it will be automatically calculated
img  vector containing all names (must equal the label column in the dataframe data) of pictures (pnm files) displayed beside the label
aes.geom ggplot2 object to define the aesthetics of the geom

margin_yaxis allows to adjust the distance from the plot to the left border of the grid for all elements with an y-axis and its annotation, because the difference number formats on the y-axis might lead to a little displaced plot area

margin_yaxis2 allows to adjust the distance from the plot to the left border of the grid for all elements with an y-axis but without annotation

margin_xaxis allows to adjust the distance from the plot to the left border of the grid for all elements with an x-axis and annotation, because the difference number formats on the y-axis might lead to a little displaced plot area

margin_xaxis2 allows to adjust the distance from the plot to the left border of the grid for all elements with an x-axis but without annotation

opts ggplot2 object to change settings in the ggplot2 plots

... further arguments passed through

Author(s)
Karin Fuerst, Alexander Kowarik, Matthias Templ

See Also
optimal_grid_allocation, export

Examples

```r
## Not run:
## Directory of the package with flags
dirflags = paste(searchpaths()[grep("sparkTable", searchpaths())], "/etc/Flaggen/", sep="")
## EXAMPLE for EUROPE
data(EU_data)
order_eu = optimal_grid_allocation(EU_data[,16:17],8,7)
colnames(EU_data)[1] <- c("country")
EU_data[,18] <- order_eu
colnames(EU_data)[1] <- c("country")
colnames(EU_data)[18] <- c("order")
data_eu = data.frame(cbind(rep(1997,34), EU_data$unempl_f_1997, EU_data$country, EU_data$order))
colnames(data_eu) = c("x", "y", "names", "order")
for(year in 1998:2008){
  XX <- data.frame(cbind(rep(year,34), EU_data[,paste("unempl_f_",year,sep="")], EU_data$country, EU_data$order))
colnames(XX) = c("x", "y", "names", "order")
data_eu = rbind(data_eu,XX)
}
dirflags_eu=paste(dirflags,"EU/",sep="")
checkerplot(data_eu, cols=8, rows=7, geom="line", img=dirflags_eu,
title = "Unemployment rate in Europe 1997 to 2008 (in perc)", title.size=18,
ylables=c(5,10,15,20),
```
### Example for US data
```
data(coordUS)

## Rearrange states which are far away:
coordUS[coordUS$state == "AK","x"] <- 0.4  ## Rearrange Alaska
coordUS[coordUS$state == "HI","x"] <- 0.5  ## Rearrange Hawaii

## Optimal arrangement of the states:
order <- optimal_grid_allocation(coordUS[,1:2],13,6)
order <- data.frame(names=coordUS$state,order=order)

## Load US healthy insurance data
data(USdata1)
data(USdata2)
nam <- names(USdata1)

## Delete previous order (optional)
USdata1 <- USdata1[,which(names(USdata1)=="order")]

## Combine data and ordering:
USdata1 <- merge(USdata1,order,all.x=TRUE,all.y=FALSE)
USdata1[is.na(USdata1$order),"order"] <- 14

## Define directory with flags (pmn's):
dirflags_us=paste(dirflags,"USstates/",sep="")
USdata2[,3] <- USdata2[,3]*100

cunkerplot(USdata2[-,2],cols=13, rows=6, geom="bar",
title="US private health insurance (in percent of the population)",
title.size=18, ylabels=c(0,60,90), ybreaks=c(0,60,90),
img=dirflags_us, margin_yaxis=-0.05, margin_yaxis2=-0.2,
margin_xaxis=-0.4, margin_xaxis2=0)
```

## End (Not run)

---

**CustomizeSparkTable**  
*Customize a sparkTable within a shiny App*

### Description
Interactive modification of an `sparkTable-class`

### Usage
```
customizeSparkTable(object, outputDir=getwd())
```

### Arguments
- **object**: an object of class `sparkTable-class`
- **outputDir**: a path to a directory for the output (Default=current working directory)
See Also

export

Examples

```r
## Not run:
data(pop, package="sparkTable")
b <- newSparkBox()
l <- newSparkLine()
bb <- newSparkBar()
content <- list(
  function(x) { round(mean(x),2) },
  b,
  l,
  bb,
  function(x) { round(tail(x,1),2) }
)
names(content) <- paste("column",1:length(content),sep="")
varType <- rep("value",length(content))
pop <- pop$C("variable","value","time")
pop$time <- as.numeric(as.character(pop$time))
xx <- reshapeExt(pop, varying=list(2))
x1 <- newSparkTable(xx, content, varType)
x1$dataObj$v1 <- rpois(nrow(xx),1)
x1$dataObj$v2 <- rpois(nrow(xx),2)
x1$dataObj$v3 <- rpois(nrow(xx),3)
customizeSparkTable(x1, outputDir=getwd())

## End(Not run)
```

---

**export-methods**  
Saves objects of class 'sparkline', 'sparkbar', 'sparkhist' or 'sparkbox' to a file

Description

Spark-Objects are plotted and saved into different file-formats while for objects of class `sparkTable` all required graphs are stored and the necessary code to include the graphical table in the desired format is returned to the prompt.

Usage

```r
export(object, ...)
## S4 method for signature 'sparkline'
export(object, outputType="pdf", filename="sparkLine", ...)
## S4 method for signature 'sparkbar'
export(object, outputType="pdf", filename="sparkBar", ...)
## S4 method for signature 'sparkhist'
```
export-methods

export(object, outputType="pdf", filename="sparkHist", ...)  
## S4 method for signature 'sparkBox'
export(object, outputType="pdf", filename="sparkBox", ...)  
## S4 method for signature 'sparkTable'
export(object, outputType="html", filename=NULL,  
       graphNames="out", infonote=TRUE, scaleByCol=FALSE,...)
## S4 method for signature 'geoTable'
export(object, outputType="html", filename=NULL,  
       graphNames="out", transpose=FALSE, include.rownames=FALSE,include.colnames=FALSE,  
       rownames=NULL, colnames=NULL,...)

Arguments

object       an object of class 'sparkline', 'sparkbox' or 'sparkbar'.
outputType   for objects of class sparkline, sparkbar or sparkbox a character vector specifying the desired output formats. Supported formats are:
             • 'pdf': a pdf image is produced
             • 'eps': an eps image is produced
             • 'png': a png image is produced
             • 'svg': a svg image is produced
For objects of class sparkTable a character vector specifying the desired output format, supported types are:
             • "tex": latex output is produced
             • "html": html output (using png-graphs) is generated
             • "htmlsvg": html output (using svg-graphs) is generated
filename     the filename of the output (minus '.pdf', '.eps','.eps', 'svg' for single graphs, minus '.tex' or '.html' for graphical tables
graphNames   the main part of the single graphic files that are produced (minus '-someIndex.extension'
infonote     TRUE/FALSE if the latex command should be included in the output, only used for graphical tables (sparkTable)
scaleByCol   Either TRUE/FALSE to set the scaling for all columns or a TRUE/FALSE vector of the same length as the content object. Keeps the scaling the same in all rows of a column. This parameter is used only for graphical tables (sparkTable)
transpose    logical vector of length 1 defining if the plot be transposed (for geoTable-objects)
include.rownames logical vector of length 1 defining if rownames should be included (for geoTable-objects)
include.colnames logical vector of length 1 defining if colnames should be included (for geoTable-objects)
rownames     optional character vector specifying row names (for geoTable-objects)
colnames     optional character vector specifying column names (for geoTable-objects)
...         additional parameters to be passed, currently not used
Author(s)

Bernhard Meindl, Alexander Kowarik, Statistics Austria

Examples

```r
## Not run:
data(pop)
x <- pop[pop[,2]=="Insgesamt",3]
a <- newSparkLine(values=x, pointWidth=8)
export(a, outputType=c('pdf','png'), filename='myFirstSparkLine')

# simple graphical table
data(pop, package="sparkTable")
b <- newSparkBox()
l <- newSparkLine()
bb <- newSparkBar()
content <- list(function(x) { round(mean(x),2) },
                 b,l,bb,function(x) { round(tail(x,1),2) })
names(content) <- paste("column",1:length(content),sep="")
varType <- rep("value",length(content))
pop$[c("variable","value","time")]
xx <- reshapeExt(pop, varying=list(2))
x1 <- newSparkTable(xx, content, varType)
export(x1, outputType="html", graphNames="o2",filename="t1")
export(x1, outputType="tex", graphNames="o3",filename="t2")

##Geo-Table: EU population and debt
data(popEU, package="sparkTable")
data(debtEU, package="sparkTable")
data(coordsEU, package="sparkTable")
popEU <- popEU[popEU$country%in%coordsEU$country,]
debtEU <- debtEU[debtEU$country%in%coordsEU$country,]
EU <- cbind(popEU,debtEU[,1])
EUlong <- reshapeExt(EU,
                     idvar="country",v.names=c("pop","debt"),
                     varying=list(2:13,14:25),geographicVar="country",timeValues=1999:2010)
l <- newSparkLine()
l <- setParameter(l, 'lineWidth', 2.5)
content <- list(
              function(x)("Population:"),
              function(x)("Debt:"),l)
varType <- c(rep("pop",2),rep("debt",2))
xGeoEU <- newGeoTable(EUlong, content, varType, geographicVar="country",
                       geographicInfo=coordsEU)
export(xGeoEU, outputType="html", graphNames="outEU",
       filename="testEUT",transpose=TRUE)
export(xGeoEU, outputType="html", graphNames="outEU1",
       filename="testEU", transpose=FALSE)
export(xGeoEU, outputType="tex", graphNames="out1",
       filename="testEUT", transpose=FALSE)
geoTable-class

```r
filename="testEU", transpose=FALSE)
#export(xGeoEU, outputType="tex", graphNames="out1",
filename="testEUT", transpose=TRUE)

## End(Not run)
```

---

**geoTable-class**  
*Class "geoTable"*

---

**Description**

This class defines data objects holding all information required to create a geoTable.

**Objects from the Class**

Objects can be created by using function `newGeoTable` and exported using method `export`.

**Slots**

- `dataObj`: Object of class "listOrNULL" ~
- `varType`: Object of class "characterOrNULL" ~
- `tableContent`: Object of class "listOrNULL" ~
- `geographicVar`: Object of class "characterOrNULL" ~
- `geographicInfo`: Object of class "dfOrNULL" ~
- `geographicOrder`: Object of class "dfOrNULL" ~

**Author(s)**

Bernhard Meindl, Alexander Kowarik, Statistics Austria

**See Also**

- `export`

**Examples**

```r
showClass("geoTable")
```
**getParameter**

*Functions to interact with a Sparkline object*

**Description**

Basic functions to query parameters for objects of class 'sparkline', 'sparkbar', 'sparkbox', 'sparkTable' or 'geoTable'.

**Usage**

```r
getParameter(object, type)
```

**Arguments**

- `object`: objects of class 'sparkline', 'sparkbar', 'sparkbox', 'sparkTable' or 'geoTable'
- `type`: one of the following:
  - 'width': query slot 'width' for objects of class 'spark' and classes that directly extend this class.
  - 'height': query slot 'height' for objects of class 'spark' and classes that directly extend this class.
  - 'values': query slot 'values' for objects of class 'spark' and classes that directly extend this class.
  - 'padding': query slot 'padding' for objects of class 'spark' and classes that directly extend this class.
  - 'allColors': query slot 'allColors' for objects of class 'sparkline'.
  - 'lineWidth': query slot 'lineWidth' for objects of class 'sparkline'.
  - 'pointWidth': query slot 'pointWidth' for objects of class 'sparkline'.
  - 'showIQR': query slot 'showIQR' for objects of class 'sparkline'.
  - 'boxCol': query slot 'boxCol' for objects of class 'sparkbox'.
  - 'outCol': query slot 'outCol' for objects of class 'sparkbox'.
  - 'boxLineWidth': query slot 'boxLineWidth' for objects of class 'sparkbox'.
  - 'barCol': query slot 'barCol' for objects of class 'sparkbar'.
  - 'barSpacingPerc': query slot 'barSpacingPerc' for objects of class 'sparkbar'.
  - 'dataObj': query slot 'dataObj' for objects of class 'sparkTable'.
  - 'tableContent': query slot 'tableContent' for objects of class 'sparkTable'.
  - 'varType': query slot 'varType' for objects of class 'sparkTable'.
  - 'geographicVar': query slot 'geographicVar' for objects of class 'geoTable'.
  - 'geographicInfo': query slot 'geographicInfo' for objects of class 'geoTable'.
  - 'geographicOrder': query slot 'geographicOrder' for objects of class 'geoTable'.

**Author(s)**

Bernhard Meindl, Alexander Kowarik, Statistics Austria
newGeoTable

See Also

setParameter

Examples

data(pop)
  x <- pop[pop[,2]=="Insgesamt",3]
  a <- newSparkLine(values=x, pointWidth=8)
  a <- setParameter(a, type='values', value=sample(1:10, 15, replace=TRUE))
  getParameter(a, 'values')

  a <- setParameter(a, type='allColors',
    value=c("darkred", "darkgreen","darkblue", "white", "black", "red"))
  getParameter(a, 'allColors')

  getParameter(a, 'pointWidth')
  a <- setParameter(a, type='pointWidth', value=3)
  getParameter(a, 'pointWidth')

  a <- setParameter(a, type='lineWidth', value=1)
  a <- setParameter(a, type='width', value=6)
  a <- setParameter(a, type='height', value=.6)

newGeoTable

Functions to create a new object of class 'geoTable'

Description

User-function to create objects of the class 'geoTable'.

Usage

newGeoTable(dataObj, tableContent, varType, geographicVar, geographicInfo=NULL)

Arguments

dataObj a data frame containing information to be plotted.

tableContent a list with elements of class 'sparkline','sparkbox','sparkbar' or 'function'

varType a character vector containing variable names existing in dataObj.

geographicVar a character variable of length 1 with a variable name of dataObj that holds re-

geographicInfo if specified, a data.frame containing 3 columns.
  • first column: row-indices
  • second column: column-indices
  • third column: regional codes
newSparkBar

Functions to create new Spark object

Description

Basic functions to create objects of the class 'spark'. The functions are the base for creating a graphical table.

Usage

newSparkLine(width=NULL, height=NULL, values=NULL, padding=NULL, allColors=NULL, pointWidth=NULL, lineWidth=NULL, showIQR=NULL, vMin=NULL, vMax=NULL, outputType="html")
newSparkBar(width=NULL, height=NULL, values=NULL, padding=NULL, barCol=NULL, barWidth=NULL, barSpacingPerc=NULL, vMin=NULL, vMax=NULL, bgCol=NULL, outputType="html")
newSparkBox(width=NULL, height=NULL, values=NULL, padding=NULL, boxOutCol=NULL, boxMedCol=NULL, boxShowOut=NULL, boxCol=NULL, boxLineWidth=NULL, vMin=NULL, vMax=NULL, bgCol=NULL, outputType="html")
newSparkHist(width=NULL, height=NULL, values=NULL, padding=NULL, barCol=NULL, barWidth=NULL, barSpacingPerc=NULL, vMin=NULL, vMax=NULL, bgCol=NULL, outputType="html")

Examples

## Not run:
### Example EU population and debt
data(popEU, package="sparkTable")
data(debtEU, package="sparkTable")
data(coordsEU, package="sparkTable")
popEU <- popEU[popEU$country%in%coordsEU$country,]
debtEU <- debtEU[debtEU$country%in%coordsEU$country,]
EU <- cbind(popEU, debtEU[, -1])
EUlong <- reshapeExt(EU, idvar = "country", v.names = c("pop", "debt"),
                   varying = list(2:13, 14:25), geographicVar = "country",
                   timeValues = 1999:2010)
1 <- newSparkLine()
1 <- setParameter(1, 'lineWidth', 2.5)
content <- list(function(x){"Population:"}, 1,
                function(x){"Debt:"}, 1)
varType <- c(rep("pop", 2), rep("debt", 2))
xGeoEU <- newGeoTable(EUlong, content, varType, geographicVar = "country",
                      geographicInfo = coordsEU)

## End(Not run)
newSparkBar

Arguments

width described in setParameter
height described in setParameter
values described in setParameter
padding described in setParameter
allColors described in setParameter
pointWidth described in setParameter
lineWidth described in setParameter
showIQR described in setParameter
vMin numeric vector of length 1 defining minimum value required for data scaling
vMax numeric vector of length 1 defining maximum value required for data scaling
barCol described in setParameter
barWidth described in setParameter
barSpacingPerc described in setParameter
boxOutCol character vector of length 1 defining the color of outliers in spark boxplots
boxMedCol character vector of length 1 defining the color of median line in spark boxplots
boxShowOut logical vector specifying if outliers should be displayed in spark boxplots
boxCol described in setParameter
boxLineWidth described in setParameter
bgCol described in setParameter
outputType described in plot

Author(s)

Bernhard Meindl, Alexander Kowarik, Statistics Austria

See Also

plot, export, setParameter, getParameter

Examples

```r
## Not run:
data(pop)
x <- pop[pop[,2] == "Insgesamt", 3]

## SparkLine
a <- newSparkLine(values=x, pointWidth=8)
export(a, outputType='png', filename='testLine1')

a <- setParameter(a, sample(1:10, 15, replace=TRUE), type='values')
getParameter(a, type='values')

a <- setParameter(a, c("darkred", "darkgreen", "darkblue", "white", "black", "red"),
```
newSparkTable

Function to create new SparkTable object

Description

User-function to create objects of the class 'sparkTable'.

```r

newSparkTable

type='allColors')
getParameter(a, type='allColors')

a <- setParameter(a, 3, type='pointWidth')
a <- setParameter(a, 1, type='lineWidth')

export(a, outputType="pdf", filename='testLine2')

a <- setParameter(a, 6, type='width')
a <- setParameter(a, .6, type='height')
export(a, outputType='eps', filename='testLine2')

### SparkBar
b <- newSparkBar(values=x-min(x))
getParameter(b, type='values')

b <- setParameter(b, c("darkred", "darkgreen","black"), type='barCol')
export(b, outputType='pdf', filename='testBar1')

b <- setParameter(b, 0:10, type='values')
export(b, outputType='pdf', filename='testBar2')

b <- setParameter(b, 0:-10, type='values')
export(b, outputType='pdf', filename='testBar3')

### SparkBox
cc <- newSparkBox(values=x)
cc <- setParameter(cc, "darkgreen", type='outCol')
getParameter(cc, type='outCol')
cc <- setParameter(cc, c("black","red"), type='boxCol')

export(cc, outputType='pdf', filename='testBox1')

cc <- setParameter(cc, c("black","darkgreen"), type='boxCol')
cc <- setParameter(cc, "darkred", type='outCol')
export(cc, outputType='pdf', filename='testBox2')

### SparkHist
hh <- newSparkHist(values=rnorm(100))
export(hh, outputType='pdf', filename='testHist1')

## End(Not run)
```
newSparkTable

Usage

newSparkTable(dataObj, tableContent, varType)

Arguments

dataObj a data frame containing information to be plotted.
tableContent a list with elements of class 'sparkline','sparkbox','sparkbar' or 'function'
varType a character vector containing variable names existing in dataObj.

Author(s)

Bernhard Meindl, Alexander Kowarik, Statistics Austria

See Also

plot, export, setParameter, getParameter

Examples

## Not run
## Soccer
data(AT_Soccer,package="sparkTable")
content <- list(
  function(x) {sum(x)},
  function(x) { round(sum(x),2) },
  function(x) { round(sum(x),2) },
  newSparkLine(), newSparkBar()
)
names(content) <- c("Points","ShotGoal","GetGoal","GoalDiff","WinLose")
varType <- c("points","shotgoal","getgoal","goaldiff","wl")
x1 <- newSparkTable(AT_Soccer, content, varType)
showSparkTable(x1)
export(x1, outputType="html")

#Population
data(pop)
b <- newSparkBox()
l <- newSparkLine()
bb <- newSparkBar()
content <- list(function(x) { round(mean(x),2) },
  b, l, bb,
  function(x) { round(tail(x,1),2) })
names(content) <- paste("column",1:5,sep="")
varType <- rep("value",5)
pop <- pop[,c("variable","value","time")]
pop$time <- as.numeric(as.character(pop$time))
xx <- reshapeExt(pop, varying=list(2))
x1 <- newSparkTable(xx, content, varType)
export(x1, outputType="html", graphNames="out1")
optimal_grid_allocation

Optimal Allocation of Coordinates to a grid

Description

optimal_grid_allocation() ... newGeoTable.

Usage

optimal_grid_allocation(data, grid.cols=NULL, grid.rows=NULL, addGrid=0, plot=FALSE)

Arguments

data                  data frame with first column X-coordinate and second column Y-coordinate
grid.cols             number of columns in the grid
grid.rows             number of rows in the grid
addGrid               additional columns and rows in the grid
plot                  TRUE/FALSE for plotting the allocation

Author(s)

Alexander Kowarik, Statistics Austria

See Also

export

Examples

data <- data.frame(x=c(0,2,1.24,2,1.98,1.62,1.24,1.91,0.48),
y=c(2.93,2.45,1.94,1.46,0.98,3,0.70,0.56,0))
rownames(data) <- c("IS","FI","NO","EE","LV","SE","DK","LT","IE")
index <- optimal_grid_allocation(data,plot=TRUE)
index2 <- optimal_grid_allocation(data,grid.cols=3,grid.rows=4,plot=TRUE)
plot-methods

Plot objects of class 'sparkline', 'sparkbar', 'sparkhist' or 'sparkbox'

Description

Function that calls plot-methods for objects of class 'sparkline', 'sparkbar', 'sparkhist' or 'sparkbox'.

Usage

plot(x, y, ...)

Arguments

x an object of class 'sparkline', 'sparkbox' or 'sparkbar'.
y not used, only for compatibility.
... additional parameters passed. Currently possible values:
   - padding: numeric vector of length 4 containing positive values. These are
     internally rescaled and appropriate margins are added to the resulting plots.

Author(s)

Bernhard Meindl, Alexander Kowarik, Statistics Austria

Examples

data(pop)
x <- pop[pop[,2]=="Insgesamt",3]
a <- newSparkLine(values=x, pointWidth=8)
plot(a)

reshapeExt

Reshaping datasets

Description

reshapeExt() can be used to transform data that are already in 'long' format to the form that the data
can be used by newSparkTable or newGeoTable.
Usage

\[
\text{reshapeExt(data, timeValues=NULL, } \\
g\text{ geographicVar=NULL, varying = NULL, v.names = NULL, timevar = "time", } \\
idvar = "id", ids = 1:NROW(data), } \\
drop = NULL, new.row.names = NULL, } \\
sep = ".", } \\
split = if (sep == ") \{ \\
list(regexp = "[A-Za-z][0-9]", include = TRUE) \} else \{ \\
list(regexp = sep, include = FALSE, fixed = TRUE)\}
\]

Arguments

data a data frame

timeValues if specified, vector of valid time-points

geographicVar if specified, name of a variable in x holding regional information.

varying names of sets of variables in the wide format that correspond to single variables in long format ('time-varying'). This is canonically a list of vectors of variable names, but it can optionally be a matrix of names, or a single vector of names. In each case, the names can be replaced by indices which are interpreted as referring to names(data). See 'Details of ?reshape' for more details and options.

v.names names of variables in the long format that correspond to multiple variables in the wide format. See 'Details of ?reshape'.

timevar the variable in long format that differentiates multiple records from the same group or individual. If more than one record matches, the first will be taken (with a warning).

idvar Names of one or more variables in long format that identify multiple records from the same group/individual. These variables may also be present in wide format.

ids the values to use for a newly created idvar variable in long format.

drop a vector of names of variables to drop before reshaping.

new.row.names character or NULL: a non-null value will be used for the row names of the result.

sep A character vector of length 1, indicating a separating character in the variable names in the wide format. This is used for guessing v.names and times arguments based on the names in varying. If sep == ", the split is just before the first numeral that follows an alphabetic character. This is also used to create variable names when reshaping to wide format.

split A list with three components, regexp, include, and (optionally) fixed. This allows an extended interface to variable name splitting. See 'Details of ?reshape'.

Note

Wrapper for the stats function reshape.
setParameter

Author(s)
Bernhard Meindl, Alexander Kowarik, Statistics Austria

See Also
setParameter, getParameter, reshape

Examples

data(pop, package='sparkTable')
content <- list(
  function(x) { round(mean(x),2) },
  newSparkBox(), newSparkLine(), newSparkBar(),
  function(x) { round(tail(x,1),2) })
names(content) <- paste('column',i,sep='')
varType <- rep('value',5)
pop <- pop[,c('variable','value','time')]
pop$time <- as.numeric(as.character(pop$time))
xx <- reshapeExt(pop, varying=list(2))
x1 <- newSparkTable(xx, content, varType)
#export(x1, outputType='html', graphNames='o2', filename='t1')

Description
Basic functions to set parameters for objects of class 'sparkline', 'sparkbar', 'sparkbox', 'sparkTable' or 'geoTable'.

Usage
setParameter(object, value, type)

Arguments

<table>
<thead>
<tr>
<th>object</th>
<th>objects of class 'sparkline', 'sparkbar', 'sparkbox', 'sparkTable' or 'geoTable'</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>one of the following:</td>
</tr>
<tr>
<td></td>
<td>• 'width': set/change slot 'width' for objects of class 'spark' and classes that</td>
</tr>
<tr>
<td></td>
<td>directly extend this class.</td>
</tr>
<tr>
<td></td>
<td>• 'height': set/change slot 'height' for objects of class 'spark' and classes that</td>
</tr>
<tr>
<td></td>
<td>directly extend this class.</td>
</tr>
<tr>
<td></td>
<td>• 'values': set/change slot 'values' for objects of class 'spark' and classes that</td>
</tr>
<tr>
<td></td>
<td>directly extend this class.</td>
</tr>
<tr>
<td></td>
<td>• 'padding': set/change slot 'padding' for objects of class 'spark' and classes</td>
</tr>
<tr>
<td></td>
<td>that directly extend this class.</td>
</tr>
<tr>
<td></td>
<td>• 'allColors': set/change slot 'allColors' for objects of class 'sparkline'.</td>
</tr>
</tbody>
</table>
• 'lineWidth': set/change slot 'lineWidth' for objects of class 'sparkline'.
• 'pointWidth': set/change slot 'pointWidth' for objects of class 'sparkline'.
• 'showIQR': set/change slot 'showIQR' for objects of class 'sparkline'.
• 'boxCol': set/change slot 'boxCol' for objects of class 'sparkbox'.
• 'outCol': set/change slot 'outCol' for objects of class 'sparkbox'.
• 'boxLineWidth': set/change slot 'boxLineWidth' for objects of class 'sparkbox'.
• 'barCol': set/change slot 'barCol' for objects of class 'sparkbar'.
• 'barSpacingPerc': set/change slot 'barSpacingPerc' for objects of class 'sparkbar'.
• 'bgCol': set/change slot 'bgCol' for objects of class 'sparkbar','sparkhist' and 'sparkbox'.
• 'dataObj': set/change slot 'dataObj' for objects of class 'sparkTable' or 'geoTable'.
• 'tableContent': set/change slot 'tableContent' for objects of class 'sparkTable' or 'geoTable'.
• 'varType': set/change slot 'varType' for objects of class 'sparkTable' or 'geoTable'.
• 'geographicVar': set/change slot 'geographicVar' for objects of class 'geoTable'.
• 'geographicInfo': set/change slot 'geographicInfo' for objects of class 'geoTable'.
• 'geographicOrder': set/change slot 'geographicOrder' for objects of class 'geoTable'.

value

values that are used to updated the slot chosen with argument 'type':

• if type=='width': numeric vector of length 1 defining the width of the resulting plot
• if type=='height': numeric vector of length 1 defining the height of the resulting plot
• if type=='values': numeric vector defining the values to be plotted
• if type=='padding': numeric vector of length 4 defining the padding of the plot in percent. The order is: top,bottom,left,right.
• if type=='allColors': a character vector of length 6 (including NA's) containing colors. The elements of the color vector are used as:
  – first element: color for minimal value
  – second element: color for maximal value
  – third element: color for last value
  – fourth element: color for filling
  – fifth element: color for the line
  – sixth element: color for interquartil range
• if type=='lineWidth': numeric vector of length 1 defining the line width of the resulting sparkline
• if type=='pointWidth': numeric vector of length 1 defining the width of points (min, max, last) of the resulting sparkline.
• if type=='showIQR': logical vector of length 1 defining if the IQR of the data should be plotted in the sparkline.
- if type=='boxCol': character vector of length 2 defining colors to be used in a sparkbox plot.
  - first element: color of the lines surrounding the boxes
  - second element: fill color of the box
- if type=='outCol': character vector of length 1 defining the color of outliers in a sparkboxplot.
- if type=='bgCol': character vector of length 1 defining the color of the plot background.
- if type=='boxLineWidth': numeric vector of length 1 defining the width of the surrounding lines of a sparkboxplot.
- if type=='barCol': character vector of length 3 defining colors to be used in a sparkbar plot.
  - first element: color of bars showing negative values
  - second element: color of bars showing positive values
  - third element: color of lines in the plot
- if type=='barSpacingPerc': numeric vector of length 1 defining the spacing in percent used between the bars in the sparkbar plot
- if type=='dataObj': a data frame containing information to be plotted.
- if type=='tableContent': a list with elements of class 'sparkline','sparkbox','sparkbar' or 'function'
- if type=='varType': a character vector containing variable names existing in dataObj.
- if type=='geographicVar': a character variable of length 1 with a variable name of dataObj that holds regional information.
- if type=='geographicInfo': a data.frame with information on coordinates of regions to be plotted.
- if type=='geographicOrder': a data.frame containing 3 columns that is usually automatically created.
  - first column: row-indices
  - second column: column-indices
  - third column: regional codes

**Author(s)**

Bernhard Meindl, Alexander Kowarik, Statistics Austria

**See Also**

getParameter

**Examples**

data(pop)
x <- pop[pop[,2]=='Insgesamt',3]
a <- newSparkLine(values=x, pointWidth=8)
a <- setParameter(a, type='values', value=sample(1:10, 15, replace=TRUE))
getParameter(a, 'values')

a <- setParameter(a, type='allColors',
                     value=c("darkred", "darkgreen", "darkblue", "white", "black", "red"))

getParameter(a, 'pointWidth')

a <- setParameter(a, type='pointWidth', value=3)

getParameter(a, 'pointWidth')

a <- setParameter(a, type='lineWidth', value=1)

a <- setParameter(a, type='width', value=6)

a <- setParameter(a, type='height', value=.6)

---

showSparkTable-methods

*Look at your sparkTable in a shiny App*

**Description**

--- Methods for function showSparkTable ---

**Usage**

```r
showSparkTable(object, outputDir=tempdir(), outputType="html", filename=NULL,
               graphNames="out", ...)
```

**Arguments**

- `object` an object of class 'sparkTable' or 'data.frame'
- `outputDir` a path to a directory for the output (Default=temporary directory)
- `outputType` a character vector of length one specifying the desired output format:
  - 'tex': latex output is produced (does not work)
  - 'html': html output is procuced
- `filename` the filename of the output (minus '.tex' or '.html')
- `graphNames` the main part of the single graphic files that are produced (minus '-someIndex.extension')
- `...` additional parameters to be passed to `export` and `summaryST`

**Methods**

signature(object = "sparkTable")

**See Also**

- `export`
showSparkTable-methods

Examples

```r
## Not run:
data(pop.package="sparkTable")
b <- newSparkBox()
l <- newSparkLine()
bb <- newSparkBar()
content <- list(function(x) { round(mean(x),2) },
b,
l,
bb,
  function(x) { round(tail(x,1),2) }
)
names(content) <- paste("column",1:length(content),sep="")
varType <- rep("value",length(content))
pop <- pop[,c("variable","value","time")]
pop$time <- as.numeric(as.character(pop$time))
xx <- reshapeExt(pop, varying=list(2))
x1 <- newSparkTable(xx, content, varType)
showSparkTable(x1)

#Example Hist+Box with 2 variables in 10 different groups
dataEx <- data.frame(
  variable=sample(paste("Cat",1:10,sep="_"),1000,replace=TRUE),
  value=rnorm(1000),value2=rlnorm(1000)
)
b <- newSparkBox()
h <- newSparkHist()
content <- list(
  function(x) { round(mean(x),2) },
  function(x) { round(median(x),2) },
  function(x) { round(quantile(x,.25),2) },
  function(x) { round(quantile(x,.75),2) },
  b,
h,
  function(x) { round(mean(x),2) },
  function(x) { round(median(x),2) },
  function(x) { round(quantile(x,.25),2) },
  function(x) { round(quantile(x,.75),2) },
  b,
h
)
names(content) <- c(
  paste(c("Mean","Median","Q25","Q75","Boxplot","Histogram"),"_v1",sep=""),
  paste(c("Mean","Median","Q25","Q75","Boxplot","Histogram"),"_v2",sep=""
)
varType <- c(rep("value",length(content)/2),rep("value2",length(content)/2))
dataEx <- reshapeExt(dataEx, varying=list(2,3))
x2 <- newSparkTable(dataEx, content, varType)
showSparkTable(x2)

#Example for the data.frame method (uses summaryST)
data2 <- data.frame(x1=rnorm(100),x2=rnorm(100)+1,x3=rnorm(100)+5)
```

Description

This class defines data objects holding all information required to plot sparkbars.

Objects from the Class

Objects can be created by using function `newSparkBar`.

Slots

- `barCol`: Object of class "ANY"
- `bgCol`: Object of class "ANY"
- `barWidth`: Object of class "numeric"
- `barSpacingPerc`: Object of class "numeric"
- `width`: Object of class "numeric"
- `height`: Object of class "numeric"
- `values`: Object of class "numeric"
- `padding`: Object of class "numeric"
- `availableWidth`: Object of class "numeric"
- `availableHeight`: Object of class "numeric"
- `stepWidth`: Object of class "numeric"
- `coordsX`: Object of class "numeric"
- `coordsY`: Object of class "numeric"

Methods

No methods defined with class "sparkbar" in the signature.

Author(s)

Bernhard Meindl, Alexander Kowarik, Statistics Austria

See Also

`newSparkBar`, `plot`, `export`, `setParameter`, `getParameter`

Examples

`showClass("sparkbar")`
Description

This class defines data objects holding all information required to plot sparkboxes.

Objects from the Class

Objects can be created by using function `newSparkBox`.

Slots

outCol: Object of class "ANY" ~~
boxCol: Object of class "ANY" ~~
bgCol: Object of class "ANY" ~~
boxLineWidth: Object of class "numeric" ~~
width: Object of class "numeric" ~~
height: Object of class "numeric" ~~
values: Object of class "numeric" ~~
padding: Object of class "numeric" ~~
availableWidth: Object of class "numeric" ~~
availableHeight: Object of class "numeric" ~~
stepWidth: Object of class "numeric" ~~
coordsX: Object of class "numeric" ~~
coordsY: Object of class "numeric" ~~

Methods

No methods defined with class "sparkbox" in the signature.

Author(s)

Bernhard Meindl, Alexander Kowarik, Statistics Austria

See Also

newSparkBox, plot, export setParameter, getParameter

Examples

showClass("sparkbox")
Description
This class defines data objects holding all information required to plot sparklines.

Objects from the Class
Objects can be created by using function `newSparkLine`.

Slots
- `allColors`: Object of class "ANY"
- `pointWidth`: Object of class "numeric"
- `lineWidth`: Object of class "numeric"
- `showIQR`: Object of class "logical"
- `width`: Object of class "numeric"
- `height`: Object of class "numeric"
- `values`: Object of class "numeric"
- `padding`: Object of class "numeric"
- `availableWidth`: Object of class "numeric"
- `availableHeight`: Object of class "numeric"
- `stepWidth`: Object of class "numeric"
- `coordsX`: Object of class "numeric"
- `coordsY`: Object of class "numeric"

Methods
No methods defined with class "sparkline" in the signature.

Author(s)
Bernhard Meindl, Alexander Kowarik, Statistics Austria

See Also
`newSparkLine, plot, export, codesetParameter, getParameter`

Examples
`showClass("sparkline")`
sparkTable-class

Class "sparkTable"

Description

This class defines data objects holding all information required to create a sparkTable.

Objects from the Class

Objects can be created by using function `newSparkTable`.

Slots

dataObj: Object of class "dfOrNULL"

varType: Object of class "characterOrNULL"

tableContent: Object of class "listOrNULL"

Author(s)

Bernhard Meindl, Alexander Kowarik, Statistics Austria

See Also

export

Examples

sshowClass("sparkTable")

summaryST

summaryST - data frame in a graphical table

Description

summary for a data frame in a graphical table

Usage

summaryST(.Object, outputType="html", filename=NULL, graphNames="out", hist=TRUE, boxplot=TRUE, min=TRUE, quantile=TRUE, median=TRUE, mean=TRUE, max=TRUE, changeOrder=NULL, addFun=NULL, digits=2, scaleHistByCol=FALSE, scaleBoxByCol=FALSE)
Arguments

- `.Object` a data frame
- `outputType` a character vector of length one specifying the desired output format:
  - 'tex': latex output is produced
  - 'html': html output is procuded
- `filename` the filename of the output (minus '.tex' or '.html'
- `graphNames` the main part of the single graphic files that are produced (minus '-someIndex.extension'
- `hist` TRUE/FALSE for a histogramm
- `boxplot` TRUE/FALSE for a boxplot
- `min` TRUE/FALSE for the minimum
- `quantile` TRUE/FALSE for 1st and 3rd Quartile
- `median` TRUE/FALSE for the median
- `mean` TRUE/FALSE for the mean
- `max` TRUE/FALSE for the maximum
- `changeOrder` Indices for reordering the columns of the table
- `addFun` named list of additional functions e.g. var
- `digits` number of digits used for rounding
- `scaleHistByCol` TRUE/FALSE if the histograms of all variables should be on the same x-axis scale
- `scaleBoxByCol` TRUE/FALSE if the boxplots of all variables should be on the same x-axis scale

Value

object of class 'sparkTable' for further customizing the output (with setParameter)

Author(s)

Alexander Kowarik, Statistics Austria

See Also

export

Examples

```r
## Not run:
data1 <- data.frame(x=rnorm(100),y=rlnorm(100),
    z=rbeta(100,1,1))
#default summary table
summaryST(data1, filename="st1", graphNames="out1")
#changing the order of the columns
summaryST(data1, filename="st1b", changeOrder=c(6,7,2,3,1,4,5,8), graphNames="out1b")
#adding a custom column
summaryST(data1, filename="st1c", addFun=list(var=function(x)round(var(x,na.rm=TRUE),2)),
    graphNames="out1c")
```
data2 <- data.frame(x1=rnorm(100), x2=rnorm(100)+1, x3=rnorm(100)+5)
summaryST(data2, filename="st1d", graphNames="out1d", scaleHistByCol=TRUE, scaleBoxByCol=TRUE)
# the same results in a shiny app:

showSparkTable(data2)

## End(Not run)
Index

*Topic classes
  geoTable-class, 9
  sparkbar-class, 24
  sparkbox-class, 25
  sparkline-class, 26
  sparkTable-class, 27

alcohol (brp), 2
AT_Soccer (brp), 2

brp, 2

checkerplot, 3
coordSEU (brp), 2
coordUS (brp), 2
customizeSparkTable, 5

dat (brp), 2
debtEU (brp), 2

EU_data (brp), 2
export, 4, 6, 9, 12, 13, 15, 16, 22, 24–28
export (export-methods), 6
export, geoTable-method
  (export-methods), 6
export, sparkbar-method
  (export-methods), 6
export, sparkbox-method
  (export-methods), 6
export, sparkhist-method
  (export-methods), 6
export, sparkline-method
  (export-methods), 6
export, sparkTable-method
  (export-methods), 6
export-methods, 6
gdp (brp), 2
goToTable, 7
goToTable-class, 9
getParameter, 10, 13, 15, 19, 21, 24–26

gini (brp), 2
newGeoTable, 9, 11, 16, 17
newSparkBar, 12, 24
newSparkBox, 25
newSparkBox (newSparkBar), 12
newSparkHist (newSparkBar), 12
newSparkLine, 26
newSparkLine (newSparkBar), 12
newSparkTable, 14, 17, 27

optimal_grid_allocation, 4, 16

plot, 13, 15, 24–26
plot (plot-methods), 17
plot, sparkbar-method (plot-methods), 17
plot, sparkbox-method (plot-methods), 17
plot, sparkhist-method (plot-methods), 17
plot, sparkline-method (plot-methods), 17
plot-methods, 17
pop (brp), 2
popEU (brp), 2

reshape, 19
reshapeExt, 17

setParameter, 11, 13, 15, 19, 24–26
showSparkTable
  (showSparkTable-methods), 22
showSparkTable, data.frame-method
  (showSparkTable-methods), 22
showSparkTable, sparkTable-method
  (showSparkTable-methods), 22
showSparkTable-methods, 22
sparkbar, 7
sparkbar-class, 24
sparkbox, 7
sparkbox-class, 25
sparkline, 7
sparkline-class, 26
sparkTable, 6, 7
sparkTable-class. 27
summaryST. 22, 27
summaryST, data.frame-method
  (summaryST). 27
summaryST-methods (summaryST). 27

USdata1 (brp), 2
USdata2 (brp), 2