

# Package ‘sparkTable’

February 15, 2012

**Type** Package

**Title** Sparklines and graphical tables for tex and html

**Version** 0.9.3

**Date** 2012-02-06

**Author** Alexander Kowarik, Bernhard Meindl, Matthias Templ

**Maintainer**

Alexander Kowarik <alexander.kowarik@statistik.gv.at>, Bernhard Meindl <bernhard.meindl@statistik.gv.at>  
Templ <matthias.templ@statistik.gv.at>

**Description** Create Sparklines and graphical tables for documents and websites

**Depends** graphics, xtable, utils, grid, Cairo, methods, Rglpk, StatMatch, gridExtra, RGraphics, ggplot2, pixmap

**License** GPL

**Repository** CRAN

**Date/Publication** 2012-02-07 08:44:12

## R topics documented:

brp . . . . .	2
checkerplot . . . . .	3
geoTable-class . . . . .	5
getParameter . . . . .	6
newGeoTable . . . . .	7
newSparkBar . . . . .	8
newSparkTable . . . . .	10
optimal_grid_allocation . . . . .	11
plotGeoTable-methods . . . . .	12
plotSparks-methods . . . . .	13
plotSparkTable-methods . . . . .	14
reshapeExt . . . . .	15

setParameter . . . . .	16
sparkbar-class . . . . .	19
sparkbox-class . . . . .	20
sparkline-class . . . . .	21
sparkTable-class . . . . .	22

<b>Index</b>	<b>23</b>
--------------	-----------

---

brp	<i>data sets for testing the sparkTable package</i>
-----	---

---

## 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 Aus-  
tria pop...the number of people living in Austria in different subgroups and years popEU ... popula-  
tion of the EU debtEU ... debt of the European countries coordsEU ... coordinates of the European  
capitals alcohol ... alcohol conumption in the European Union from 1970 until 2008 EU\_data ...  
Unemployment rates of females in the EU from 1997 until 2008 USdata1 ... Health insurance cov-  
erage 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)

## Usage

```
data(brp)
data(EU_data)
data(USdata1)
data(USdata2)
data(coordUS)
data(alcohol)
data(gdp)
data(gini)
data(dat)
data(pop)
data(popEU)
data(debtEU)
data(coordsEU)
```

## Author(s)

Bernhard Meindl, Alexander Kowarik, Statistics Austria

## Examples

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

---

 checkerplot
*Checkerplot***Description**

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, formatter=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
formatter	allows to format the y-axis (scale_x_continuous(formatter = formatter))
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](#), [plotGeoTable](#)

### Examples

```
### 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="")
## Not run:
checkerplot(data_eu, cols=8, rows=7, geom="line", img=dirflags_eu,
            title = "Unemployment rate in Europe 1997 to 2008 (in %)", title.size=18,
            ylabels=c(5,10,15,20),
            ybreaks=c(5,10,15,20), xbreaks=c(1998,2001,2004,2007), xlabels=c("98","01","04","07"),
            margin_yaxis=-0.05, margin_yaxis2=-0.2,
            margin_xaxis=-0.4, margin_xaxis2=0 )

## End(Not run)

### EXAMPLE for US data
data(coordUS)
## rearrange states which are far away:
coordUS[coordUS$state %in% c("AK"),"x"] <- 0.4 ## rearrange Alaska
coordUS[coordUS$state %in% c("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
USdata1 <- USdata1[,nam]
## define directory with flags (pnm's):
dirflags_us=paste(dirflags,"USStates/",sep="")
USdata2[,3] <- USdata2[,3]*100
## Not run:
checkerplot(USdata2[,-2], cols=13, rows=6, geom="bar",
            title="US private health insurance (in % of the population)",
            title.size=18, ylabels=c(30,60,90), ybreaks=c(30,60,90),
            img=dirflags_us, margin_yaxis=-0.05, margin_yaxis2=-0.2,
            margin_xaxis=-0.4, margin_xaxis2=0,
            formatter=function(x,...) {sprintf("%.1f", x)})

## 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](#).

## 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**[plotGeoTable](#)**Examples**

```
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**

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

**Arguments**

object	objects of class 'sparkline', 'sparkbar', 'sparkbox', 'sparkTable' or 'geoTable'
type	one of the following: <ul style="list-style-type: none"> <li>• 'width': query slot 'width' for objects of class 'spark' and classes that directly extend this class.</li> <li>• 'height': query slot 'height' for objects of class 'spark' and classes that directly extend this class.</li> <li>• 'values': query slot 'values' for objects of class 'spark' and classes that directly extend this class.</li> <li>• 'padding': query slot 'padding' for objects of class 'spark' and classes that directly extend this class.</li> <li>• 'allColors': query slot 'allColors' for objects of class 'sparkline'.</li> <li>• 'lineWidth': query slot 'lineWidth' for objects of class 'sparkline'.</li> <li>• 'pointWidth': query slot 'pointWidth' for objects of class 'sparkline'.</li> <li>• 'showIQR': query slot 'showIQR' for objects of class 'sparkline'.</li> <li>• 'boxCol': query slot 'boxCol' for objects of class 'sparkbox'.</li> <li>• 'outCol': query slot 'outCol' for objects of class 'sparkbox'.</li> <li>• 'boxLineWidth': query slot 'boxLineWidth' for objects of class 'sparkbox'.</li> <li>• 'barCol': query slot 'barCol' for objects of class 'sparkbar'.</li> <li>• 'barSpacingPerc': query slot 'barSpacingPerc' for objects of class 'sparkbar'.</li> <li>• 'dataObj': query slot 'dataObj' for objects of class 'sparkTable'.</li> <li>• 'tableContent': query slot 'tableContent' for objects of class 'sparkTable'.</li> <li>• 'varType': query slot 'varType' for objects of class 'sparkTable'.</li> </ul>

- '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

**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 regional information.
geographicInfo	if specified, a data.frame containing 3 columns. <ul style="list-style-type: none"> <li>• first column: row-indices</li> <li>• second column: column-indices</li> <li>• third column: regional codes</li> </ul>

**Author(s)**

Bernhard Meindl, Alexander Kowarik, Statistics Austria

**See Also**

[plotGeoTable](#)

**Examples**

```
###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")
l <- newSparkLine()
l <- setParameter(l,'lineWidth',2.5)
content <- list(function(x){"Population:"},l,function(x){"Debt:"},l)
varType <- c(rep("pop",2),rep("debt",2))
xGeoEU <- newGeoTable(Eulong,content,varType,geographicVar="country",geographicInfo=coordsEU)
```

---

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, 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, outputType="html")
```

**Arguments**

width	described in <a href="#">setParameter</a>
height	described in <a href="#">setParameter</a>
values	described in <a href="#">setParameter</a>
padding	described in <a href="#">setParameter</a>
allColors	described in <a href="#">setParameter</a>
pointWidth	described in <a href="#">setParameter</a>
lineWidth	described in <a href="#">setParameter</a>
showIQR	described in <a href="#">setParameter</a>
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 <a href="#">setParameter</a>
barWidth	described in <a href="#">setParameter</a>
barSpacingPerc	described in <a href="#">setParameter</a>
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 <a href="#">setParameter</a>
boxLineWidth	described in <a href="#">setParameter</a>
outputType	described in <a href="#">plotSparks</a>

**Author(s)**

Bernhard Meindl, Alexander Kowarik, Statistics Austria

**See Also**

[plotSparkTable](#), [plotSparks](#), [setParameter](#), [getParameter](#)

**Examples**

```

data(pop)
x <- pop[pop[,2]=="Insgesamt",3]

### SparkLine
a <- newSparkLine(values=x, pointWidth=8)
plotSparks(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"), type='allColors')
getParameter(a, type='allColors')

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

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

a <- setParameter(a, 6, type='width')
a <- setParameter(a, .6, type='height')
plotSparks(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')
plotSparks(b, outputType='pdf', filename='testBar1')

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

b <- setParameter(b, 0:-10, type='values')
plotSparks(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')

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

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

```

**Description**

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

**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**

[plotSparkTable](#), [plotSparks](#), [setParameter](#), [getParameter](#)

**Examples**

```
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,idvar="variable", varying=list(2))
x1 <- newSparkTable(xx, content, varType)
plotSparkTable(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**

[plotGeoTable](#)

**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)
```

---

plotGeoTable-methods *Plot objects of class 'geoTable'*

---

**Description**

Function that plots objects of class 'geoTable'.

**Usage**

```
plotGeoTable(object, outputType='html', filename=NULL, graphNames='out', transpose=FALSE, include.row
```

**Arguments**

object	an object of class 'geoTable'
outputType	a character vector of length one specifying the desired output format: <ul style="list-style-type: none"> <li>'tex': latex output is produced</li> <li>'html': html output is produced</li> </ul>
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'  
 transpose        logical vector of length 1 defining if the plot be transposed  
 include.rownames  
                   logical vector of length 1 defining if rownames should be included  
 include.colnames  
                   logical vector of length 1 defining if colnames should be included  
 rownames        optional character vector specifying row names  
 colnames        optional character vector specifying column names  
 ...              additional parameters to be passed

**Author(s)**

Bernhard Meindl, Alexander Kowarik, Statistics Austria

**Examples**

```

###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")
l <- newSparkLine()
l <- setParameter(1, 'lineWidth', 2.5)
content <- list(function(x){"Population:"},l,function(x){"Debt:"},l)
varType <- c(rep("pop",2),rep("debt",2))
xGeoEU <- newGeoTable(EULong, content, varType,geographicVar="country",geographicInfo=coordsEU)
plotGeoTable(xGeoEU, outputType="html", graphNames="outEU",filename="testEUT",transpose=TRUE)
plotGeoTable(xGeoEU, outputType="html", graphNames="outEU1",filename="testEU",transpose=FALSE)
plotGeoTable(xGeoEU, outputType="tex", graphNames="out1",filename="testEU",transpose=FALSE)
plotGeoTable(xGeoEU, outputType="tex", graphNames="out1",filename="testEUT",transpose=TRUE)

```

---

plotSparks-methods      *Plot objects of class 'sparkline', 'sparkbar' or 'sparkbox'*

---

**Description**

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

**Usage**

```
plotSparks(object, outputType='pdf', filename='testSpark', ...)
```

**Arguments**

object	an object of class 'sparkline', 'sparkbox' or 'sparkbar'.
outputType	a character vector of length one specifying the desired output format: <ul style="list-style-type: none"> <li>• 'pdf': a pdf image is produced</li> <li>• 'eps': an eps image is procuded</li> <li>• 'png': a png image is procuded</li> </ul>
filename	the filename of the output (minus '.pdf', '.eps' or '.eps')
...	additional parameters to be passed

**Author(s)**

Bernhard Meindl, Alexander Kowarik, Statistics Austria

**Examples**

```
data(pop)
x <- pop[pop[,2]=="Insgesamt",3]
a <- newSparkLine(values=x, pointWidth=8)
plotSparks(a, outputType='pdf', filename='myFirstSparkLine')
```

---

plotSparkTable-methods

*Plot objects of class 'sparkTable'*

---

**Description**

Function that plots objects of class 'sparkTable'.

**Usage**

```
plotSparkTable(object, outputType='html', filename=NULL, graphNames='out', ...)
```

**Arguments**

object	an object of class 'sparkTable'
outputType	a character vector of length one specifying the desired output format: <ul style="list-style-type: none"> <li>• 'tex': latex output is produced</li> <li>• 'html': html output is procuded</li> </ul>
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

**Author(s)**

Bernhard Meindl, Alexander Kowarik, Statistics Austria

**Examples**

```

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",1:5,sep="")
varType <- rep("value",5)
pop <- pop[,c("variable","value","time")]
pop$time <- as.numeric(as.character(pop$time))
xx <- reshapeExt(pop,idvar="variable", varying=list(2))
x1 <- newSparkTable(xx, content, varType)
plotSparkTable(x1, outputType="html", graphNames="o2",filename="t1")
plotSparkTable(x1, outputType="tex", graphNames="o3",filename="t2")

```

---

 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**

```
reshapeExt(x,timeValues=NULL,geographicVar=NULL,...)
```

**Arguments**

x	data frame
timeValues	if specified, vector of valid time-points
geographicVar	if specified, name of a variable in x holding regional information.
...	additional parameter used for reshape()

**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',1:5,sep='')
varType <- rep('value',5)
pop <- pop[,c('variable','value','time')]
pop$time <- as.numeric(as.character(pop$time))
xx <- reshapeExt(pop, idvar='variable', varying=list(2))
x1 <- newSparkTable(xx, content, varType)
#plotSparkTable(x1, outputType='html', graphNames='o2',filename='t1')

```

---

setParameter

*Functions to interact with a Sparkline object*


---

**Description**

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

**Usage**

```
setParameter(object, value, type)
```

**Arguments**

object	objects of class 'sparkline', 'sparkbar', 'sparkbox', 'sparkTable' or 'geoTable'
type	one of the following: <ul style="list-style-type: none"> <li>'width': set/change slot 'width' for objects of class 'spark' and classes that directly extend this class.</li> <li>'height': set/change slot 'height' for objects of class 'spark' and classes that directly extend this class.</li> <li>'values': set/change slot 'values' for objects of class 'spark' and classes that directly extend this class.</li> <li>'padding': set/change slot 'padding' for objects of class 'spark' and classes that directly extend this class.</li> <li>'allColors': set/change slot 'allColors' for objects of class 'sparkline'.</li> <li>'lineWidth': set/change slot 'lineWidth' for objects of class 'sparkline'.</li> <li>'pointWidth': set/change slot 'pointWidth' for objects of class 'sparkline'.</li> <li>'showIQR': set/change slot 'showIQR' for objects of class 'sparkline'.</li> <li>'boxCol': set/change slot 'boxCol' for objects of class 'sparkbox'.</li> <li>'outCol': set/change slot 'outCol' for objects of class 'sparkbox'.</li> <li>'boxLineWidth': set/change slot 'boxLineWidth' for objects of class 'sparkbox'.</li> </ul>

- 'barCol': set/change slot 'barCol' for objects of class 'sparkbar'.
- 'barSpacingPerc': set/change slot 'barSpacingPerc' for objects of class 'sparkbar'.
- '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=='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, '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)
```

---

sparkbar-class	Class "sparkbar"
----------------	------------------

---

### 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" ~~  
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](#), [plotSparks](#), [setParameter](#), [getParameter](#)

### Examples

```
showClass("sparkbar")
```

---

sparkbox-class	Class "sparkbox"
----------------	------------------

---

### 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" ~~  
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](#), [plotSparks](#), [setParameter](#), [getParameter](#)

### Examples

```
showClass("sparkbox")
```

---

sparkline-class	Class "sparkline"
-----------------	-------------------

---

### 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](#), [plotSparks](#), [setParameter](#), [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**

[plotSparkTable](#)

**Examples**

```
showClass("sparkTable")
```

# Index

## \*Topic **classes**

- geoTable-class, [5](#)
  - sparkbar-class, [19](#)
  - sparkbox-class, [20](#)
  - sparkline-class, [21](#)
  - sparkTable-class, [22](#)
- alcohol (brp), [2](#)
- brp, [2](#)
- checkerplot, [3](#)
- coordsEU (brp), [2](#)
- coordUS (brp), [2](#)
- dat (brp), [2](#)
- debtEU (brp), [2](#)
- EU\_data (brp), [2](#)
- gdp (brp), [2](#)
- geoTable-class, [5](#)
- getParameter, [6](#), [9](#), [11](#), [15](#), [18–21](#)
- gini (brp), [2](#)
- newGeoTable, [5](#), [7](#), [11](#), [15](#)
- newSparkBar, [8](#), [19](#)
- newSparkBox, [20](#)
- newSparkBox (newSparkBar), [8](#)
- newSparkLine, [21](#)
- newSparkLine (newSparkBar), [8](#)
- newSparkTable, [10](#), [15](#), [22](#)
- optimal\_grid\_allocation, [4](#), [11](#)
- plotGeoTable, [4](#), [6](#), [8](#), [12](#)
- plotGeoTable (plotGeoTable-methods), [12](#)
- plotGeoTable, geoTable-method  
(plotGeoTable-methods), [12](#)
- plotGeoTable-methods, [12](#)
- plotSparks, [9](#), [11](#), [19–21](#)
- plotSparks (plotSparks-methods), [13](#)
- plotSparks, sparkbar-method  
(plotSparks-methods), [13](#)
- plotSparks, sparkbox-method  
(plotSparks-methods), [13](#)
- plotSparks, sparkline-method  
(plotSparks-methods), [13](#)
- plotSparks-methods, [13](#)
- plotSparkTable, [9](#), [11](#), [22](#)
- plotSparkTable  
(plotSparkTable-methods), [14](#)
- plotSparkTable, sparkTable-method  
(plotSparkTable-methods), [14](#)
- plotSparkTable-methods, [14](#)
- pop (brp), [2](#)
- popEU (brp), [2](#)
- reshape, [15](#)
- reshapeExt, [15](#)
- setParameter, [7](#), [9](#), [11](#), [15](#), [16](#), [19–21](#)
- sparkbar-class, [19](#)
- sparkbox-class, [20](#)
- sparkline-class, [21](#)
- sparkTable-class, [22](#)
- USdata1 (brp), [2](#)
- USdata2 (brp), [2](#)