Package 'svgtools'

November 20, 2020

Type Package

Title Manipulate SVG (Template) Files of Charts

Version 1.0.0

Date 2020-10-30

Description The purpose of this package is to manipulate SVG files that are templates of charts the user wants to produce.
In vector graphics one copes with x-/y-coordinates of elements (e.g. lines, rectangles, text). Their scale is often dependent on the program that is used to produce the graphics.
In applied statistics one usually has numeric values on a fixed scale (e.g. percentage values between 0 and 100) to show in a chart.
Basically, 'svgtools' transforms the statistical values into coordinates and widths/heights of the vector graphics.
This is done by stackedBar() for bar charts, by linesSymbols() for charts with lines and/or symbols (dot markers) and scatterSymbols() for scatterplots.

Encoding UTF-8

License GPL-3

LazyData true

RoxygenNote 7.1.1

Depends R (>= 4.0.0)

Imports xml2 (>= 1.3.0), stringr (>= 1.4.0), rsvg (>= 2.1), magick (>= 2.4.0)

Suggests knitr, rmarkdown

VignetteBuilder knitr, rmarkdown

NeedsCompilation no

Author Konrad Oberwimmer [aut, cre],
Christian Wimmer [aut],
Michael Bruneforth [ctb]

Maintainer Konrad Oberwimmer <konrad.oberwimmer@iqs.gv.at>

Repository CRAN

Date/Publication 2020-11-20 09:00:02 UTC

R topics documented:

`changeText` *Change text of text elements*

Description

Changes the text entry of XML element of type 'text'. The XML element may be found by its name (XML attribute 'id') or based on its current text entry.

Usage

```
changeText(
  svg,
  element_name,
  text,
  alignment = NULL,
  in_group = NULL,
  hide_blank = FALSE
)
```

Arguments

`svg`	XML document with SVG content
`element_name`	Name (attribute 'id') of text (XML element 'text') or current text entry of text (XML element 'text').
`text`	New text entry.
`alignment`	Character value for text alignment. Accepts 'start', 'middle', and 'end' (default NULL = no change).
`in_group`	Name (attribute 'id') of group (XML element 'g') that contains the text element (default NULL = no group, search the entire SVG).
`hide_blank`	Should text elements with empty strings be hidden (set attribute 'display' to 'none')? (default FALSE)

Value

XML document with SVG content

Examples

```
#read SVG file
fpath <- system.file("extdata", "fig1.svg", package="svgtools")
svg <- read_svg(file = fpath)

#change a text
svg <- changeText(svg = svg, element_name = "Category A", text = "low")
svg <- changeText(svg = svg, element_name = "Category B", text = "medium")
svg <- changeText(svg = svg, element_name = "Category C", text = "high")
```

`diffBar` *Adjust bar chart where bars lie to the left/right or bottom/top of a given nullvalue*

Description

Adjusts the horizontal (XML attribute 'x') or vertical (XML attribute 'y') position as well as width/height of bar segments (XML elements of type 'rect') and optionally value labels (XML elements of type 'text'). Positions are calculated relative to a given frame (XML element of type 'rect'), a nullvalue and the position of a data value within the minimum and maximum of a given scale. Bar segments and, optionally, value labels with values lower than the nullvalue are positioned to the left (in horizontal alignment) or to the bottom (in vertical alignment) of the scaled nullvalue, higher values on the opposite side.
For further description see `stackedBar`.

Usage

```
diffBar(
  svg,
  frame_name,
  group_name,
  scale_real,
  values,
  nullvalue = 0,
  alignment = "horizontal",
  has_labels = TRUE,
  label_position = "center",
  decimals = 0,
  display_limits = c(0, 0)
)
```

Arguments

svg	XML document with SVG content
frame_name	Name (attribute 'id') of frame (XML element 'rect') for positioning bar segments (and value labels).
group_name	Name (attribute 'id') of group (XML element 'g') containing either bar segments (and value labels) or further groups, containing bar segments (and value labels) themselves.
scale_real	Numeric vector (e.g. `c(0,100)`) of arbitrary length. Only minimum and maximum are used for scaling of values.
values	Either a numeric vector, a numeric matrix or a dataframe with only numeric columns. If a vector is given, it corresponds to one bar (group of bar segments and, optionally, value labels). If a matrix or dataframe is given, rows define the value set for several (stacked) bars grouped together.
nullvalue	Value that defines the "center" of the bar segments (for left/right or bottom/top positioning)
alignment	Character value. Accepts 'horizontal' (default) or 'vertical'. See details.
has_labels	Are there value labels (of XML type 'text') to adjust? (default TRUE)
label_position	Character value. Accepts 'start', 'center' (default) and 'end'. This refers to the underlying bar segments.
decimals	Integer value defining the number of decimal digits of value labels (default 0).
display_limits	Interval for (small) values, that lead to suppression of the corresponding value labels. If only one value x is given, it is turned into the interval c(-x,x). (default 0 = no suppression)

Details

See `stackedBar`.

Value

XML document with SVG content

Examples

```
#read SVG file
fpath <- system.file("extdata", "fig7.svg", package="svgtools")
svg <- read_svg(file = fpath)

#adjust bars
df <- data.frame(diff_negative = c(NA,NA,-0.7,NA,-0.33),
                 diff_positive = c(0.4,0.55,NA,0.02,NA))
svg <- diffBar(svg = svg, frame_name = "frame", group_name = "group",
               scale_real = c(-1,1), values = df, nullvalue = 0,
               label_position = "end", decimals = 1, display_limits = 0.1)
```

display_svg *Display SVG on standard graphic display port*

Description

Display SVG on standard graphic display port

Usage

```
display_svg(svg, width = NULL, height = NULL)
```

Arguments

svg	XML document with SVG content.
width	Desired width (in px) of image (default NULL).
height	Desired height (in px) of image (default NULL).

Details

Viewport depends on system and IDE. In RStudio the image is displayed under 'Viewer'.
If neither width nor height are specified the image will have its size depending on DPI settings. If only one of these is specified, the other one is scaled accordingly.

Examples

```
#read SVG file
fpath <- system.file("extdata", "fig1.svg", package="svgtools")
svg <- read_svg(file = fpath)

#display SVG file in standard viewport
display_svg(svg = svg, width = 500)
```

linesSymbols *Adjust line and/or symbol charts*

Description

Adjusts the horizontal (XML attributes 'x', 'x1', 'x2', 'cx' etc.) or vertical (XML attributes 'y', 'y1', 'y2', 'cy' etc.) position of lines (XML elements of type 'line') and/or symbols (see details). Positions are calculated relative to a given frame (XML element of type 'rect') and the position of a data value within the minimum and maximum of a given scale. This process is called scaling.
In preparation, it is necessary to name a group (set attribute 'id' of XML element of type 'g') of lines and/or symbols.

Usage

```
linesSymbols(
  svg,
  frame_name,
  group_name,
  scale_real,
  values,
  alignment = "vertical",
  has_lines = TRUE,
  symbol_type = NULL,
  ...
)
```

Arguments

svg	XML document with SVG content
frame_name	Name (attribute 'id') of frame (XML element 'rect') for positioning of elements.
group_name	Name (attribute 'id') of group (XML element 'g') with lines and/or symbols.
scale_real	Numeric vector (e.g. `c(0,100)`) of arbitrary length. Only minimum and maximum are used for scaling of values.
values	Numeric vector. If a dataframe or matrix is provided, only the first row will be used.
alignment	Character value. Accepts 'horizontal' or 'vertical' (default). See details.
has_lines	Are there lines? (default TRUE)
symbol_type	Character value. Accepts 'circle', 'rect', 'polygon' or 'linegroup'; see details. (default NULL = no symbols)
...	Further arguments used internally by `scatterSymbols`.

Details

Note: 'Horizontal' alignment refers to adjustment of the x-coordinates of elements, 'vertical' alignment to adjustment of the y-coordinates. This is not to be confused with the orientation of the resulting polyline (and/or the sequence of symbols) that goes from left to right (with `alignment='vertical'`) or from top to bottom (with `alignment='horizontal'`).
Line elements and/or symbols may be grouped together in any order in the SVG file. The function will automatically use XML elements from left to right (with `alignment='vertical'`) or top to bottom (with `alignment='horizontal'`) according to their x/y-coordinates.
Line elements and/or symbols must be prepared in the SVG file in one of the following amounts: a) same amount as data values (or one element less in case of lines), b) one line and/or or two symbols or c) one line *and* one symbol. In case of b) and c) the function will automatically duplicate line elements and/or symbols with the assumption of fixed shapes (that is, all lines and/or all symbols will look the same as the 'template' that is provided) and constant distance between the elements on the coordinate that will not be adjusted by `values` ('x' with `alignment='vertical'` or 'y' with `alignment='horizontal'`).
The function currently supports the following `symbol_types`:

- circle: XML elements of type 'circle'. Attributes 'cx' or 'cy' are adjusted.

- rect: XML elements of type 'rect'. Attributes 'x' or 'y' are adjusted.
- polygon: XML elements of type 'polygon'. Attribute 'points' is adjusted so that the centroid of the shape matches the scaled value position on the chart.
- linegroup: XML elements of type 'g' that contain elements of type 'line'. Attributes 'x1' and 'x2' or 'y1' and 'y2' of those lines are adjusted so that the mean x- or y-coordinate of all lines in the group matches the scaled value position on the chart.

Value

XML document with SVG content

Examples

```
#read SVG file
fpath <- system.file("extdata", "fig11.svg", package="svgtools")
svg <- read_svg(file = fpath)

#adjust lines and/or symbols
set.seed(12345)
values <- matrix(c(rnorm(10,0.95,0.03), rnorm(10,0.75,0.05),
                   rnorm(10,0.55,0.07), rnorm(10,0.35,0.05),
                   rnorm(10,0.15,0.03)), nrow = 5, byrow = TRUE)
values[2,8] <- as.numeric(NA)
svg <- linesSymbols(svg = svg, frame_name = "frame", group_name = "gA",
                    scale_real = c(0,1), values = values[1,],
                    symbol_type = "rect")
svg <- linesSymbols(svg = svg, frame_name = "frame", group_name = "gB",
                    scale_real = c(0,1), values = values[2,],
                    symbol_type = "circle")
svg <- linesSymbols(svg = svg, frame_name = "frame", group_name = "gC",
                    scale_real = c(0,1), values = values[3,],
                    has_lines = FALSE, symbol_type = "polygon")
svg <- linesSymbols(svg = svg, frame_name = "frame", group_name = "gD",
                    scale_real = c(0,1), values = values[4,],
                    symbol_type = "linegroup")
svg <- linesSymbols(svg = svg, frame_name = "frame", group_name = "gE",
                    scale_real = c(0,1), values = values[5,],
                    symbol_type = NULL)
```

---

percentileBar *Adjust (stacked) bar chart representing percentiles*

Description

Adjusts the horizontal (XML attribute 'x') or vertical (XML attribute 'y') position as well as width/height of bar segments (XML elements of type 'rect') and optionally value labels (XML elements of type 'text'). First, values are ordered and transformed to differences between percentiles, providing a starting point and widths/heights for bar segments. Then, positions are calculated relative to a given frame (XML element of type 'rect') and the position of a value within the minimum

and maximum of a given scale.
No value labels are possible for this type of bar chart.
For further description see stackedBar.

Usage

```
percentileBar(
  svg,
  frame_name,
  group_name,
  scale_real,
  values,
  alignment = "horizontal"
)
```

Arguments

svg	XML document with SVG content
frame_name	Name (attribute 'id') of frame (XML element 'rect') for positioning bar segments.
group_name	Name (attribute 'id') of group (XML element 'g') containing either bar segments or further groups, containing bar segments themselves.
scale_real	Numeric vector (e.g. c(0,100)) of arbitrary length. Only minimum and maximum are used for scaling of values.
values	Either a numeric vector, a numeric matrix or a dataframe with only numeric columns. If a vector is given, it corresponds to one bar (group of bar segments). If a matrix or dataframe is given, rows define the value set for several (stacked) bars grouped together.
alignment	Character value. Accepts 'horizontal' (default) or 'vertical'. See details.

Details

In contrast to stackedBar the value set(s) need(s) to have one more element than bar segments in each group.
For everything else, see stackedBar.

Value

XML document with SVG content

Examples

```
#read SVG file
fpath <- system.file("extdata", "fig9.svg", package="svgtools")
svg <- read_svg(file = fpath)

#adjust bars
set.seed(12345)
```

```
yy <- -2:2*15
percentiles <- data.frame(p5=rnorm(5,350,10)+yy, p25=rnorm(5,450,10)+yy,
                          p50low=rnorm(5,510,10)+yy-1.5, p75=rnorm(5,560,10)+yy,
                          p95=rnorm(5,640,10)+yy)
percentiles$p50upp <- percentiles$p50low+3
svg <- percentileBar(svg = svg, frame_name = "frame", group_name = "group",
                     scale_real = c(250,750), values = percentiles)
```

`read_svg` *Read SVG file and return XML document*

Description

Read SVG file and return XML document

Usage

```
read_svg(file, enc = "UTF-8", summary = FALSE, display = FALSE)
```

Arguments

`file`	A string, a connection, or a raw vector. See `read_xml`.
`enc`	Encoding (default 'UTF-8').
`summary`	Show summary of SVG file (`summary_svg`)? (default FALSE)
`display`	Display SVG on standard display port (`display_svg`)? (default FALSE)

Value

XML document with SVG content

Examples

```
fpath <- system.file("extdata", "fig1.svg", package="svgtools")
svg <- read_svg(file = fpath, summary = TRUE)
```

`referenceBar` *Adjust (stacked) bar chart that is aligned around a reference category*

Description

Adjusts the horizontal (XML attribute 'x') or vertical (XML attribute 'y') position as well as width/height of bar segments (XML elements of type 'rect') and optionally value labels (XML elements of type 'text'). Positions are calculated relative to a given frame (XML element of type 'rect'), a nullvalue and the position of a data value within the minimum and maximum of a given scale. The first n bar segments and, optionally, value labels of each bar (n is called the `reference` category) are position to the left (in horizontal alignment) or bottom (in vertical alignment) of the nullvalue, while the others are positioned to the right or top.
For further description see `stackedBar`.

Usage

```
referenceBar(
  svg,
  frame_name,
  group_name,
  scale_real,
  values,
  reference,
  nullvalue = 0,
  alignment = "horizontal",
  has_labels = TRUE,
  label_position = "center",
  decimals = 0,
  display_limits = 0
)
```

Arguments

`svg`	XML document with SVG content
`frame_name`	Name (attribute 'id') of frame (XML element 'rect') for positioning bar segments (and value labels).
`group_name`	Name (attribute 'id') of group (XML element 'g') containing either bar segments (and value labels) or further groups, containing bar segments (and value labels) themselves.
`scale_real`	Numeric vector (e.g. `c(0,100)`) of arbitrary length. Only minimum and maximum are used for scaling of values.
`values`	Either a numeric vector, a numeric matrix or a dataframe with only numeric columns. If a vector is given, it corresponds to one bar (group of bar segments and, optionally, value labels). If a matrix or dataframe is given, rows define the value set for several (stacked) bars grouped together.

reference	Reference category (=column number of values). Bar segments up to this category lie to the left (horizontal) or to the bottom (vertical), bar segments above this category lie to the right (horizontal) or the top (vertical) of the bar chart.
nullvalue	Value that defines the "center" of the bar segments (for left/right or bottom/top positioning)
alignment	Character value. Accepts 'horizontal' (default) or 'vertical'. See details.
has_labels	Are there value labels (of XML type 'text') to adjust? (default TRUE)
label_position	Character value. Accepts 'start', 'center' (default) and 'end'. This refers to the underlying bar segments.
decimals	Integer value defining the number of decimal digits of value labels (default 0).
display_limits	Interval for (small) values, that lead to suppression of the corresponding value labels. If only one value x is given, it is turned into the interval c(-x,x). (default 0 = no suppression)

Details

See stackedBar.

Value

XML document with SVG content

Examples

```
#read SVG file
fpath <- system.file("extdata", "fig5.svg", package="svgtools")
svg <- read_svg(file = fpath)

#adjust bars
values <- matrix(c(1,2,3,4,2,3,4,1,3,4,1,2,4,1,2,3,1,2,3,4)*10,
                 nrow = 5, byrow = TRUE)
svg <- referenceBar(svg = svg, frame_name = "frame", group_name = "group",
                    scale_real = c(-100,100), values = values,
                    reference = 2, nullvalue = 0)
```

---

scatterSymbols *Adjust symbols of a scatter plot*

Description

Adjusts the horizontal (XML attributes 'x', 'x1', 'x2', 'cx' etc.) and vertical (XML attributes 'y', 'y1', 'y2', 'cy' etc.) positions of symbols (see details). Positions are calculated relative to a given frame (XML element of type 'rect') and the position of a data value within the minimum and maximum of two given scales for x- and y-axis. This process is called scaling.
In preparation, it is necessary to name a group (set attribute 'id' of XML element of type 'g') of symbols. Symbols are automatically duplicated or removed to match the amount of data values.

Usage

```
scatterSymbols(
  svg,
  frame_name,
  group_name,
  scale_real_x,
  scale_real_y,
  values,
  symbol_type
)
```

Arguments

svg	XML document with SVG content
frame_name	Name (attribute 'id') of frame (XML element 'rect') for positioning of elements.
group_name	Name (attribute 'id') of group (XML element 'g') with symbols.
scale_real_x	Numeric vector (e.g. `c(0,100)`) of arbitrary length for x-axis. Only minimum and maximum are used for scaling of values.
scale_real_y	Numeric vector (e.g. `c(0,100)`) of arbitrary length for y-axis. Only minimum and maximum are used for scaling of values.
values	Dataframe or matrix with numeric vectors. First column corresponds to x-axis. Second column corresponds to y-axis.
symbol_type	Character value. Accepts 'circle', 'rect', 'polygon' or 'linegroup'; see details.

Details

Symbols may be prepared in the SVG file in any amount. But be aware that the function will simply duplicate the first one (in the group) or remove the last ones to match the amount of data values. When, for example, you need to have different colors for different subgroups of cases, prepare several groups of symbols and call this function for each of them.
The function currently supports the following `symbol_types`:

- circle: XML elements of type 'circle'. Attributes 'cx' and 'cy' are adjusted.
- rect: XML elements of type 'rect'. Attributes 'x' and 'y' are adjusted.
- polygon: XML elements of type 'polygon'. Attribute 'points' is adjusted so that the centroid of the shape matches the scaled value positions on the chart.
- linegroup: XML elements of type 'g' that contain elements of type 'line'. Attributes 'x1', 'x2', 'y1' and 'y2' of those lines are adjusted so that the mean x- and y-coordinate of all lines in the group matches the scaled value positions on the chart.

Value

XML document with SVG content

Examples

```
#read SVG file
fpath <- system.file("extdata", "fig13.svg", package="svgtools")
svg <- read_svg(file = fpath)

#scatter symbols
set.seed(12345)
df <- data.frame(g=rep(1:4,10), x=rnorm(40,500,75), y=rnorm(40,500,75))
df[df$g==1,]$x <- df[df$g==1,]$x-35
df[df$g==2,]$y <- df[df$g==2,]$y-35
df[df$g==3,]$x <- df[df$g==3,]$x+35
df[df$g==4,]$y <- df[df$g==4,]$y+35
svg <- scatterSymbols(svg = svg, frame_name = "frame", group_name = "gA",
                      scale_real_x = c(250,750), scale_real_y = c(250,750),
                      values = df[df$g==1,2:3], symbol_type = "rect")
svg <- scatterSymbols(svg = svg, frame_name = "frame", group_name = "gB",
                      scale_real_x = c(250,750), scale_real_y = c(250,750),
                      values = df[df$g==2,2:3], symbol_type = "circle")
svg <- scatterSymbols(svg = svg, frame_name = "frame", group_name = "gC",
                      scale_real_x = c(250,750), scale_real_y = c(250,750),
                      values = df[df$g==3,2:3], symbol_type = "polygon")
svg <- scatterSymbols(svg = svg, frame_name = "frame", group_name = "gD",
                      scale_real_x = c(250,750), scale_real_y = c(250,750),
                      values = df[df$g==4,2:3], symbol_type = "linegroup")
```

---

stackedBar *Adjust (stacked) bar chart to values on a given scale*

Description

Adjusts the horizontal (XML attribute 'x') or vertical (XML attribute 'y') position as well as width/height of bar segments (XML elements of type 'rect') and optionally value labels (XML elements of type 'text'). Positions are calculated relative to a given frame (XML element of type 'rect') and the position of a data value within the minimum and maximum of a given scale. This process is called scaling.

In preparation, it is necessary to name a group (set attribute 'id' of XML element of type 'g') of bar segments (and value labels). Bar segments (and value labels) need to be of the same amount as there are data values for adjustment.

It is possible to group several such groups together. Only the outer group needs to be named in that case, for convenience.

Usage

```
stackedBar(
  svg,
  frame_name,
  group_name,
  scale_real,
```

```
  values,
  alignment = "horizontal",
  has_labels = TRUE,
  label_position = "center",
  decimals = 0,
  display_limits = 0,
  ...
)
```

Arguments

svg	XML document with SVG content
frame_name	Name (attribute 'id') of frame (XML element 'rect') for positioning bar segments (and value labels).
group_name	Name (attribute 'id') of group (XML element 'g') containing either bar segments (and value labels) or further groups, containing bar segments (and value labels) themselves.
scale_real	Numeric vector (e.g. `c(0,100)`) of arbitrary length. Only minimum and maximum are used for scaling of values.
values	Either a numeric vector, a numeric matrix or a dataframe with only numeric columns. If a vector is given, it corresponds to one bar (group of bar segments and, optionally, value labels). If a matrix or dataframe is given, rows define the value set for several (stacked) bars grouped together.
alignment	Character value. Accepts 'horizontal' (default) or 'vertical'. See details.
has_labels	Are there value labels (of XML type 'text') to adjust? (default TRUE)
label_position	Character value. Accepts 'start', 'center' (default) and 'end'. This refers to the underlying bar segments.
decimals	Integer value defining the number of decimal digits of value labels (default 0).
display_limits	Interval for (small) values, that lead to suppression of the corresponding value labels. If only one value x is given, it is turned into the interval c(-x,x). (default 0 = no suppression)
...	Further arguments used internally by `referenceBar`, `diffBar` and `percentileBar`.

Details

'Horizontal' alignment refers to adjustment of the x-coordinates of elements, 'vertical' alignment to adjustment of the y-coordinates.
Bar segments and, optionally, value labels may be grouped together in any order in the SVG file. The function will automatically use XML elements from left to right (with `alignment='horizontal'`) or bottom to top (with `alignment='vertical'`) according to their x/y-coordinates.
Furthermore, the SVG file order of several bars grouped together in an outer group is irrelevant. The function will automatically use bars (that is, groups of bar segments and, optionally, value labels) from top to bottom (with `alignment='horizontal'`) or left to right (with `alignment='vertical'`) according to the lowest x/y-coordinate of any element.

Bar segments and value labels (if any) are automatically hidden (XML attribute 'diplay' is set to 'none'), when a value of 0 or NA is provided. Subsequent calls to the function with non-zero or non-NA values make such elements reappear in the output.

Value

XML document with SVG content

Examples

```
#read SVG file
fpath <- system.file("extdata", "fig3.svg", package="svgtools")
svg <- read_svg(file = fpath)

#adjust bars
svg <- stackedBar(svg = svg, frame_name = "frame", group_name = "overall",
                  scale_real = c(0,160), values = c(9.97,42.42,105.71),
                  alignment = "vertical", has_labels = TRUE,
                  label_position = "end", decimals = 0, display_limits = 10)
df.subgroups <- matrix(1:9*8, nrow=3)
svg <- stackedBar(svg = svg, frame_name = "frame", group_name = "subgroups",
                  scale_real = c(0,160), values = df.subgroups,
                  alignment = "vertical", display_limits = 10)
```

---

summary_svg *Print summary of SVG file structure in console*

Description

Print summary of SVG file structure in console

Usage

```
summary_svg(svg)
```

Arguments

svg	XML document with SVG content.

Details

Prints helpful information to verify the content of the SVG file:

- Named groups (XML elements 'g' with attribute 'id') and number of their child elements
- Available frames (XML elements 'rect' with attribute 'id')
- Used fonts, font sizes and font colors (in any XML elements with attributes 'font-family', 'font-size', 'fill' and 'stroke')

Examples

```
#read SVG file
fpath <- system.file("extdata", "fig1.svg", package="svgtools")
svg <- read_svg(file = fpath)

#show a summary of SVG file
summary_svg(svg = svg)
```

`write_svg` *Writes SVG to file*

Description

Writes SVG to file

Usage

```
write_svg(svg, file, remove_hidden = TRUE, flatten = FALSE)
```

Arguments

`svg`	XML document with SVG content.
`file`	Path to file or connection to write to (see `write_xml`).
`remove_hidden`	Should hidden elements (with XML attribute display="none") be removed? (default TRUE)
`flatten`	Should grouping of SVG elements be removed? (default FALSE)

Details

Both `remove_hidden=TRUE` and `flatten=TRUE` do not alter the XML document object itself. Therefore, subsequent calls to `stackedBar` and other functions remain possible.

Examples

```
#read SVG file
fpath <- system.file("extdata", "fig3.svg", package="svgtools")
svg <- read_svg(file = fpath)

#adjust some elements of SVG
svg <- stackedBar(svg = svg, frame_name = "frame", group_name = "overall",
                  scale_real = c(0,160), values = c(10,42,106),
                  alignment = "vertical")

## Not run:
#write SVG file to disk and remove all groupings
write_svg(svg = svg, file = "myChart.svg", flatten = TRUE)

## End(Not run)
```

Index