Package ‘googleVis’

May 10, 2022

Type Package

Title R Interface to Google Charts

Version 0.7.0

Date 2022-05-10

Description R interface to Google's chart tools, allowing users to create interactive charts based on data frames. Charts are displayed locally via the R HTTP help server. A modern browser with an Internet connection is required. The data remains local and is not uploaded to Google.

Depends R (>= 3.0.2)

Imports methods,
    jsonlite,
    utils

Suggests shiny (>= 0.4.0),
    httpuv (>= 1.2.0),
    knitr (>= 1.5),
    rmarkdown,
    markdown,
    data.table

License GPL (>= 2)

URL https://mages.github.io/googleVis/

BugReports https://github.com/mages/googleVis/issues

LazyLoad yes

LazyData yes

VignetteBuilder knitr, rmarkdown

RoxygenNote 7.1.2

Encoding UTF-8

R topics documented:

googleVis-package .......................................................... 2
Andrew ................................................................. 3
Cairo ................................................................. 4
Cats ................................................................. 5
R interface to Google Charts API, allowing users to create interactive charts based on data frames. Charts are displayed locally via the R HTTP help server. A modern browser with Internet connection is required and for some charts Flash. The data remains local and is not uploaded to Google.

Please visit the project web site for more information: https://github.com/mages/googleVis.
You find further notes on Markus’ blog: https://magesblog.com/tags/googlevis/
Andrew

Note
See vignette("googleVis") for an introduction to the googleVis package.

Author(s)
Markus Gesmann, Diego de Castillo

References
• Google Charts API: https://developers.google.com/chart/
• Google Terms of Use: https://developers.google.com/terms/
• Google Maps API Terms of Service: https://cloud.google.com/maps-platform/terms/

Examples
## Not run:
demo(googleVis)
## For other demos see
demo(package='googleVis')
## End(Not run)

Andrew Hurricane Andrew: googleVis example data set

Description
Hurricane Andrew storm path from 16 August to 28 August 1992

Usage
data(Andrew)

Format
A data frame with 47 observations on the following 8 variables.

Date/Time UTC a POSIXct
Lat a numeric vector
Long a numeric vector
Pressure_mb a numeric vector
Speed_kt a numeric vector
Category a factor with levels Hurricane Tropical Depression Tropical Storm
LatLong a character vector
Tip a character vector

Source
National Hurricane Center: https://www.nhc.noaa.gov/1992andrew.html
Examples

data(Andrew)

AndrewGeoMap <- gvisGeoMap(Andrew, locationvar='LatLong', numvar='Speed_kt',
    hovervar='Category',
    options=list(width=800,height=400,
        region='US', dataMode='Markers'))

AndrewMap <- gvisMap(Andrew, 'LatLong', 'Tip',
    options=list(showTip=TRUE, showLine=TRUE,
        enableScrollWheel=TRUE,
        mapType='hybrid', useMapTypeControl=TRUE,
        width=800,height=400))

AndrewTable <- gvisTable(Andrew,options=list(width=800))

## Combine the outputs into one page:
AndrewVis <- gvisMerge(AndrewGeoMap, AndrewMap)

plot(AndrewVis)

---

Cairo

Daily temperature data for Cairo

Description

The average air temperature (C) in Cairo from 2002 to 2004.

Usage

data(Cairo)

Format

A data frame with 1091 observations on the following 2 variables.

Date calendar date
Temp average temperatur in Celsius

References


Examples

data(Cairo)
plot(gvisCalendar(Cairo))
**Cats**

**Cat Search Terms**

**Description**
A set of responses for search terms for cats

**Usage**
data(Cats)

**Format**
A data frame with 18 observations on the following 3 variables.

- **Phrase**  Search term for cats
- **Size**   Number of times term has been searched
- **Sentiment**  Numeric value from 0-10 of sentiment of term

**Examples**
data(Cats)
plot(gvisWordTree(Cats, textvar = "Phrase"))

---

**CityPopularity**  

**CityPopularity: googleVis example data set**

**Description**
Example data set to illustrate the use of the googleVis package.

**Usage**
data(CityPopularity)

**Format**
A data frame with 6 observations on the following 2 variables.

- **City**  a factor with levels Boston Chicago Houston Los Angeles Miami New York
- **Popularity**  a numeric vector

**Source**
Google Geo Map API: [https://developers.google.com/chart/interactive/docs/gallery/geomap.html](https://developers.google.com/chart/interactive/docs/gallery/geomap.html)

**Examples**
data(CityPopularity)
str(CityPopularity)
createGoogleGadget  

Create a Google Gadget

Description
Create a Google Gadget based on a Google Visualisation Object

Usage
createGoogleGadget(gvis)

Arguments
gvis  
an object of class 'gvis', e.g. output of a googleVis visualisation functions.

Value
createGoogleGadget returns a Google Gadget XML string.

Note
Google Gadgets can be embedded in various Google products, for example as part of a Google Code wiki page, Blogger or Google Sites. In all cases the XML gadget file has to be hosted online, e.g. using Google Docs.

In Blogger the gadgets can be embedded via the design tab, and in a Google Sites via the menu "Insert" -> "More gadgets ..." -> "Add gadget ULR".

In a Google Code wiki the gadget can be embedded via

<wiki:gadget url="https://example.com/gadget.xml" height="200" border="0" />

Author(s)
Markus Gesmann

References
For more information about Google Gadgets see: https://developers.google.com/gadgets/

See Also
See also as print.gvis, cat

Examples
M <- gvisMotionChart(Fruits, idvar="Fruit", timevar="Year")
gdgt <- createGoogleGadget(M)
cat(gdgt)
**dino**

**Dinosaur data**

**Description**

Length of dinosaurs

**Usage**

```r
data(dino)
```

**Format**

A data frame with 28 observations on 2 variables, dinosaur and length.

**Source**

[https://developers.google.com/chart/interactive/docs/gallery/histogram](https://developers.google.com/chart/interactive/docs/gallery/histogram)

**Examples**

```r
data(dino)
str(dino)
```

---

**Exports**

**Exports: googleVis example data set**

**Description**

Example data set to illustrate the use of the googleVis package.

**Usage**

```r
data(Exports)
```

**Format**

A data frame with 10 observations on the following 3 variables.

- **Country**: a factor with levels Brazil, Germany ...
- **Profit**: a numeric vector
- **Online**: a logical vector

**Examples**

```r
data(Exports)
str(Exports)
```
Fruits: googleVis example data set

Description

Example data set to illustrate the use of the googleVis package.

Usage

data(Fruits)

Format

A data frame with 9 observations on the following 7 variables.

- **Fruit**: a factor with levels Apples Bananas Oranges
- **Year**: a numeric vector
- **Location**: a factor with levels East West
- **Sales**: a numeric vector
- **Expenses**: a numeric vector
- **Profit**: a numeric vector
- **Date**: a Date

Examples

data(Fruits)
M <- gvisMotionChart(Fruits, idvar="Fruit", timevar="Year")

## Not run:
plot(M)
## End(Not run)

---

Print and plot gvis objects

Description

Methods to print and plot gvis objects

Usage

## S3 method for class 'gvis'
print(x, tag=NULL, file = "", ...)

## S3 method for class 'gvis'
plot(x, tag=NULL,...)
Arguments

- **x**: An object of class `gvis`, or a HTML file in case of `plot.gvis`.
- **tag**: Default `NULL`. Name tag of the objects to be extracted from a `gvis` object. If `tag` is missing then the values of `getOption("gvis.print.tag"), getOption("gvis.plot.tag")` will be used for `print.gvis` and `plot.gvis` respectively. A complete list of available tags is given by the command `getOption("gvis.tags")`. The default value of `gvis.print.tag` is "html", which means `print.gvis` will show a complete web page with the visualisation, while the tag "chart" will present the code for the visualisation chart only. For more information see the details section.

The default tag for `plot.gvis.tag` is `NULL`, which will result in R opening a browser window, while any tag which is not `NULL` will give the same behaviour as `print.gvis` for `tag='chart'` or setting `options(gvis.plot.tag='chart')` will produce the same output as `print(x, tag='chart')`. This behaviour is particular helpful when `googleVis` is used in scripts, like `knitr` or `R.rsp`. The plot commands can be used initially in an interactive mode and with one change in `options()` produce the HTML output required for a programmatic run of the script. See the example section below for a `knitr` case study.

- **file**: file name. If "" (the default), output will be printed to the standard output connection, the console unless redirected by `sink`.
- **...**: arguments passed on to `cat` (print.gvis) or `browseURL` (plot.gvis).

Details

An object of class "gvis" is a list containing at least the following components (tags):

- **type**: Google visualisation type, e.g. 'MotionChart'
- **chartid**: character id of the chart object. Unique chart ids are required to place several charts on the same page.
- **html**: a list with the building blocks for a page
  - **header**: a character string of a html page header: `<html>...<body>`,
  - **chart**: a named character vector of the chart's building blocks:
    - **jsHeader**: Opening `<script>` tag and reference to Google's JavaScript library.
    - **jsData**: JavaScript function defining the input data as a JSON object.
    - **jsDrawChart**: JavaScript function combing the data with the visualisation API and user options.
    - **jsDisplayChart**: JavaScript function calling the handler to display the chart.
    - **jsFooter**: End tag `</script>`.
    - **jsChart**: Call of the `jsDisplayChart` function.
  - **divChart**: `<div>` container to embed the chart into the page.
  - **caption**: character string of a standard caption, including data name and chart id.
  - **footer**: character string of a html page footer: `</body>...</html>`, including the used R and `googleVis` version and link to Google's Terms of Use.

Value

- **print.gvis**: None (invisible `NULL`).
- **plot.gvis**: Returns the file name invisibly.
Note

The plot command does not open a graphics device in the traditional way. Instead it creates HTML files in a temporary directory and uses the R HTTP server to display the output of a googleVis function locally. A browser with Flash and Internet connection is required. The displayed page includes a link (click on the chart id) to a further page, which shows the code of the chart for the user to copy and paste into her own page.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

References

Please see also the package vignette for the usage of the googleVis package with RApache, brew, knitr and R.rsp.

See Also

See also cat, browseURL, createGoogleGadget and gvisMerge for combining charts.

Examples

```r
## Show gvis options
sapply(c("gvis.print.tag", "gvis.plot.tag", "gvis.tags"),getOption)
M <- gvisMotionChart(Fruits, "Fruit", "Year")
str(M)
## The output for a complete web page
M

## Access only the plot,
M$html$chart

## wrap it in cat and it becomes more readable,
cat(unlist(M$html$chart))

## or use the print function.
print(M, tag="chart")

## Extract the data as a JavaScript function.
print(M, tag="jsData")

## Display the visualisation.
## A web browser with Internet connection and Flash is required.
plot(M)

## Combine with another chart, e.g. table
#tbl <- gvisTable(Fruits, options=list(height=220))
#Mtbl <- gvisMerge(M, tbl)
#plot(Mtbl)

## Example of using googleVis with knitr and markdown
```
## Not run:
## Simple knitr/markdown file with googleVis
knitrRmd <-
# Markdown example with knitr and googleVis
===========================================
This is a little Markdown example file.
Set the googleVis options first.
In this case change the behaviour of plot.gvis
```
(r setOptions, message=FALSE)
library(googleVis)
op <- options(gvis.plot.tag='chart')
```
The following plot statements will automatically return the HTML
required for the 'knitted' output.

## Combo chart
```
(r ComboExample, results='asis', tidy=FALSE)
## Add the mean
CityPopularity$Mean=mean(CityPopularity$Popularity)
CC <- gvisComboChart(CityPopularity, xvar='City',
yvar=c('Mean', 'Popularity'),
options=list(seriesType='bars',
width=450, height=300,
title='City Popularity',
series='{0: {type:"line"}}'))
plot(CC)
```
Example of gvisComboChart with R code shown above.

## Place two charts next to each other
```
(r gvisMergeExample, results='asis', echo=FALSE)
Geo <- gvisGeoChart(Exports, locationvar='Country', colorvar='Profit',
options=list(height=300, width=350))
Tbl <- gvisTable(Exports, options=list(height=300, width=200))
plot(gvisMerge(Geo, Tbl, horizontal=TRUE))
```
Example of a gvisGeoChart with gvisTable and R code hidden.

## Motion Chart
```
(r MotionChartExample, results='asis', tidy=FALSE)
M <- gvisMotionChart(Fruits, 'Fruit', 'Year',
options=list(width=400, height=350))
plot(M)
```
Please note that the Motion Chart is only displayed when hosted on a
web server, or is placed in a directory which has been added to the
trusted sources in the [security settings of Macromedia]
See the googleVis package vignette for more details.
```
(r resetOptions)
## Set options back to original options
options(op)
```

## Write the content of knitrRmd into a Rmd-file, knit it and convert it
## gvis Methods

### into a html output. Finally show the file with the R-help http
### server, this will ensure that also the motion chart is visible.

```r
library(knitr)
library(markdown)

wd <- getwd()
setwd(tempdir())
fn=tempfile()
fn.Rmd <- paste(fn, ".Rmd", sep="")
fn.md <- paste(fn, ".md", sep="")
fn.html <- paste(fn, ".out.html", sep="")
## Write R Markdown into a file
cat(knitrRmd, file=fn.Rmd)
render_markdown()
knit(fn.Rmd, fn.md)
knit2html(fn.md)

## Open output in browser
## Use plot.gvis which will use the R-help http server
## The URL will start with http://127.0.0.1...
## The HTML file will be copied into a temporary directory
plot.gvis(fn.html)
## Compare to browseURL, its URL will start with file://... the motion
## chart is unlikely to be displayed because of Flash security
## settings. See the googleVis vignette for more details.
browseURL(fn.html)
setwd(wd)
```

## Not run:

## Updating the data of an existing googleVis web page

## Suppose you have an existing web page in which you embedded a
## motion chart with the id "mtnc".
## Now you have a new set of data, but you would like to avoid
## touching the html file again.
## The idea is to write the data and JavaScript functions into separate
## files and to refer to these in the html page.

## In this example we call the chart id "mtnc"
## To display the example we use the R HTTP server again, and
## write the files into a temp directory

```r
myChartID <- "mtnc"
## baseURL should reflect your web address, e.g. http://myHomePage.com
baseURL <- sprintf("http://127.0.0.1:%s/custom/googleVis", tools::.httpdPort)
wwwdir <- tempdir()  ## the www repository on your computer

## Create a motion chart
M=gvisMotionChart(Fruits, "Fruit", "Year", chartid=myChartID)

## Here is our plot again
plot(M)
```
## gvisAnnotatedTimeLine

Google Annotated Time Line with R

### Description

The `gvisAnnotatedTimeLine` function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page.
Usage

gvisAnnotatedTimeLine(
    data,
    datevar = "",
    numvar = "",
    idvar = "",
    titlevar = "",
    annotationvar = "",
    date.format = "%Y/%m/%d",
    options = list(),
    chartid
)

Arguments

data a data.frame. The data has to have at least two columns, one with date information (datevar) and one numerical variable.
datevar column name of data which shows the date dimension. The information has to be of class Date or POSIX* time series.
numvar column name of data which shows the values to be displayed against datevar. The information has to be numeric.
idvar column name of data which identifies different groups of the data. The information has to be of class character or factor.
titlevar column name of data which shows the title of the annotations. The information has to be of class character or factor. Missing information can be set to NA. See section 'Details' for more details.
annotationvar column name of data which shows the annotation text. The information has to be of class character or factor. Missing information can be set to NA. See section 'Details' for more details.
date.format if datevar is of class Date then this argument specifies how the dates are reformatted to be used by JavaScript.
options list of configuration options, see: https://developers.google.com/chart/interactive/docs/gallery/annotatedtimeline#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

• Boolean arguments are set to either TRUE or FALSE, using the R syntax.
• Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is options=list(width=200,height=300). Exceptions to this rule are the width and height options for gvisAnnotatedTimeline and gvisAnnotationChart. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options=list(width="200px",height="300px").
• Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g. options=list(colors="["#cbb69d","#603913","#c69c6e"]")
• Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in "
"gvisAnnotatedTimeLine" returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

- **type**: Google visualisation type
- **chartid**: character id of the chart object. Unique chart ids are required to place several charts on the same page.
- **html**: a list with the building blocks for a page
  - **header**: a character string of a html page header: `<html>...<body>`,
  - **chart**: a named character vector of the chart’s building blocks:
    - **jsHeader**: Opening `<script>` tag and reference to Google’s JavaScript library.
    - **jsData**: JavaScript function defining the input data as a JSON object.
    - **jsDrawChart**: JavaScript function combing the data with the visualisation API and user options.
    - **jsDisplayChart**: JavaScript function calling the handler to display the chart.
    - **jsFooter**: End tag `</script>`.
    - **jsChart**: Call of the jsDisplayChart function.
  - **divChart**: `<div>` container to embed the chart into the page.
- **caption**: character string of a standard caption, including data name and chart id.
- **footer**: character string of a html page footer: `</body>...</html>`, including the used R and googleVis version and link to Google’s Terms of Use.

Details

An annotated time line is an interactive time series line chart with optional annotations. The chart is rendered within the browser using Flash.

Value

gvisAnnotatedTimeLine returns list of class "gvis" and "list". Boolean values have to be stated as 'true' or 'false'. For example the Google documentation states the formatting options for the vertical axis and states the parameter as `vAxis.format`. Then this parameter can be set in R as:

```r
options=list(vAxis="(format:'#,###\%')")
```

- If several sub-options have to be set, e.g. `titleTextStyle.color`, `titleTextStyle.fontName` and `titleTextStyle.fontSize`, then those can be combined in one list item such as:

```r
options=list(titleTextStyle="(color:'red',fontName:'Courier',fontSize:16)")
```

- Parameters that can have more than one value per sub-option are wrapped in "[ ]". For example to set the labels for left and right axes use:

```r
options=list(vAxes="[title:'val1'],(title:'val2')]")
```

- `gvis.editor` a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.
Warnings

AnnotatedTimeline (gvisAnnotatedTimeLine) is no longer supported by Google. The annotated timeline now automatically uses Annotation Charts instead.

Important: To use this visualization, you must specify the height and width of the container element explicitly on your page. So, for example: options=list(width="600px", height="350px")

Use code gvisAnnotationChart for a non-Flash version of this plot.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

References

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/annotatedtimeline

See Also

See also print.gvis, plot.gvis for printing and plotting methods. Further see reshape for re-shaping data, e.g. from a wide format into a long format.

Examples

## Please note that by default the googleVis plot command
## will open a browser window and requires Flash and Internet
## connection to display the visualisation.

data(Stock)
Stock
A1 <- gvisAnnotatedTimeLine(Stock, datevar="Date",
  numvar="Value", idvar="Device",
  titlevar="Title", annotationvar="Annotation",
  options=list(displayAnnotations=TRUE,
    legendPosition='newRow',
    width="600px", height="350px")
)
plot(A1)

## Two Y-axis
A2 <- gvisAnnotatedTimeLine(Stock, datevar="Date",
  numvar="Value", idvar="Device",
  titlevar="Title", annotationvar="Annotation",
  options=list(displayAnnotations=TRUE,
    width="600px", height="350px", scaleColumns="[0,1]",
    scaleType='allmaximized')
)
plot(A2)

## Colouring the area below the lines to create an area chart
A3 <- gvisAnnotatedTimeLine(Stock, datevar="Date",
gvisAnnotationChart

numvar="Value", idvar="Device",
titlevar="Title", annotationvar="Annotation",
options=list(
    width="600px", height="350px",
    fill=10, displayExactValues=TRUE,
    colors=['#0000ff', '#00ff00'])
)

plot(A3)

## Data with POSIXct datetime variable
A4 <- gvisAnnotatedTimeLine(Andrew, datevar="Date/Time UTC",
numvar="Pressure_mb",
options=list(scaleType='maximized',
    width="600px", height="350px")
)

plot(A4)

---

**gvisAnnotationChart**  
*Google Annotation Chart with R*

**Description**

gvisAnnotationChart charts are interactive time series line charts that support annotations. Unlike the gvisAnnotatedTimeLine, which uses Flash, annotation charts are SVG/VML and should be preferred whenever possible.

**Usage**

gvisAnnotationChart(
    data,
    datevar = "",
    numvar = "",
    idvar = "",
    titlevar = "",
    annotationvar = "",
    date.format = "%Y/%m/%d",
    options = list(),
    chartid
)

**Arguments**

data a data.frame. The data has to have at least two columns, one with date information (datevar) and one numerical variable.
datevar column name of data which shows the date dimension. The information has to be of class Date or POSIX* time series.
numvar  column name of data which shows the values to be displayed against datevar. The information has to be numeric.

idvar  column name of data which identifies different groups of the data. The information has to be of class character or factor.

titlevar  column name of data which shows the title of the annotations. The information has to be of class character or factor. Missing information can be set to NA. See section 'Details' for more details.

annotationvar  column name of data which shows the annotation text. The information has to be of class character or factor. Missing information can be set to NA. See section 'Details' for more details.

date.format  if datevar is of class Date then this argument specifies how the dates are reformatted to be used by JavaScript.

options  list of configuration options, see: https://developers.google.com/chart/interactive/docs/gallery/annotationchart#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- Boolean arguments are set to either TRUE or FALSE, using the R syntax.
- Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is options=list(width=200,height=300).
  Exceptions to this rule are the width and height options for gvisAnnotatedTimeLine and gvisAnnotationChart. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options=list(width="200px",height="300px").
- Google API parameters with names that don’t include a ",". but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g.
  options=list(colors="['#cb69d9','#603913','#c69c6e']")
- Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as ‘true’ or ‘false’. For example the Google documentation states the formatting options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as:
  options=list(vAxis="{format:'#,###%'}")
- If several sub-options have to be set, e.g.
  titleTextStyle.color,titleTextStyle.fontName and titleTextStyle.fontSize, then those can be combined in one list item such as:
  options=list(titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}")
- parameters that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use:
  options=list(vAxes="[{'title:'val1'},{'title:'val2'}]"
- gvis.editor a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

chartid  character. If missing (default) a random chart id will be generated based on chart type and tempfile.
gvisAnnotationChart returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

- **type** Google visualisation type
- **chartid** character id of the chart object. Unique chart ids are required to place several charts on the same page.
- **html** a list with the building blocks for a page
  - **header** a character string of a html page header: <html>...</body>,
  - **chart** a named character vector of the chart's building blocks:
    - **jsHeader** Opening <script> tag and reference to Google's JavaScript library.
    - **jsData** JavaScript function defining the input data as a JSON object.
    - **jsDrawChart** JavaScript function combing the data with the visualisation API and user options.
    - **jsDisplayChart** JavaScript function calling the handler to display the chart.
    - **jsFooter** End tag </script>.
    - **jsChart** Call of the jsDisplayChart function.
  - **divChart** <div> container to embed the chart into the page.
  - **caption** character string of a standard caption, including data name and chart id.
  - **footer** character string of a html page footer: </body>...</html>, including the used R and googleVis version and link to Google's Terms of Use.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>, Diego de Castillo <decastillo@gmail.com>

References

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/annotationchart

See Also

See also print.gvis, plot.gvis for printing and plotting methods. Further see reshape for re-shaping data, e.g. from a wide format into a long format.

Examples

```r
## Please note that by default the googleVis plot command
## will open a browser window and requires Internet
## connection to display the visualisation.

data(Stock)
Stock
A1 <- gvisAnnotationChart(Stock, datevar="Date",
    numvar="Value", idvar="Device",
    titlevar="Title", annotationvar="Annotation",
    options=list(displayAnnotations=TRUE,
           legendPosition="newRow",
           width=600, height=350)
)
```
## Two Y-axis

```r
A2 <- gvisAnnotationChart(Stock, datevar="Date",
numvar="Value", idvar="Device",
titlevar="Title", annotationvar="Annotation",
options=list(displayAnnotations=TRUE,
width=600, height=350, scaleColumns='[0,1]',
scaleType='allmaximized')
)
```

plot(A2)

## Colouring the area below the lines to create an area chart

```r
A3 <- gvisAnnotationChart(Stock, datevar="Date",
numvar="Value", idvar="Device",
titlevar="Title", annotationvar="Annotation",
options=list(
width=600, height=350,
fill=10, displayExactValues=TRUE,
colors="["#0000ff","#00ff00"]")
)
```

plot(A3)

## Data with POSIXct datetime variable

```r
A4 <- gvisAnnotationChart(Andrew, datevar="Date/Time UTC",
numvar="Pressure_mb",
options=list(scaleType='maximized')
)
```

plot(A4)

# Change background to blue

```r
A5 <- gvisAnnotationChart(Stock, datevar="Date",
numvar="Value", idvar="Device",
titlevar="Title", annotationvar="Annotation",
options=list(
displayAnnotations=TRUE,
chart = "{chartArea:{backgroundColor:'#003b70'}},
legendPosition='newRow',
width=600, height=350})
```

plot(A5)

---

gvisAreaChart

### Description

The gvisAreaChart function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page.
Usage

gvisAreaChart(data, xvar = "", yvar = "", options = list(), chartid)

Arguments

data a data.frame to be displayed as an area chart
xvar name of the character column which contains the category labels for the x-axes.
yvar a vector of column names of the numerical variables to be plotted. Each column is displayed as a separate line.
options list of configuration options, see:
https://developers.google.com/chart/interactive/docs/gallery/areachart#Configuration_Options
The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- Boolean arguments are set to either TRUE or FALSE, using the R syntax.
- Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is options=list(width=200,height=300). Exceptions to this rule are the width and height options for gvisAnnotatedTimeLine and gvisAnnotationChart. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options=list(width="200px",height="300px").
- Google API parameters with names that don’t include a ",", but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g.
  options=list(colors="["#cbb69d",'#603913','#c69c6e']")
- Google API parameters with names that do include a "," present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as ‘true’ or ‘false’. For example the Google documentation states the formating options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as:
  options=list(vAxis="{"format: '#,###%'}").
- If several sub-options have to be set, e.g.
  titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize, then those can be combined in one list item such as:
  options=list(titleTextStyle="{'color: 'red',fontName: 'Courier',fontSize:16'}")
- parameters that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use:
  options=list(vAxes="[{title: 'val1'}, {title: 'val2'}]"
- gvis.editor a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

chartid character. If missing (default) a random chart id will be generated based on chart type and tempfile

Details

The area chart is rendered within the browser using SVG or VML and displays tips when hovering over points.
Value

gvisAreaChart returns list of **class** "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

- **type** Google visualisation type
- **chartId** character id of the chart object. Unique chart ids are required to place several charts on the same page.
- **html** a list with the building blocks for a page
  - **header** a character string of a html page header: `<html>...<body>`,
  - **chart** a named character vector of the chart’s building blocks:
    - **jsHeader** Opening `<script>` tag and reference to Google’s JavaScript library.
    - **jsData** JavaScript function defining the input data as a JSON object.
    - **jsDrawChart** JavaScript function combing the data with the visualisation API and user options.
    - **jsDisplayChart** JavaScript function calling the handler to display the chart.
    - **jsFooter** End tag `</script>.
    - **jsChart** Call of the `jsDisplayChart` function.
  - **divChart** `<div>` container to embed the chart into the page.
- **caption** character string of a standard caption, including data name and chart id.
- **footer** character string of a html page footer: `</body>...</html>`

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

References

Google Chart Tools API: [https://developers.google.com/chart/interactive/docs/gallery/areachart](https://developers.google.com/chart/interactive/docs/gallery/areachart)

See Also

See also `print.gvis`, `plot.gvis` for printing and plotting methods

Examples

```r
## Please note that by default the googleVis plot command
## will open a browser window and requires an internet
## connection to display the visualisation.

df = data.frame(country=c("US", "GB", "BR"), val1=c(1,3,4), val2=c(23,12,32))

## Area chart
Area1 <- gvisAreaChart(df, xvar="country", yvar=c("val1", "val2"))
plot(Area1)

## Stacked chart
Area2 <- gvisAreaChart(df, xvar="country", yvar=c("val1", "val2"),
                        options=list(isStacked=TRUE))
```
## gvisBarChart

The `gvisBarChart` function reads a data.frame and creates text output referring to the Google Visualization API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser using SVG or VML.

### Description

The `gvisBarChart` function reads a data.frame and creates text output referring to the Google Visualization API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser using SVG or VML.

### Usage

```r
gvisBarChart(data, xvar = "", yvar = "", options = list(), chartid)
```

### Arguments

- **data**: a `data.frame` to be displayed as a bar chart
- **xvar**: name of the character column which contains the category labels for the x-axes.
- **yvar**: a vector of column names of the numerical variables to be plotted. Each column is displayed as a separate bar/column.
- **options**: list of configuration options, see:
  - `https://developers.google.com/chart/interactive/docs/gallery/barchart#Configuration_Options`
  - The parameters can be set via a named list. The parameters have to map those of the Google documentation.
    - Boolean arguments are set to either `TRUE` or `FALSE`, using the R syntax.
    - Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is `options=list(width=200,height=300)`. Exceptions to this rule are the width and height options for `gvisAnnotatedTimeLine` and `gvisAnnotationChart`. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, `options=list(width="200px",height="300px")`.

---

```r
plot(Area2)

### Not run:

## Add a customised title
Area3 <- gvisAreaChart(df, xvar="country", yvar=c("val1", "val2"),
                       options=list(title="Hello World",
                                     titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}'"))
plot(Area3)

## Change y-axis to percentages
Area3 <- gvisAreaChart(df, xvar="country", yvar=c("val1", "val2"),
                       options=list(vAxis="{format:'#,###%'}"))
plot(Area3)

## End(Not run)
```
gvisBarChart

- Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g.
  
  options=list(colors=['#cbb69d','#603913','#c69c6e'])

- Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as ‘true’ or ‘false’. For example the Google documentation states the formatting options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as:
  
  options=list(vAxis="{"format: '#,###%'}")

- If several sub-options have to be set, e.g.
  titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize, then those can be combined in one list item such as:
  options=list(titleTextStyle="{"color: 'red',fontName: 'Courier',fontSize:16'}")

- parameters that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use:
  
  options=list(vAxes="["{title: 'val1'},{"title: 'val2']}"]")

- gvis.editor a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

chartid character. If missing (default) a random chart id will be generated based on chart type and tempfile

Value

gvisBarChart returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

type Google visualisation type
chartid character id of the chart object. Unique chart ids are required to place several charts on the same page.
html a list with the building blocks for a page
header a character string of a html page header: <html>...<body>,
chart a named character vector of the chart’s building blocks:
  jsHeader Opening <script> tag and reference to Google’s JavaScript library.
  jsData JavaScript function defining the input data as a JSON object.
  jsDrawChart JavaScript function combing the data with the visualisation API and user options.
  jsDisplayChart JavaScript function calling the handler to display the chart.
  jsFooter End tag </script>.
  jsChart Call of the jsDisplayChart function.
  divChart <div> container to embed the chart into the page.
caption character string of a standard caption, including data name and chart id.
footer character string of a html page footer: </body>...</html>, including the used R and googleVis version and link to Google’s Terms of Use.
Author(s)
Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

References
Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/barchart

See Also
See also print.gvis, plot.gvis for printing and plotting methods

Examples
## Please note that by default the googleVis plot command
## will open a browser window and requires an internet
## connection to display the visualisation.

df <- data.frame(country=c("US", "GB", "BR"),
                 val1=c(1,3,4),
                 val2=c(23,12,32))

## Bar chart
Bar1 <- gvisBarChart(df, xvar="country", yvar=c("val1", "val2"))
plot(Bar1)

## Stacked bar chart
Bar2 <- gvisBarChart(df, xvar="country", yvar=c("val1", "val2"),
                      options=list(isStacked=TRUE))
plot(Bar2)

## Add a customised title and change width of bars
Bar3 <- gvisBarChart(df, xvar="country", yvar=c("val1", "val2"),
                      options=list(title="Hello World",
                                    titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}'",
                                    bar="{groupWidth:'100%'}"))
plot(Bar3)

## Not run:
## Change x-axis to percentages
Bar4 <- gvisBarChart(df, xvar="country", yvar=c("val1", "val2"),
                      options=list(hAxis="{format:'#,###%'}"))
plot(Bar4)

## The following example reads data from a Wikipedia table and displays
## the information in a bar chart.
## We use the readHTMLTable function of the XML package to get the data
library(XML)

## Get the data of the biggest ISO container companies from Wikipedia
##(table 3):
df=readHTMLTable(readLines("https://en.wikipedia.org/wiki/Intermodal_freight_transport"))[3][,1:2]

## Rename the second column
names(df)[2]="TEU capacity"

## The numbers are displayed with commas to separate thousands, so let's
## get rid of them:

```r
df[,2]=as.numeric(gsub("","", as.character(df[,2])))
```

## Finally we can create a nice bar chart:

```r
Bar5 <- gvisBarChart(df, options=list(
    chartArea="{left:250,top:50,width:"50\%",height:"75\%"}",
    legend="bottom",
    title="Top 20 container shipping companies in order of TEU capacity")

plot(Bar5)
```

## End(Not run)

---

### gvisBubbleChart

**Google Bubble Chart with R**

**Description**

The `gvisBubbleChart` function reads a `data.frame` and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page.

**Usage**

```r
gvisBubbleChart(
    data, 
    idvar = "", 
    xvar = "", 
    yvar = "", 
    colorvar = "", 
    sizevar = "", 
    options = list(), 
    chartid
)
```

**Arguments**

- `data` a `data.frame` to be displayed as a bubble chart. The data has to have at least three columns for `idvar`, `xvar`, and `yvar`.
- `idvar` column name of `data` with the bubble
- `xvar` column name of a numerical vector in `data` to be plotted on the x-axis.
- `yvar` column name of a numerical vector in `data` to be plotted on the y-axis.
- `colorvar` column name of data that identifies bubbles in the same series. Use the same value to identify all bubbles that belong to the same series; bubbles in the same series will be assigned the same color. Series can be configured using the `series` option.
- `sizevar` values in this column are mapped to actual pixel values using the `sizeAxis` option.
The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- **Boolean arguments** are set to either `TRUE` or `FALSE`, using the R syntax.
- **Google API parameters** with a single value and with names that don't include a `.` are set like one would do in R, that is `options=list(width=200,height=300)`.

Exceptions to this rule are the width and height options for `gvisAnnotatedTimeLine` and `gvisAnnotationChart`. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, `options=list(width="200px",height="300px")`.

- **Google API parameters** with names that don't include a `.` and require multivalues are set as a character, wrapped in "[]" and separated by commas, e.g. `options=list(colors="['#cbb69d','#603913','#c69c6e']")`.

- **Google API parameters** with names that do include a `.` present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter: value. Boolean values have to be stated as 'true' or 'false'. For example the Google documentation states the formatting options for the vertical axis and states the parameter as `vAxis.format`. Then this parameter can be set in R as: `options=list(vAxis="{format:'#,###%'}")`.

- If several sub-options have to be set, e.g. `titleTextStyle.color`, `titleTextStyle.fontName` and `titleTextStyle.fontSize`, then those can be combined in one list item such as: `options=list(titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}')")`.

- Parameters that can have more than one value per sub-options are wrapped in "[]". For example to set the labels for left and right axes use: `options=list(vAxes="[[title:'val1'],[title:'val2']]")`.

- `gvis.editor` a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

**chartid** character. If missing (default) a random chart id will be generated based on chart type and `tempfile`.

### Details

A bubble chart is used to visualize a data set with 2 to 4 dimensions. The first two dimensions are visualized as coordinates, the 3rd as color and the 4th as size.

The bubble chart is rendered within the browser using SVG or VML and displays tips when hovering over points.

### Value

gvisBubbleChart returns list of `class"gvis"` and "list". An object of class "gvis" is a list containing at least the following components:

- **type** Google visualisation type
chartid character id of the chart object. Unique chart ids are required to place several charts on the same page.

html a list with the building blocks for a page

header a character string of a html page header: `<html>...<body>`,

chart a named character vector of the chart's building blocks:

jsHeader Opening `<script>` tag and reference to Google's JavaScript library.

jsData JavaScript function defining the input data as a JSON object.

jsDrawChart JavaScript function combing the data with the visualisation API and user options.

jsDisplayChart JavaScript function calling the handler to display the chart.

jsFooter End tag `</script>`.

jsChart Call of the `jsDisplayChart` function.

divChart `<div>` container to embed the chart into the page.

caption character string of a standard caption, including data name and chart id.

footer character string of a html page footer: `</body>...</html>`, including the used R and googleVis version and link to Google's Terms of Use.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>,

Diego de Castillo <decastillo@gmail.com>

References

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/bubblechart

See Also

See also `gvisMotionChart` for a moving bubble chart over time, and `print.gvis`, `plot.gvis` for printing and plotting methods.

Examples

```r
bubble1 <- gvisBubbleChart(Fruits, idvar="Fruit", xvar="Sales", yvar="Expenses")
plot(bubble1)

## Set color and size
bubble2 <- gvisBubbleChart(Fruits, idvar="Fruit", xvar="Sales", yvar="Expenses",
                          colorvar="Location", sizevar="Profit",
                          options=list(hAxis=quote(minValue:75, maxValue:125)))
plot(bubble2)

## Use year to color the bubbles
bubble3 <- gvisBubbleChart(Fruits, idvar="Fruit", xvar="Sales", yvar="Expenses",
                          colorvar="Year", sizevar="Profit",
                          options=list(hAxis=quote(minValue:75, maxValue:125)))
plot(bubble3)

## Gradient colour example
def
```
bubble4 <- gvisBubbleChart(Fruits, idvar="Fruit", xvar="Sales", yvar="Expenses", sizevar="Profit", options=list(hAxis='(minValue:75, maxValue:125)', colorAxis='colors: ["lightblue", "blue"]'))

plot(bubble4)

## Not run:
## Moving bubble chart over time, aka motion chart

M <- gvisMotionChart(Fruits, Fruit, Year)
plot(M)

## End(Not run)

gvisCalendar

Google Calendar Chart with R

Description

A calendar chart is a visualization used to show activity over the course of a long span of time, such as months or years. They’re best used when you want to illustrate how some quantity varies depending on the day of the week, or how it trends over time.

Usage

gvisCalendar(data, datevar = "", numvar = "", options = list(), chartid)

Arguments

data a data.frame. The data has to have at least two columns, one with date information (datevar) and one numerical variable.
datevar column name of data which shows the date dimension. The information has to be of class Date or POSIX* time series.
numvar column name of data which shows the values to be displayed against datevar. The information has to be numeric.
options list of configuration options, see:
 https://developers.google.com/chart/interactive/docs/gallery/calendar#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

• Boolean arguments are set to either TRUE or FALSE, using the R syntax.
• Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is options=list(width=200,height=300). Exceptions to this rule are the width and height options for gvisAnnotatedTimeLine and gvisAnnotationChart. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options=list(width="200px",height="300px").
**gvisCalendar**

- Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g. `options=list(colors=["#cbb69d","#683913","#69c6e"])

- Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter: value. Boolean values have to be stated as 'true' or 'false'. For example the Google documentation states the format options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as: `options=list(vAxis='{format: "#,###\%"})`

- If several sub-options have to be set, e.g. titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize, then those can be combined in one list item such as: `options=list(titleTextStlye='{color: 'red',fontName: 'Courier',fontSize:16}')`

- Parameters that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use: `options=list(vAxes=['{title: 'val1'}, {title: 'val2'}'])`

- `gvis.editor` a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

**Value**

`gvisCalendar` returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

- **type** Google visualisation type
- **chartid** character id of the chart object. Unique chart ids are required to place several charts on the same page.
- **html** a list with the building blocks for a page
  - **header** a character string of a html page header: `<html>...<body>`,
  - **chart** a named character vector of the chart's building blocks:
    - **jsHeader** Opening `<script>` tag and reference to Google's JavaScript library.
    - **jsData** JavaScript function defining the input data as a JSON object.
    - **jsDrawChart** JavaScript function combing the data with the visualisation API and user options.
    - **jsDisplayChart** JavaScript function calling the handler to display the chart.
    - **jsFooter** End tag `</script>`,
    - **jsChart** Call of the `jsDisplayChart` function.
    - **divChart** `<div>` container to embed the chart into the page.
  - **caption** character string of a standard caption, including data name and chart id.
  - **footer** character string of a html page footer: `</body>...</html>`, including the used R and googleVis version and link to Google's Terms of Use.

**Warning**

The calendar chart may be undergoing substantial revisions in future Google Charts releases.
Author(s)

Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

References

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/calendar

See Also

See also print.gvis, plot.gvis for printing and plotting methods.

Examples

c11 <- gvisCalendar(Cairo, datevar="Date", numvar="Temp")
plot(c11)

## Not all months shown?
## We can change the setting of the width ...

c12 <- gvisCalendar(Cairo, datevar="Date", numvar="Temp",
                   options=list(width=1000))
plot(c12)

## ... or the cell size

c13 <- gvisCalendar(Cairo, datevar="Date",
                   numvar="Temp",
                   options=list(calendar="{ cellSize: 10 }"))
plot(c13)

## Example with many options set

c14 <- gvisCalendar(Cairo, datevar="Date", numvar="Temp",
                   options=list(
                      title="Daily temperature in Cairo",
                      height=320,
                      calendar="{yearLabel: {fontName: 'Times-Roman',
                        fontSize: 32, color: '#1A8763', bold: true},
                        cellSize: 10,
                        cellColor: {stroke: 'red', strokeOpacity: 0.2 },
                        focusedCellColor: {stroke: 'red'})
                     )
                   )

plot(c14)
Usage

gvisCandlestickChart(  
data,  
xvar = "",  
low = "",  
open = "",  
close = "",  
high = "",  
options = list(),  
chartid
)

Arguments

data a `data.frame` to be displayed as a candlestick chart. The data has to have at least 5 columns.
xvar name of the character column which contains the category labels for the x-axes.
low name of the numeric column specifying the low/minimum value of this marker. This is the base of the candle’s center line.
open name of the numeric column specifying the opening/initial value of this marker. This is one vertical border of the candle. If less than the close value, the candle will be filled; otherwise it will be hollow.
close name of the numeric column specifying the closing/final value of this marker. This is the second vertical border of the candle. If less than the open value, the candle will be hollow; otherwise it will be filled.
high name of the numeric column specifying the high/maximum value of this marker. This is the top of the candle’s center line.
options list of configuration options, see: https://developers.google.com/chart/interactive/docs/gallery/candlestickchart#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- Boolean arguments are set to either `TRUE` or `FALSE`, using the R syntax.
- Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is `options=list(width=200,height=300)`. Exceptions to this rule are the width and height options for `gvisAnnotatedTimeline` and `gvisAnnotationChart`. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, `options=list(width="200px",height="300px")`.
- Google API parameters with names that don’t include a "." but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g. `options=list(colors="["#cbb69d","#603913","#c69c6e"]")`
- Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as ‘true’ or ‘false’. For example the Google documentation states the formatting options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as: `options=list(vAxis="{"format:"#,###%"})"`).
• If several sub-options have to be set, e.g. `titleTextStyle.color`, `titleTextStyle.fontName` and `titleTextStyle.fontSize`, then those can be combined in one list item such as: `options=list(titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}")`

• Parameters that can have more than one value per sub-options are wrapped in `"[ ]"`. For example to set the labels for left and right axes use: `options=list(vAxes="[{title:'val1'},{{title:'val2'}}]")`

• `gvis.editor` a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

chartid character. If missing (default) a random chart id will be generated based on chart type and `tempfile`

details

The `gvisCandlestickChart` function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser using SVG or VML.

A candlestick chart is used to show an opening and closing value overlaid on top of a total variance. Candlestick charts are often used to show stock value behavior. In this chart, items where the opening value is less than the closing value (a gain) are drawn as filled boxes, and items where the opening value is more than the closing value (a loss) are drawn as hollow boxes.

value

`gvisCandlestickChart` returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

- **type** Google visualisation type
- **chartid** character id of the chart object. Unique chart ids are required to place several charts on the same page.
- **html** a list with the building blocks for a page
  - **header** a character string of a html page header: `<html>...<body>`,
  - **chart** a named character vector of the chart’s building blocks:
    - `jsHeader` Opening `<script>` tag and reference to Google’s JavaScript library.
    - `jsData` JavaScript function defining the input data as a JSON object.
    - `jsDrawChart` JavaScript function combining the data with the visualisation API and user options.
    - `jsDisplayChart` JavaScript function calling the handler to display the chart.
    - `jsFooter` End tag `</script>`.
    - `jsChart` Call of the `jsDisplayChart` function.
    - `divChart` `<div>` container to embed the chart into the page.
  - **caption** character string of a standard caption, including data name and chart id.
  - **footer** character string of a html page footer: `</body>...</html>`, including the used R and googleVis version and link to Google’s Terms of Use.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>
References

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/candlestickchart

See Also

See also print.gvis, plot.gvis for printing and plotting methods

Examples

## Please note that by default the googleVis plot command
## will open a browser window and requires an internet
## connection to display the visualisation.

## Example data set
OpenClose

C1 <- gvisCandlestickChart(OpenClose, xvar="Weekday", low="Low",
open="Open", close="Close",
high="High",
options=list(legend='none'))

plot(C1)

---

**gvisColumnChart** Google Column Chart with R

Description

The gvisColumnChart function reads a data.frame and creates text output referring to the Google
Visualisation API, which can be included into a web page, or as a stand-alone page. The actual
chart is rendered by the web browser using SVG or VML.

Usage

gvisColumnChart(data, xvar = "", yvar = "", options = list(), chartid)

Arguments

data a data.frame to be displayed as a column chart
xvar name of the character column which contains the category labels for the x-axes.
yvar a vector of column names of the numerical variables to be plotted. Each column
is displayed as a separate bar/column.
options list of configuration options, see:
https://developers.google.com/chart/interactive/docs/gallery/columnchart#Configuration_Options
The parameters can be set via a named list. The parameters have to map those
of the Google documentation.
  • Boolean arguments are set to either TRUE or FALSE, using the R syntax.
• Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is `options=list(width=200,height=300)`. Exceptions to this rule are the width and height options for `gvisAnnotatedTimeLine` and `gvisAnnotationChart`. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, `options=list(width="200px",height="300px")`.

• Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "] [" and separated by commas, e.g. `options=list(colors=['#cbb69d', '#603913', '#c69c6e'])`.

• Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as 'true' or 'false'. For example the Google documentation states the formatting options for the vertical axis and states the parameter as `vAxis.format`. Then this parameter can be set in R as: `options=list(vAxis={format:'#,###%'})`.

• If several sub-options have to be set, e.g. `titleTextStyle.color,titleTextStyle.fontName` and `titleTextStyle.fontSize`, then those can be combined in one list item such as: `options=list(titleTextStyle=('color: red',fontName:'Courier',fontSize:16))`.

• parameters that can have more than one value per sub-options are wrapped in "] [". For example to set the labels for left and right axes use: `options=list(vAxes=[{title:val1},{title:val2}])`.

• `gvis.editor` a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

- **chartid** character. If missing (default) a random chart id will be generated based on chart type and `tempfile`.

### Value

gvisColumnChart returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

- **type** Google visualisation type
- **chartid** character id of the chart object. Unique chart ids are required to place several charts on the same page.
- **html** a list with the building blocks for a page
  - **header** a character string of a html page header: `<html>...<body>`,
  - **chart** a named character vector of the chart’s building blocks:
    - **jsHeader** Opening `<script>` tag and reference to Google’s JavaScript library.
    - **jsData** JavaScript function defining the input data as a JSON object.
    - **jsDrawChart** JavaScript function combing the data with the visualisation API and user options.
    - **jsDisplayChart** JavaScript function calling the handler to display the chart.
    - **jsFooter** End tag `</script>`.
    - **jsChart** Call of the `jsDisplayChart` function.
  - **divChart** `<div>` container to embed the chart into the page.
caption  character string of a standard caption, including data name and chart id.
footer  character string of a html page footer: </body>...</html>, including the used R
and googleVis version and link to Google’s Terms of Use.

Author(s)
Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

References
Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/
columnchart

See Also
See also print.gvis, plot.gvis for printing and plotting methods

Examples

## Please note that by default the googleVis plot command
## will open a browser window and requires an internet
## connection to display the visualisation.

df=data.frame(country=c("US", "GB", "BR"), val1=c(1,3,4), val2=c(23,12,32))

## Column chart
Col1 <- gvisColumnChart(df, xvar="country", yvar=c("val1", "val2"))
plot(Col1)

## Stacked column chart
Col2 <- gvisColumnChart(df, xvar="country", yvar=c("val1", "val2"),
options=list(isStacked=TRUE))
plot(Col2)

## Add a customised title and and change width of columns
Col3 <- gvisColumnChart(df, xvar="country", yvar=c("val1", "val2"),
options=list(title="Hello World",
  titleTextStyle="(color:'red',fontName:'Courier',fontSize:16)",
  bar="(groupWidth:'100%')")
plot(Col3)

## Not run:
## Change y-axis to percentages
Col4 <- gvisColumnChart(df, xvar="country", yvar=c("val1", "val2"),
options=list(vAxis="{format:'#,###%'}")
plot(Col4)

## End(Not run)
gvisComboChart: Google Combo Chart with R

Description

A chart that lets you render each series as a different marker type from the following list: columns, lines, and area lines.

Usage

gvisComboChart(data, xvar = "", yvar = "", options = list(), chartid)

Arguments

data: a data.frame to be displayed as a columns, line and area chart.
xvar: name of the character column which contains the category labels for the x-axes.
yvar: a vector of column names of the numerical variables to be plotted. Each column is displayed as a separate column, line or area series.
options: list of configuration options, see: https://developers.google.com/chart/interactive/docs/gallery/combochart#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- Boolean arguments are set to either TRUE or FALSE, using the R syntax.
- Google API parameters with a single value and with names that don't include a "." are set like one would do in R, that is options=list(width=200,height=300). Exceptions to this rule are the width and height options for gvisAnnotatedTimeLine and gvisAnnotationChart. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options=list(width="200px",height="300px").
- Google API parameters with names that don't include a ".", but require multivalues are set as a character, wrapped in "]" and separated by commas, e.g. options=list(colors="['#cbb69d','603913','c69c6e']")
- Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in "]". The values of those sub-options are set via parameter:value. Boolean values have to be stated as 'true' or 'false'. For example the Google documentation states the formating options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as: options=list(vAxis="{'format:##.##'}")
- If several sub-options have to be set, e.g. titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize, then those can be combined in one list item such as: options=list(titleTextStyle="{'color: red',fontName: 'Courier',fontSize:16'}")
- Parameters that can have more than one value per sub-options are wrapped in "["]. For example to set the labels for left and right axes use: options=list(vAxes="[['val1'],[title: 'val2']]")
• **gvis.editor** a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

**chartid** character. If missing (default) a random chart id will be generated based on chart type and tempfile

### Details

The `gvisComboChart` function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser using SVG or VML.

### Value

`gvisComboChart` returns list of **class** "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

- **type** Google visualisation type
- **chartid** character id of the chart object. Unique chart ids are required to place several charts on the same page.
- **html** a list with the building blocks for a page
  - **header** a character string of a html page header: `<html>...<body>`,
  - **chart** a named character vector of the chart's building blocks:
    - **jsHeader** Opening `<script>` tag and reference to Google's JavaScript library.
    - **jsData** JavaScript function defining the input data as a JSON object.
    - **jsDrawChart** JavaScript function combing the data with the visualisation API and user options.
    - **jsDisplayChart** JavaScript function calling the handler to display the chart.
    - **jsFooter** End tag `</script>`.
    - **jsChart** Call of the **jsDisplayChart** function.
    - **divChart** `<div>` container to embed the chart into the page.
  - **caption** character string of a standard caption, including data name and chart id.
  - **footer** character string of a html page footer: `</body>...</html>`, including the used R and googleVis version and link to Google's Terms of Use.

### Author(s)

Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

### References

Google Chart Tools API: [https://developers.google.com/chart/interactive/docs/gallery/combochart](https://developers.google.com/chart/interactive/docs/gallery/combochart)

### See Also

See also `print.gvis, plot.gvis` for printing and plotting methods
Examples

```r
## Please note that by default the googleVis plot command
## will open a browser window and requires an internet
## connection to display the visualisation.

CityPopularity
## Add the mean
CityPopularity$Mean=mean(CityPopularity$Popularity)

C1 <- gvisComboChart(CityPopularity, xvar="City",
yvar=c("Mean", "Popularity"),
options=list(seriesType="bars",
title="City Popularity",
series=’{0: {type:"line"}}’))

plot(C1)

## Changing the width of column
C2 <- gvisComboChart(CityPopularity, xvar="City",
yvar=c("Mean", "Popularity"),
options=list(seriesType="bars",
bar="{groupBy:'100%'}",
title="City Popularity",
series=’{0: {type:"line"}}’))

plot(C2)
```

gvisGantt

Google Gantt Chart with R

Description

A Gantt chart is a type of chart that illustrates the breakdown of a project into its component tasks.

Usage

```r
gvisGantt(
data, 
taskID = "", 
taskName = "", 
resource = "", 
start = "", 
end = "", 
duration = "", 
percentComplete = "", 
dependencies = "", 
options = list(), 
chartid
)
```
Arguments

- **data**: data.frame that contains the data to be visualised
- **taskID**: a string that refers to the column name in `data` for the task ID to be used
- **taskName**: a string that refers to the column name in `data` for the task name to be used
- **resource**: a string that refers to the column name in `data` for the resource to be used
- **start**: a string that refers to the date column name in `data` for the start dates
- **end**: a string that refers to the date column name in `data` for the end dates
- **duration**: a string that refers to the numeric column name in `data` for the task duration in milliseconds
- **percentComplete**: a string that refers to the numeric column name in `data` for the percent complete to be used
- **dependencies**: a string that refers to the column name in `data` for the dependencies to be used
- **options**: list of configuration options. The options are documented in detail by Google online:
  
  https://developers.google.com/chart/interactive/docs/gallery/ganttchart#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- Boolean arguments are set to either `TRUE` or `FALSE`, using the R syntax.
- Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is `options=list(width=200,height=300)`.
  Exceptions to this rule are the width and height options for `gvisAnnotatedTimeLine` and `gvisAnnotationChart`. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, `options=list(width="200px",height="300px")`.
- Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g.,
  ```r
  options=list(colors="['#cbb69d','#603913','#c69c6e']")
  ```
- Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in ".". The values of those sub-options are set via parameter:value. For example the Google documentaion states the formating options for the vertical axis and states the parameter as `vAxis.format`. Then this parameter can be set in R as:
  ```r
  options=list(vAxis="{format:'#,###%'}")
  ```
- If several sub-options have to be set, e.g.
  ```r
  titleTextStyle.color,titleTextStyle.fontName and titleTextStyle.fontSize,
  ```
  then those can be combined in one list item such as:
  ```r
  options=list(titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}')
  ```
- Parameters that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use:
  ```r
  options=list(vAxes="[{title:'val1'},(title:'val2')]")
  ```
- `gvis.editor` a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.
chartid character. If missing (default) a random chart id will be generated based on chart type and tempfile

Value
gvisGantt returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:
type Google visualisation type
chartid character id of the chart object. Unique chart ids are required to place several charts on the same page.
html a list with the building blocks for a page
  header a character string of a html page header: <html>...</body>,
  chart a named character vector of the chart’s building blocks:
    jsHeader Opening <script> tag and reference to Google's JavaScript library.
    jsData JavaScript function defining the input data as a JSON object.
    jsDrawChart JavaScript function combing the data with the visualisation API and user options.
    jsDisplayChart JavaScript function calling the handler to display the chart.
    jsFooter End tag </script>.
    jsChart Call of the jsDisplayChart function.
    divChart <div> container to embed the chart into the page.
  caption character string of a standard caption, including data name and chart id.
  footer character string of a html page footer: </body>...</html>, including the used R and googleVis version and link to Google’s Terms of Use.

Author(s)
Markus Gesmann <markus.gesmann@gmail.com>

References
Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/ganttchart

Examples
# Helper function to generate example data
daysToMilliseconds <- function(days){
  days * 24 * 60 * 60 * 1000
}
dat <- data.frame(
  taskID = c("Research", "Write", "Cite", "Complete", "Outline"),
  taskName = c("Find sources", "Write Paper", "Create bibliography", "Hand in paper", "Outline paper"),
  resource = c(NA, "write", "write", "complete", "write"),
  start = as.Date(c("2015-01-01", "2015-01-01"), NA, NA, NA, NA),
  end = as.Date(c("2015-01-05", "2015-01-09", "2015-01-07", "2015-01-10", "2015-01-06")),
  duration = c(NA, daysToMilliseconds(c(3, 1, 1, 1))),
  percentComplete = c(100, 25, 20, 0, 100),
  dependencies = c(NA, "Research, Outline", "Research",...
gvisGantt

"Cite, Write", "Research")

gntt1 <- gvisGantt(dat, taskID = "taskID",
                   taskName = "taskName",
                   resource = "resource",
                   start = "start",
                   end = "end",
                   duration = "duration",
                   percentComplete = "percentComplete",
                   dependencies = "dependencies")

plot(gntt1)

## gantt chart with options set

gntt2 <- gvisGantt(dat, taskID = "taskID",
                   taskName = "taskName",
                   resource = "resource",
                   start = "start",
                   end = "end",
                   duration = "duration",
                   percentComplete = "percentComplete",
                   dependencies = "dependencies",
                   options = list(
                     height = 275,
                     gantt = "{[
                       criticalPathEnabled: true,
                       innerGridHorizLine: {
                         stroke: '#ffe0b2',
                         strokeWidth: 2
                       },
                       innerGridTrack: {fill: '#fff3e0'},
                       innerGridDarkTrack: {fill: '#ffcc80'},
                       labelStyle: {
                         fontName: 'Arial',
                         fontSize: 14
                       }
                     ]}"
                   ))

plot(gntt2)

# Example with date time

dat <- data.frame(
                  taskID = c("Research", "Write", "Complete"),
                  taskName = c("Find sources", "Write Paper", "Hand in paper"),
                  resource = c(NA, "write", "complete"),
                  start = as.POSIXct("2015-01-01 6:00:00"), NA, NA),
                  end = as.POSIXct(c("2015-01-01 8:00:00", "2015-01-01 13:30:00", "2015-01-01 20:30:00")),
                  duration = c(NA, daysToMilliseconds(c(.1, .05)) ),
                  percentComplete = c(100, 25, 0),
                  dependencies = c(NA, "Research", "Write")
)

gntt3 <- gvisGantt(dat, taskID = "taskID",
                   taskName = "taskName",
                   resource = "resource",
                   start = "start",
                   end = "end")
The gvisGauge function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser using SVG or VML.

Usage

gvisGauge(data, labelvar = "", numvar = "", options = list(), chartid)

Arguments

data a data.frame to be displayed as a gauge
labelvar name of the character column which contains the category labels for the slice labels.
numvar a vector of column names of the numerical variables of the slice values.
options list of configuration options, see: https://developers.google.com/chart/interactive/docs/gallery/gauge#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- Boolean arguments are set to either TRUE or FALSE, using the R syntax.
- Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is options=list(width=200,height=300). Exceptions to this rule are the width and height options for gvisAnnotatedTimeLine and gvisAnnotationChart. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options=list(width="200px",height="300px").
- Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "]" and separated by commas, e.g. options=list(colors=["#cbb69d","#603913","#c69c6e"])
- Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as ‘true’ or ‘false’. For example the Google documentation states the formatting options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as: options=list(vAxis="(format:'#,###%')").
If several sub-options have to be set, e.g. titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize, then those can be combined in one list item such as:

```r
options=list(titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}")
```

- Parameters that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use:

```r
options=list(vAxes="[{title:'val1'},{title:'val2'}]")
```

- `gvis.editor` a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

**Value**

gvisGauge returns list of `class "gvis"` and "list". An object of class "gvis" is a list containing at least the following components:

- `type` Google visualisation type
- `chartid` character id of the chart object. Unique chart ids are required to place several charts on the same page.
- `html` a list with the building blocks for a page
  - `header` a character string of a HTML page header: `<html>...<body>`,
  - `chart` a named character vector of the chart’s building blocks:
    - `jsHeader` Opening `<script>` tag and reference to Google’s JavaScript library.
    - `jsData` JavaScript function defining the input data as a JSON object.
    - `jsDrawChart` JavaScript function combing the data with the visualisation API and user options.
    - `jsDisplayChart` JavaScript function calling the handler to display the chart.
    - `jsFooter` End tag `</script>`.
    - `jsChart` Call of the `jsDisplayChart` function.
  - `divChart` `<div>` container to embed the chart into the page.
  - `caption` character string of a standard caption, including data name and chart id.
  - `footer` character string of a HTML page footer: `</body>...</html>`, including the used R and googleVis version and link to Google’s Terms of Use.

**Author(s)**

Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

**References**

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/gauge

**See Also**

See also `print.gvis`, `plot.gvis` for printing and plotting methods
Examples

## Please note that by default the googleVis plot command
## will open a browser window and requires an internet
## connection to display the visualisation.

Gauge1 <- gvisGauge(CityPopularity, options=list(min=0, max=800, greenFrom=500,
greenTo=800, yellowFrom=300, yellowTo=500,
redFrom=0, redTo=300))

plot(Gauge1)

gvisGeoChart

Google Geo Chart with R

Description

The gvisGeoChart function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page.

Usage

gvisGeoChart(
  data,
  locationvar = "",
  colorvar = "",
  sizevar = "",
  hovervar = "",
  options = list(),
  chartid
)

Arguments

data a data.frame. The data has to have at least one column with location name
(locationvar), value to be mapped to location. The format of the data varies
depending on which display mode that you use: Regions or Markers.

locationvar column name of data with the geo locations to be analysed. The locations can
be provide in two formats:

Format 1 'latitude:longitude'. See the example below.

Format 2 Address, country name, region name locations, or US metropolitan
area codes, see https://developers.google.com/adwords/api/docs/
appendix/geotargeting?csw=11. This format works with the dataMode
option set to either 'markers' or 'regions'. The following formats are ac-
cepted: A specific address (for example, "1600 Pennsylvania Ave"). A
country name as a string (for example, "England"), or an uppercase ISO-
3166 code or its English text equivalent (for example, "GB" or "United
Kingdom"). An uppercase ISO-3166-2 region code name or its English
text equivalent (for example, "US-NJ" or "New Jersey").
colorvar  column name of data with the optional numeric column used to assign a color to this marker, based on the scale specified in the colorAxis.colors property. If this column is not present, all markers will be the same color. If the column is present, null values are not allowed. Values are scaled relative to each other, or absolutely to values specified in the colorAxis.values property.

sizevar  only used for displayMode='markers'. Column name of data with the optional numeric column used to assign the marker size, relative to the other marker sizes. If this column is not present, the value from the previous column will be used (or default 'size', if no color column is provided as well). If the column is present, null values are not allowed.

hovervar  column name of data with the additional string text displayed when the user hovers over this region.

options  list of configuration options, see:
https://developers.google.com/chart/interactive/docs/gallery/geochart#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

• Boolean arguments are set to either TRUE or FALSE, using the R syntax.
• Google API parameters with a single value and with names that don’t include a “.” are set like one would do in R, that is `options=list(width=200,height=300)`. Exceptions to this rule are the width and height options for `gvisAnnotatedTimeLine` and `gvisAnnotationChart`. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, `options=list(width="200px",height="300px")`.
• Google API parameters with names that don’t include a “.”, but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g.
`options=list(colors=['#cbb69d', '#603913', '#c69c6e'])`
• Google API parameters with names that do include a “.” present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as 'true' or 'false'. For example the Google documentation states the formatting options for the vertical axis and states the parameter as `vAxis.format`. Then this parameter can be set in R as:
`options=list(vAxis='{format: #,###%}')`.
• If several sub-options have to be set, e.g.
titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize, then those can be combined in one list item such as:
`options=list(titleTextStyle={(color: 'red',fontName: 'Courier',fontSize:16)})`
• Parameters that can have more than one value per sub-options are wrapped in "[]". For example to set the labels for left and right axes use:
`options=list(vAxes=['[{title: 'val1'}, {title: 'val2'}]'])`
• `gvis.editor` a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

chartid  character. If missing (default) a random chart id will be generated based on chart type and `tempfile`
Details

A geo chart is a map of a country, a continent, or a region with two modes: The region mode colorizes whole regions, such as countries, provinces, or states. The marker mode marks designated regions using bubbles that are scaled according to a value that you specify.

A geo chart is rendered within the browser using SVG or VML. Note that the map is not scrollable or draggable.

Value

gvisGeoChart returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

- **type**  Google visualisation type
- **chartid**  character id of the chart object. Unique chart ids are required to place several charts on the same page.
- **html**  a list with the building blocks for a page

  - **header**  a character string of a html page header: `<html>...<body>`,
  - **chart**  a named character vector of the chart’s building blocks:
    - **jsHeader**  Opening `<script>` tag and reference to Google’s JavaScript library.
    - **jsData**  JavaScript function defining the input data as a JSON object.
    - **jsDrawChart**  JavaScript function combing the data with the visualisation API and user options.
    - **jsDisplayChart**  JavaScript function calling the handler to display the chart.
    - **jsFooter**  End tag `</script>`.
    - **jsChart**  Call of the jsDisplayChart function.
    - **divChart**  `<div>` container to embed the chart into the page.
- **caption**  character string of a standard caption, including data name and chart id.
- **footer**  character string of a html page footer: `</body>...</html>`, including the used R and googleVis version and link to Google’s Terms of Use.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>, Diego de Castillo <decastillo@gmail.com>

References

Google Chart Tools API: [https://developers.google.com/chart/interactive/docs/gallery/geochart](https://developers.google.com/chart/interactive/docs/gallery/geochart)

See Also

See also `print.gvis`, `plot.gvis` for printing and plotting methods.

Examples

```r
## Please note that by default the googleVis plot command
## will open a browser window and requires Internet
## connection to display the visualisation.

## Regions examples
## The regions style fills entire regions (typically countries) with
```
## colors corresponding to the values that you assign

G1a <- gvisGeoChart(Exports, locationvar='Country', colorvar='Profit')
plot(G1a)

## Change projection
G1b <- gvisGeoChart(Exports, locationvar='Country', colorvar='Profit',
                    options=list(projection="kavrayskiy-vii"))
plot(G1b)

## Plot only Europe
G2 <- gvisGeoChart(Exports, "Country", "Profit",
                   options=list(region="150"))
plot(G2)

## Example showing US data by state
require(datasets)
states <- data.frame(state.name, state.x77)
G3 <- gvisGeoChart(states, "state.name", "Illiteracy",
                   options=list(region="US", displayMode="regions",
                                  resolution="provinces",
                                  width=600, height=400))
plot(G3)

G4 <- gvisGeoChart(Andrew, "LatLong", colorvar='Speed_kt',
                   options=list(region="US"))
plot(G4)

G5 <- gvisGeoChart(Andrew, "LatLong", sizevar='Speed_kt',
                   colorvar="Pressure_mb", options=list(region="US"))
plot(G5)

## Create lat:long values and plot a map of Oceania
## Set background colour to light-blue
require(stats)
data(quakes)
head(quakes)
quakes$latlong<-paste(quakes$lat, quakes$long, sep=":")
G6 <- gvisGeoChart(quakes, "latlong", "depth", "mag",
                   options=list(displayMode="Markers", region="009",
                                  colorAxis="{colors:['red', 'grey']}",
                                  backgroundColor="lightblue"))
plot(G6)
Description

The `gvisGeoMap` function reads a data.frame and creates text output referring to the Google Visualization API, which can be included into a web page, or as a stand-alone page.

A geo map is a map of a country, continent, or region map, with colours and values assigned to specific regions. Values are displayed as a colour scale, and you can specify optional hover-text for regions. The map is rendered in the browser. Note that the map is not scroll-able or drag-gable, but can be configured to allow zooming.

Usage

```r
gvisGeoMap(
data,  
locationvar = "",  
numvar = "",  
hovervar = "",  
options = list(),  
chartid)
```

Arguments

data  data.frame. The data has to have at least two columns with location name (locationvar), value to be mapped to location (numvar) and an optional variable to display any text while the mouse hovers over the location (hovervar).

locationvar  column name of data with the geo locations to be analysed. The locations can be provide in two formats:

**Format 1** 'latitude:longitude'. See the example below.

**Format 2** Address, country name, region name locations, or US metropolitan area codes, see [https://developers.google.com/adwords/api/docs/appendix/geotargeting?csw=1](https://developers.google.com/adwords/api/docs/appendix/geotargeting?csw=1). This format works with the `dataMode` option set to either 'markers' or 'regions'. The following formats are accepted: A specific address (for example, "1600 Pennsylvania Ave"). A country name as a string (for example, "England"), or an uppercase ISO-3166 code or its English text equivalent (for example, "GB" or "United Kingdom"). An uppercase ISO-3166-2 region code name or its English text equivalent (for example, "US-NJ" or "New Jersey").

numvar  column name of data with the numeric value displayed when the user hovers over this region.

hovervar  column name of data with the additional string text displayed when the user hovers over this region.

options  list of configuration options. The options are documented in detail by Google online: [https://developers.google.com/chart/interactive/docs/gallery/geomap#Configuration_Options](https://developers.google.com/chart/interactive/docs/gallery/geomap#Configuration_Options)

The parameters can be set via a named list. The parameters have to map those of the Google documentation.
• Boolean arguments are set to either TRUE or FALSE, using the R syntax.
• Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is options=list(width=200,height=300). Exceptions to this rule are the width and height options for gvisAnnotatedTimeline and gvisAnnotationChart. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options=list(width=200px,height=300px).
• Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g. options=list(colors=['#cbb69d','#603913','#c69c6e'])
• Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in ".". The values of those sub-options are set via parameter:value. Boolean values have to be stated as 'true' or 'false'. For example the Google documentation states the formating options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as: options=list(vAxis='{format:'#,###%'}').
• If several sub-options have to be set, e.g. titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize, then those can be combined in one list item such as: options=list(titleTextStyle='[color:red,fontName:Courier,fontSize:16]')
• Parameters that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use: options=list(vAxes='[title:val1],[title:val2]')
• gvis.editor a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

chartid character. If missing (default) a random chart id will be generated based on chart type and tempfile

Value
gvisGeoMap returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

- **type** Google visualisation type
- **chartid** character id of the chart object. Unique chart ids are required to place several charts on the same page.
- **html** a list with the building blocks for a page
  - **header** a character string of an html page header: <html>...<body>,
  - **chart** a named character vector of the chart’s building blocks:
    - **jsHeader** Opening <script> tag and reference to Google's JavaScript library.
    - **jsData** JavaScript function defining the input data as a JSON object.
    - **jsDrawChart** JavaScript function combing the data with the visualisation API and user options.
    - **jsDisplayChart** JavaScript function calling the handler to display the chart.
    - **jsFooter** End tag </script>.
    - **jsChart** Call of the jsDisplayChart function.
Warnings

GeoMap (gvisGeoMap) is Flash based, consider using GeoChart (gvisGeoChart) instead. For more details visit goo.gl/tkiEV8

Because of Flash security settings the chart might not work correctly when accessed from a file location in the browser (e.g., file:///c:/webhost/myhost/myviz.html) rather than from a web server URL (e.g. https://www.myhost.com/myviz.html). See the googleVis package vignette and the Macromedia web site (https://www.macromedia.com/support/documentation/en/flashplayer/help/) for more details.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>, Diego de Castillo <decastillo@gmail.com>

References

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/geomap

Examples

## Please note that by default the googleVis plot command
## will open a browser window and requires Internet
## connection to display the visualisation.

## Regions Example
## The regions style fills entire regions (typically countries) with colors
## corresponding to the values that you assign. Specify the regions style
## by assigning options['dataMode'] = 'regions' in your code.

G1 <- gvisGeoMap(Exports, locationvar='Country', numvar='Profit',
                  options=list(dataMode="regions"))
plot(G1)

## Markers Example
## The "markers" style displays a circle, sized and colored to indicate
## a value, over the regions that you specify.
G2 <- gvisGeoMap(CityPopularity, locationvar='City', numvar='Popularity',
                  options=list(region='US', height=350,
                                dataMode='markers',
                                colors=[0xFF8747, 0xFFB581, 0xc06000]))
plot(G2)

## Example showing US data by state
require(datasets)
states <- data.frame(state.name, state.x77)
gvisHistogram

The gvisHistogram function reads a data.frame and creates text output referring to the Google Visualization API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser using SVG or VML.

Usage

```r
gvisHistogram(data, options = list(), chartid)
```

Arguments

- **data** 
  a **data.frame** to be displayed as a histogram. Each column will be displayed as a histogram.

- **options** 
  list of configuration options, see
  
  https://developers.google.com/chart/interactive/docs/gallery/histogram#Configuration_Options

  The parameters can be set via a named list. The parameters have to map those of the Google documentation.

  - Boolean arguments are set to either **TRUE** or **FALSE**, using the R syntax.
  - Google API parameters with a single value and with names that don’t include a “.” are set like one would do in R, that is options = list(width=200, height=300). Exceptions to this rule are the width and height options for gvisAnnotatedTimeLine and gvisAnnotationChart. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options = list(width="200px", height="300px").
• Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g.
  options=list(colors=["#cbb69d","#603913","#c69c6e"])

• Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter: value. Boolean values have to be stated as ‘true’ or ‘false’. For example the Google documentation states the formatting options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as:
  options=list(vAxis="(format: '#,###%')").

• If several sub-options have to be set, e.g.
  titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize, then those can be combined in one list item such as:
  options=list(titleTextStyle="{color: 'red',fontName: 'Courier',fontSize:16}")

• parameters that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use:
  options=list(vAxes=[[{title: 'val1'}, {title: 'val2'}]])

• gvis.editor a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

chartid character. If missing (default) a random chart id will be generated based on chart type and tempfile.

Value

gvisHistogram returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

  type Google visualisation type
  chartid character id of the chart object. Unique chart ids are required to place several charts on the same page.
  html a list with the building blocks for a page
    header a character string of a html page header: <html>...<body>,
    chart a named character vector of the chart’s building blocks:
      jsHeader Opening <script> tag and reference to Google’s JavaScript library.
      jsData JavaScript function defining the input data as a JSON object.
      jsDrawChart JavaScript function combing the data with the visualisation API and user options.
      jsDisplayChart JavaScript function calling the handler to display the chart.
      jsFooter End tag </script>.
      jsChart Call of the jsDisplayChart function.
    divChart <div> container to embed the chart into the page.
  caption character string of a standard caption, including data name and chart id.
  footer character string of a html page footer: </body>...</html>, including the used R and googleVis version and link to Google’s Terms of Use.
**Author(s)**
Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

**References**
Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/histogram

**See Also**
See also `print.gvis`, `plot.gvis` for printing and plotting methods

**Examples**

```r
## Please note that by default the googleVis plot command
## will open a browser window and requires an internet
## connection to display the visualisation.

hist1 <- gvisHistogram(dino)
plot(hist1)

## Histogram of the top 20 countries
pop <- Population[1:20,c("Country", "Population")]
pop=transform(pop, Population=round(Population/1e6))

hist2 <- gvisHistogram(pop, option=list(title="Country Populations",
    legend="( position: 'none' )",
    colors=c("green"))

plot(hist2)

set.seed(123)

data.frame(A=rpois(100, 20),
    B=rpois(100, 5),
    C=rpois(100, 50))

hist3 <- gvisHistogram(dat, options=list(
    legend="( position: 'top', maxLines: 2 )",
    colors=c("#5C3292", '#1A8763', '#871B47' ))

plot(hist3)
```

---

**Description**

The `gvisLineChart` function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser using SVG or VML.

**Usage**

```r
gvisLineChart(data, xvar = "", yvar = "", options = list(), chartid)
```
Arguments

- **data**: a data.frame to be displayed as a line chart.
- **xvar**: name of the character column which contains the category labels for the x-axes.
- **yvar**: a vector of column names of the numerical variables to be plotted. Each column is displayed as a separate line.
- **options**: list of configuration options, see https://developers.google.com/chart/interactive/docs/gallery/linechart#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- **Boolean arguments** are set to either `TRUE` or `FALSE`, using the R syntax.
- **Google API parameters** with a single value and with names that don’t include a ‘.’ are set like one would do in R, that is `options=list(width=200,height=300)`. Exceptions to this rule are the width and height options for `gvisAnnotatedTimeline` and `gvisAnnotationChart`. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, `options=list(width="200px",height="300px")`.
- **Google API parameters** with names that don’t include a '.', but require multi-values are set as a character, wrapped in "[ ]" and separated by commas, e.g.
  ```r
  options=list(colors=["#cbb69d","#603913","#c69c6e"])
  ```
- **Google API parameters** with names that do include a '.', present parameters with several sub-options and have to be set as a character wrapped in "[ ]". The values of those sub-options are set via `parameter:value`. Boolean values have to be stated as 'true' or 'false'. For example the Google documentation states the formatting options for the vertical axis and states the parameter as `vAxis.format`. Then this parameter can be set in R as:
  ```r
  options=list(vAxis="{format:,'#,###%'}")
  ```
- **If several sub-options** have to be set, e.g.
  ```r
  options=list(titleTextStyle=color,titleTextStyle.fontName and titleTextStyle.fontSize)
  ```
  then those can be combined in one list item such as:
  ```r
  options=list(titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}')
  ```
- **parameters** that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use:
  ```r
  options=list(vAxes=[{"title:'val1'},{title:'val2'}])
  ```
- **gvis.editor** a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

Value

gvisLineChart returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

- **type**: Google visualisation type
- **chartid**: character id of the chart object. Unique chart ids are required to place several charts on the same page.
html a list with the building blocks for a page
   header a character string of a html page header: <html>...<body>,
   chart a named character vector of the chart’s building blocks:
   jsHeader Opening <script> tag and reference to Google’s JavaScript library.
   jsData JavaScript function defining the input data as a JSON object.
   jsDrawChart JavaScript function combing the data with the visualisation API and user
   options.
   jsDisplayChart JavaScript function calling the handler to display the chart.
   jsFooter End tag </script>.
   jsChart Call of the jsDisplayChart function.
   divChart <div> container to embed the chart into the page.
   caption character string of a standard caption, including data name and chart id.
   footer character string of a html page footer: </body>...</html>, including the used R
   and googleVis version and link to Google’s Terms of Use.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

References

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/
   linechart

See Also

See also print.gvis, plot.gvis for printing and plotting methods

Examples

## Please note that by default the googleVis plot command
## will open a browser window and requires an internet
## connection to display the visualisation.

df <- data.frame(country=c("US", "GB", "BR"), val1=c(1,3,4), val2=c(23,12,32))

## Line chart
Line1 <- gvisLineChart(df, xvar="country", yvar=c("val1", "val2"))
plot(Line1)

## Add a customised title and smoothed curve
Line2 <- gvisLineChart(df, xvar="country", yvar=c("val1", "val2"),
   options=list(title="Hello World",
                 titleTextStyle="{color:’red’,fontName:’Courier’,fontSize:16}",
                 curveType=’function’))
plot(Line2)

## Not run:
## Change y-axis to percentages
Line3 <- gvisLineChart(df, xvar="country", yvar=c("val1", "val2"),
   options=list(yAxis="{format:’#,###%’}")

plot(Line3)

## End(Not run)

## Create a chart with two y-axis:
Line4 <- gvisLineChart(df, "country", c("val1", "val2"),
                      options=list(series="[[targetAxisIndex: 0],
                                    {targetAxisIndex:1}]"},
                      vAxes="[[title:'val1'], {title:'val2'}])
plot(Line4)

## Line chart with edit button
Line5 <- gvisLineChart(df, xvar="country", yvar=c("val1", "val2"),
                        options=list(gvis.editor="Edit me!"))
plot(Line5)

## Customizing lines
Dashed <- gvisLineChart(df, xvar="country", yvar=c("val1", "val2"),
                        options=list(series="[[color:'green', targetAxisIndex: 0,
                                    lineWidth: 1, lineDashStyle: [2, 2, 20, 2, 20, 2]},
                        {color: 'blue', targetAxisIndex: 1,
                                    lineWidth: 2, lineDashStyle: [4, 1]}],
                        vAxes="[[title:'val1'], {title:'val2'}])
plot(Dashed)

gvisMap

Google Maps with R

Description

The gvisMap function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page.

Usage

gvisMap(data, locationvar = ",", tipvar = ",", options = list(), chartid)

Arguments

data a data.frame. The data has to have at least two columns with location name (locationvar) and the variable to display the text in the tip icon (tipvar).

locationvar column name of data with the geo locations to be analysed. The locations can be provide in two formats:

Format 1 'latitude:longitude'. See the example below.

Format 2 The first column should be a string that contains an address. This address should be as complete as you can make it.

tipvar column name of data with the string text displayed over the tip icon.
options list of configuration options for Google Map.

https://developers.google.com/chart/interactive/docs/gallery/map#

Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- Boolean arguments are set to either TRUE or FALSE, using the R syntax.
- Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is options=list(width=200,height=300).
  Exceptions to this rule are the width and height options for gvisAnnotatedTimeLine and gvisAnnotationChart. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options=list(width="200px",height="300px").
- Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g. options=list(colors="["#cbb69d","#603913","#c69c6e"]")
- Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as 'true' or 'false'. For example the Google documentation states the formating options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as: options=list(vAxis="{format: '#,###%'}").
- If several sub-options have to be set, e.g. titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize, then those can be combined in one list item such as: options=list(titleTextStyle="{color: 'red',fontName: 'Courier',fontSize:16}")
- Parameters that can have more than one value per sub-options are wrapped in "] [". For example to set the labels for left and right axes use: options=list(vAxes="[{{title: 'val1'}},{title: 'val2'}]"")
- gvis.editor a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

chartid character. If missing (default) a random chart id will be generated based on chart type and tempfile

Details

The maps are the well known Google Maps.

Value

gvisMap returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

type Google visualisation type
chartid character id of the chart object. Unique chart ids are required to place several charts on the same page.

html a list with the building blocks for a page
header  a character string of a html page header: <html>...<body>,
chart  a named character vector of the chart’s building blocks:
  jsHeader  Opening <script> tag and reference to Google’s JavaScript library.
  jsData  JavaScript function defining the input data as a JSON object.
  jsDrawChart  JavaScript function combing the data with the visualisation API and user
               options.
  jsDisplayChart  JavaScript function calling the handler to display the chart.
  jsFooter  End tag </script>.
  jsChart  Call of the jsDisplayChart function.
  divChart  <div> container to embed the chart into the page.
caption  character string of a standard caption, including data name and chart id.
footer  character string of a html page footer: </body>...</html>, including the used R
         and googleVis version and link to Google’s Terms of Use.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

References

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/
  map

See Also

See also print.gvis, plot.gvis for printing and plotting methods, gvisGeoChart for an alternative to gvisMap.

Examples

## Please note that by default the googleVis plot command
## will open a browser window and requires Internet
## connection to display the visualisation.

## Example with latitude and longitude information
## Plot Hurricane Andrew (1992) storm path:
data(Andrew)
M1 <- gvisMap(Andrew, "LatLong", "Tip",
               options=list(showTip=TRUE, showLine=TRUE, enableScrollWheel=TRUE,
                             mapType='hybrid', useMapTypeControl=TRUE,
                             width=800, height=400))
plot(M1)

## Example with address, here UK post-code and some html code in tooltip
df <- data.frame(Postcode=c("EC3M 7HA", "EC2P 2EJ"),
                 Tip=c("<a href='https://www.lloyds.com'>Lloyd's</a>",
                     "<a href='https://www.guildhall.cityoflondon.gov.uk'>Guildhall</a>"))
M2 <- gvisMap(df, "Postcode", "Tip",
    options=list(showTip=TRUE, mapType='normal',
        enableScrollWheel=TRUE))

plot(M2)

## Change mapping icons
M3 <- gvisMap(df, "Postcode", "Tip",
    options=list(showTip=TRUE, mapType='normal',
        enableScrollWheel=TRUE,
        icons=paste0('{",
            'default': '{normal': 'https://icons.iconarchive.com/",
            "icons/icons-land/vista-map-markers/48/",
               "Map-Marker-Ball-Azure-icon.png",
            'selected': 'https://icons.iconarchive.com/",
            "icons/icons-land/vista-map-markers/48/",
               "Map-Marker-Ball-Right-Azure-icon.png"},
        '}))

plot(M3)

gvisMerge

## Merge two googleVis charts into one gvis-object

description

gvisMerge merges two gvis-objects, either next or below each other into one gvis-object. The objects are arranged in a HTML table.

usage

gvisMerge(x, y, horizontal = FALSE,
    tableOptions = "border="0"", chartid)

arguments

x
    a gvis-object.

y
    a gvis-object.

horizontal
    boolean. Default FALSE. If FALSE the two gvis-objects are arranged below each other, otherwise next to each other.

tableOptions
    a valid HTML table option string. Default "border="0"".

chartid
    character. If missing (default) a random chart id will be generated based on chart type and tempfile

value

gvisMerge returns list of class "gvis" and "list".

An object of class "gvis" is a list containing at least the following components:

type
    Google visualisation type, here 'gvisMerge'
chartid  character id of the chart object. Unique chart ids are required to place several charts on the same page.

html  a list with the building blocks for a page

header  a character string of a html page header: <html>...<body>,

chart  a named character vector of the chart's building blocks:

  jsHeader  Opening <script> tag and reference to Google's JavaScript library.
  jsData  JavaScript function defining the input data as a JSON object.
  jsDrawChart  JavaScript function combing the data with the visualisation API and user options.
  jsDisplayChart  JavaScript function calling the handler to display the chart.
  jsFooter  End tag </script>.
  jsChart  Call of the jsDisplayChart function.
  divChart  <div> container to embed the chart into the page.

caption  character string of a standard caption, including data name and chart id.

footer  character string of a html page footer: </body>...</html>, including the used R and googleVis version and link to Google's Terms of Use.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>,

References

Google Chart Tools API: https://developers.google.com/chart/
Follow the link for Google's data policy.

See Also

See also print.gvis, plot.gvis for printing and plotting methods

Examples

```r
## Please note that by default the googleVis plot command
## will open a browser window and requires Internet
## connection to display the visualisation.

Pie1 <- gvisPieChart(CityPopularity)

## Doughnut chart - a pie with a hole
Pie2 <- gvisPieChart(CityPopularity, options=list(
  slices="{"4: {offset: 0.2}, 0: {offset: 0.3}}",
  title='City popularity',
  legend='none',
  pieSliceText='label',
  pieHole=0.5))

plot(gvisMerge(Pie2, Pie1,
  tableOptions = "cellspacing="20" bgcolor="#AABBCC",
  horizontal=TRUE))
```
## Nested charts

```r
G <- gvisGeoChart(Exports, "Country", "Profit",
               options=list(width=250, height=100))
T <- gvisTable(Exports,
               options=list(width=250, height=300))

GT <- gvisMerge(G, T, horizontal=FALSE)
plot(GT)

M <- gvisMotionChart(Fruits, "Fruit", "Year",
               options=list(width=400, height=410))

GTM <- gvisMerge(GT, M, horizontal=TRUE,
               tableOptions="cellspacing=10")
plot(GTM)

line <- gvisLineChart(OpenClose, "Weekday", c("Open", "Close"),
               options=list(legend="none", width=300, height=150))

column <- gvisColumnChart(OpenClose, "Weekday", c("Open", "Close"),
               options=list(legend="none", width=300, height=150))

area <- gvisAreaChart(OpenClose, "Weekday", c("Open", "Close"),
               options=list(legend="none", width=300, height=150))

bar <- gvisBarChart(OpenClose, "Weekday", c("Open", "Close"),
               options=list(legend="none", width=300, height=150))

LBCA <- gvisMerge(gvisMerge(line, bar), gvisMerge(column, area),
               horizontal=TRUE, tableOptions="bgcolor="#AABBCC""
)
plot(LBCA)

## Applying gvisMerge successively

p <- Reduce(gvisMerge, list(line, column, area, bar))
plot(p)
```

---

gvisMotionChart | Google Motion Chart with R

### Description

The gvisMotionChart function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser in Flash. A motion chart is a dynamic chart to explore several indicators over time.

### Usage

```r
gvisMotionChart(
data,
idvar = "id",
timevar = "time",
xvar = "",
```
yvar = "",  
colorvar = "",  
sizevar = "",  
date.format = "%Y/%m/%d",  
options = list(),  
chartid )

Arguments

data

a data.frame. The data has to have at least four columns with subject name (idvar), time (timevar) and two columns of numeric values. Further columns, numeric and character/factor are optional. The combination of idvar and timevar has to describe a unique row. The column names of the idvar and timevar have to be specified. Further columns, if not specified by the other arguments (xvar, yvar, colorvar, sizevar), will be assumed to be in the order of the arguments.

idvar
column name of data with the subject to be analysed.

timevar
column name of data which shows the time dimension. The information has to be either numeric, of class Date or a character which follows the pattern 'YYYYWww' (e.g. '2010W04' for weekly data) or 'YYYYQq' (e.g. '2010Q1' for quarterly data).

xvar
column name of a numerical vector in data to be plotted on the x-axis.

yvar
column name of a numerical vector in data to be plotted on the y-axis.

colorvar
column name of data that identifies bubbles in the same series. Use the same value to identify all bubbles that belong to the same series; bubbles in the same series will be assigned the same color. Series can be configured using the series option.

sizevarvalues in this column are mapped to actual pixel values using the sizeAxis option.

date.format

if timevar is of class Date then this argument specifies how the dates are reformatted to be used by JavaScript.

options

list of configuration options for Google Motion Chart. The options are documented in detail by Google online: https://developers.google.com/chart/interactive/docs/gallery/motionchart#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

• Boolean arguments are set to either TRUE or FALSE, using the R syntax.

• Google API parameters with a single value and with names that don’t include a “.” are set like one would do in R, that is options=list(width=200, height=300). Exceptions to this rule are the width and height options for gvisAnnotatedTimeLine and gvisAnnotationChart. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options=list(width="200px", height="300px").

• Google API parameters with names that don’t include a “.”, but require multivalues are set as a character, wrapped in "[]" and separated by commas, e.g. options=list(colors="["#cbb69d", '#603913', '#c69c6e']")
• Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as 'true' or 'false'. For example the Google documentation states the formatting options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as:

```r
options=list(vAxis="[format:'#,###\%']")
```

• If several sub-options have to be set, e.g.

```r
titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize,
```

then those can be combined in one list item such as:

```r
options=list(titleTextStyle="[color:'red',fontName:'Courier',fontSize:16']")
```

• parameters that can have more than one value per sub-options are wrapped in "["]". For example to set the labels for left and right axes use:

```r
options=list(vAxes="[title:'val1'],[title:'val2']")
```

• gvis.editor a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

dismiss

### chartid

character. If missing (default) a random chart id will be generated based on chart type and tempfile

dismiss

#### Value

gvisMotionChart returns list of **class** "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

- **type** Google visualisation type
- **chartid** character id of the chart object. Unique chart ids are required to place several charts on the same page.
- **html** a list with the building blocks for a page
  - **header** a character string of a html page header: `<html>...<body>`,
  - **chart** a named character vector of the chart's building blocks:
    - **jsHeader** Opening `<script>` tag and reference to Google's JavaScript library.
    - **jsData** JavaScript function defining the input data as a JSON object.
    - **jsDrawChart** JavaScript function combing the data with the visualisation API and user options.
    - **jsDisplayChart** JavaScript function calling the handler to display the chart.
    - **jsFooter** End tag `</script>`.
    - **jsChart** Call of the jsDisplayChart function.
    - **divChart** `<div>` container to embed the chart into the page.
  - **caption** character string of a standard caption, including data name and chart id.
  - **footer** character string of a html page footer: `</body>...</html>`, including the used R and googleVis version and link to Google's Terms of Use.

#### Warnings

Because of Flash security settings the chart might not work correctly when accessed from a file location in the browser (e.g., `file:///c:/webhost/myhost/myviz.html`) rather than from a web server URL (e.g. `https://www.myhost.com/myviz.html`). See the googleVis package vignette and the Macromedia web site ([https://www.macromedia.com/support/documentation/en/flashplayer/help/](https://www.macromedia.com/support/documentation/en/flashplayer/help/)) for more details.
Note

Please note that a timevar with values less than 100 will be shown as years 19xx.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>, Diego de Castillo <decastillo@gmail.com>

References

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/motionchart

See Also

See also print.gvis, plot.gvis for printing and plotting methods.

Examples

## Please note that by default the googleVis plot command
## will open a browser window and requires Flash and Internet
## connection to display the visualisation.
M1 <- gvisMotionChart(Fruits, idvar="Fruit", timevar="Year")
plot(M1)

## Not run:
## Usage of date variable
M2 <- gvisMotionChart(Fruits, idvar="Fruit", timevar="Date",
  date.format = "\%Y\%m\%d")
plot(M2)

## Display weekly data:
M3 <- gvisMotionChart(Fruits, "Fruit", "Date", date.format="\%YW\%W")
plot(M3)

## End(Not run)
## Options: no side panel on the right
M4 <- gvisMotionChart(Fruits,"Fruit", "Year",
  options=list(showSidePanel=FALSE))
plot(M4)

## Options: trails un-ticked
M5 <- gvisMotionChart(Fruits, "Fruit", "Year",
  options=list(state=('"showTrails":false;'))
plot(M5)

## You can change some of displaying settings via the browser,
## e.g. the level of opacity of non-selected items, or the chart type.
## The state string from the 'Advanced' tab can be used to set those
## settings via R. Just copy and past the string from the browser into
## the argument state of the options list.
## Here is an example of a motion chart, with an initial line chart
## displayed. Ensure that you have a newline at the start and end of
## your settings string.
myStateSettings <-'

```r
myStateSettings <-
```
## gvisOrgChart

**Google Org Chart with R**

### Description

An organizational chart that supports selection.

### Usage

```r
gvisOrgChart(
  data,
  idvar = "",
  parentvar = "",
  tipvar = "",
  options = list(),
  chartid
)
```
**Arguments**

**data**

A `data.frame`. The data has to have at least three columns. Each row in the data table describes one node (a rectangle in the graph). Each node (except the root node) has one or more parent nodes. Each node is sized and colored according to its values relative to the other nodes currently shown.

**idvar**

Column name of data describing the ID for each node. It should be unique among all nodes, and can include any characters, including spaces. This is shown on the node. You can specify a formatted value to show on the chart instead, but the unformatted value is still used as the ID.

**parentvar**

Column name of data that match to entries in `idvar`. If this is a root node, leave this `NA`. Only one root is allowed.

**tipvar**

Column name of data for the tip variable. Tool-tip text to show, when a user hovers over this node.

**options**

List of configuration options, see:

https://developers.google.com/chart/interactive/docs/gallery/orgchart#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- Boolean arguments are set to either `TRUE` or `FALSE`, using the R syntax.
- Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is `options=list(width=200,height=300)`.
  
  Exceptions to this rule are the width and height options for `gvisAnnotatedTimeline` and `gvisAnnotationChart`. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, `options=list(width="200px",height="300px")`.
- Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g.
  ```
  options=list(colors="[#cbb69d,'#603913,'#c69c6e']")
  ```
  - Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as ‘true’ or ‘false’. For example the Google documentation states the formatting options for the vertical axis and states the parameter as `vAxis.format`. Then this parameter can be set in R as:
    ```
    options=list(vAxis="{format:'#,###%'}")
    ```
  - If several sub-options have to be set, e.g.
    ```
    titleTextStyle.color,titleTextStyle.fontName and titleTextStyle.fontSize
    ```
    then those can be combined in one list item such as:
    ```
    options=list(titleTextStyle="{color:‘red’,fontName:’Courier’,fontSize:16’}
    ```
    - Parameters that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use:
      ```
      options=list(vAxes="[{title:‘val1’},{title:‘val2’}]’
      ```
    - `gvis.editor` a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

**chartid**

Character. If missing (default) a random chart id will be generated based on chart type and `tempfile`
Details

The gvisOrgChart function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser.

Value

gvisOrgChart returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

type Google visualisation type
chartid character id of the chart object. Unique chart ids are required to place several charts on the same page.
html a list with the building blocks for a page
header a character string of a html page header: <html>...<body>,
chart a named character vector of the chart's building blocks:
  jsheader Opening <script> tag and reference to Google's JavaScript library.
  jsData JavaScript function defining the input data as a JSON object.
  jsDrawChart JavaScript function combining the data with the visualisation API and user options.
  jsDisplayChart JavaScript function calling the handler to display the chart.
  jsFooter End tag </script>.
  jsChart Call of the jsDisplayChart function.
divChart <div> container to embed the chart into the page.
caption character string of a standard caption, including data name and chart id.
footer character string of a html page footer: </body>...</html>, including the used R and googleVis version and link to Google's Terms of Use.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

References

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/orgchart

See Also

See also print.gvis, plot.gvis for printing and plotting methods.

Examples

## Please note that by default the googleVis plot command
## will open a browser window and requires Internet
## connection to display the visualisation.

Regions
Org1 <- gvisOrgChart(Regions, idvar = "Region", parentvar = "Parent", tipvar="Val")
plot(Org1)

## Set a few options
Org2 <- gvisOrgChart(Regions, idvar = "Region", parentvar = "Parent",
    tipvar="Val",
    options=list(width=600, height=400,
                size='large', allowCollapse=TRUE))
plot(Org2)

---

**gvisPieChart**

*Google Pie Chart with R*

**Description**

The `gvisPieChart` function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser using SVG or VML.

**Usage**

```r
gvisPieChart(data, labelvar = "", numvar = "", options = list(), chartid)
```

**Arguments**

- `data`: a data.frame to be displayed as a pie chart.
- `labelvar`: Name of the character column which contains the category labels for the slice labels.
- `numvar`: a vector of column names of the numerical variables of the slice values.
- `options`: list of configuration options for Google Pie Charts, see: [https://developers.google.com/chart/interactive/docs/gallery/piechart#Configuration_Options](https://developers.google.com/chart/interactive/docs/gallery/piechart#Configuration_Options).

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- **Boolean arguments**: are set to either TRUE or FALSE, using the R syntax.
- **Google API parameters**: are set like one would do in R, that is `options=list(width=200,height=300)`. Exceptions to this rule are the width and height options for `gvisAnnotatedTimeLine` and `gvisAnnotationChart`. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, `options=list(width="200px",height="300px")`.
- **Google API parameters**: require multivalues are set as a character, wrapped in ":[ ]" and separated by commas, e.g. `options=list(colors="["#cbb69d","#603913","#c69c6e"]")`
- **Google API parameters**: that don’t include a ":" present parameters with several sub-options and have to be set as a character wrapped in ".". The values of those sub-options are set via parameter:value. For example the Google documentation states the formatting options for the vertical axis and states the parameter as `vAxis.format`. Then this parameter can be set in R as: `options=list(vAxis="{format:\"#,###\%\"})`.

---
If several sub-options have to be set, e.g. `titleTextStyle.color`, `titleTextStyle.fontName` and `titleTextStyle.fontSize`, then those can be combined in one list item such as:

```r
options=list(titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}')
```

Parameters that can have more than one value per sub-options are wrapped in "[]". For example to set the labels for left and right axes use:

```r
options=list(vAxes="[{title:'val1'}, {title:'val2'}]")
```

`gvis.editor` a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

The `chartid` character. If missing (default) a random chart id will be generated based on chart type and `tempfile`

**Value**

gvisPieChart returns list of `class"gvis"` and "list". An object of class "gvis" is a list containing at least the following components:

- **type** Google visualisation type
- **chartid** character id of the chart object. Unique chart ids are required to place several charts on the same page.
- **html** a list with the building blocks for a page
  - **header** a character string of a html page header: `<html>...<body>`
  - **chart** a named character vector of the chart's building blocks:
    - **jsHeader** Opening `<script>` tag and reference to Google's JavaScript library.
    - **jsData** JavaScript function defining the input data as a JSON object.
    - **jsDrawChart** JavaScript function combing the data with the visualisation API and user options.
    - **jsDisplayChart** JavaScript function calling the handler to display the chart.
    - **jsFooter** End tag `</script>`.
    - **jsChart** Call of the `jsDisplayChart` function.
    - **divChart** `<div>` container to embed the chart into the page.
  - **caption** character string of a standard caption, including data name and chart id.
  - **footer** character string of a html page footer: `</body>...</html>`, including the used R and googleVis version and link to Google's Terms of Use.

**Author(s)**

Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

**References**

Google Chart Tools API: [https://developers.google.com/chart/interactive/docs/gallery/piechart](https://developers.google.com/chart/interactive/docs/gallery/piechart)

**See Also**

See also `print.gvis`, `plot.gvis` for printing and plotting methods
Examples

```r
## Please note that by default the googleVis plot command
## will open a browser window and requires an internet
## connection to display the visualisation.

Pie1 <- gvisPieChart(CityPopularity)
plot(Pie1)

## Doughnut chart - a pie with a hole
Pie2 <- gvisPieChart(CityPopularity, options=list(
  slices="(4: {offset: 0.2}, 0: {offset: 0.3})",
  title='City popularity',
  legend='none',
  pieSliceText='label',
  pieHole=0.5))
plot(Pie2)
```

---

gvisSankey Google Sankey Chart with R

Description

A sankey diagram is a visualization used to depict a flow from one set of values to another. The things being connected are called nodes and the connections are called links. They’re named after Captain Sankey, who created a diagram of steam engine efficiency that used arrows having widths proportional to heat loss.

Usage

gvisSankey(data, from = "", to = "", weight = "", options = list(), chartid)

Arguments

data data.frame that contains the data to be visualised
from a string that refers to the column name in data for the source nodes to be used
to a string that refers to the column name in data for the destination nodes to be used
weight name of the column with the numerical weight of the connections
options list of configuration options. The options are documented in detail by Google online:
https://developers.google.com/chart/interactive/docs/gallery/sankey#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- Boolean arguments are set to either TRUE or FALSE, using the R syntax.
**gvisSankey**

- Google API parameters with a single value and with names that don't include a "." are set like one would do in R, that is `options=list(width=200,height=300)`. Exceptions to this rule are the width and height options for `gvisAnnotatedTimeLine` and `gvisAnnotationChart`. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, `options=list(width="200px",height="300px")`.

- Google API parameters with names that don't include a ".", but require multivalues are set as a character, wrapped in "]" and separated by commas, e.g. `options=list(colors=["#cbb69d","603913","c69c6e"])

- Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as 'true' or 'false'. For example the Google documentation states the formatting options for the vertical axis and states the parameter as `vAxis.format`. Then this parameter can be set in R as: `options=list(vAxis="{format:'#,###%'}")`.

- If several sub-options have to be set, e.g. `titleTextStyle.color`, `titleTextStyle.fontName` and `titleTextStyle.fontSize`, then those can be combined in one list item such as: `options=list(titleTextStyle=("color:'red',fontName:'Courier',fontSize:16")

- parameters that can have more than one value per sub-options are wrapped in "] ". For example to set the labels for left and right axes use: `options=list(vAxes=[{"title:'val1'},{title:'val2'}])`

- `gvis.editor` a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

**chartid**

character. If missing (default) a random chart id will be generated based on chart type and `tempfile`

---

**Value**

gvisSankey returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

- **type** Google visualisation type
- **chartid** character id of the chart object. Unique chart ids are required to place several charts on the same page.
- **html** a list with the building blocks for a page
  - **header** a character string of a html page header: `<html>...<body>
  - **chart** a named character vector of the chart's building blocks:
    - **jsHeader** Opening `<script>` tag and reference to Google's JavaScript library.
    - **jsData** JavaScript function defining the input data as a JSON object.
    - **jsDrawChart** JavaScript function combing the data with the visualisation API and user options.
    - **jsDisplayChart** JavaScript function calling the handler to display the chart.
    - **jsFooter** End tag `</script>`.
    - **jsChart** Call of the `jsDisplayChart` function.
    - **divChart** `<div>` container to embed the chart into the page.
caption character string of a standard caption, including data name and chart id.
footer character string of a html page footer: </body>...</html>, including the used R and googleVis version and link to Google's Terms of Use.

Warning
The sankey chart may be undergoing substantial revisions in future Google Charts releases.

Author(s)
Markus Gesmann <markus.gesmann@gmail.com>

References
Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/sankey

Examples

dat <- data.frame(From=c(rep("A", 3), rep("B", 3)),
To=c(rep(c("X", "Y", "Z"),2)),
Weight=c(5,7,6,2,9,4))

sk1 <- gvisSankey(dat, from="From", to="To", weight="Weight")
plot(sk1)

sk2 <- gvisSankey(dat, from="From", to="To", weight="Weight",
options=list(sankey="{link: {color: { fill: #d799ae } }
node: { color: { fill: #a61d4c },
label: { color: '#871b47' } }}"))

plot(sk2)

gvisScatterChart function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser using SVG or VML.

Usage
gvisScatterChart(data, options = list(), chartid)

Arguments
data a data.frame to be displayed as a scatter chart. Two or more columns are required, all must be numeric. The values in the first column are used for the X-axis. The values in following columns are used for the Y-axis. Each column is displayed with a separate color.
gvisScatterChart

options list of configuration options, see:
https://developers.google.com/chart/interactive/docs/gallery/scatterchart#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- Boolean arguments are set to either TRUE or FALSE, using the R syntax.
- Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is options=list(width=200, height=300).
- Exceptions to this rule are the width and height options for gvisAnnotatedTimeLine and gvisAnnotationChart. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options=list(width="200px", height="300px").
- Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g.
  options=list(colors="[#cbb69d,'#603913,'#c69c6e']")
- Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter: value. For example the Google documentation states the formatting options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as:
  options=list(vAxis="{format:'#,###%'}").
- If several sub-options have to be set, e.g.
  titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize then those can be combined in one list item such as:
  options=list(titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}")
- Parameters that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use:
  options=list(vAxes="[{title:'val1'}, {title:'val2'}]"")
- gvis.editor a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

chartid character. If missing (default) a random chart id will be generated based on chart type and tempfile

Value

gvisScatterChart returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

type Google visualisation type

chartid character id of the chart object. Unique chart ids are required to place several charts on the same page.

html a list with the building blocks for a page
  header a character string of a html page header: <html>...<body>,
  chart a named character vector of the chart’s building blocks:
    jsHeader Opening <script> tag and reference to Google’s JavaScript library.
    jsData JavaScript function defining the input data as a JSON object.
jsDrawChart  JavaScript function combing the data with the visualisation API and user options.
jsDisplayChart  JavaScript function calling the handler to display the chart.
sfFooter  End tag </script>.
sfChart  Call of the jsDisplayChart function.
divChart  <div> container to embed the chart into the page.
caption  character string of a standard caption, including data name and chart id.
footer  character string of a html page footer: </body>...</html>, including the used R and googleVis version and link to Google’s Terms of Use.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

References

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/scatterchart

See Also

See also print.gvis, plot.gvis for printing and plotting methods

Examples

## Please note that by default the googleVis plot command
## will open a browser window and requires an internet
## connection to display the visualisation.

## Scatter chart
Scatter1 <- gvisScatterChart(women)
plot(Scatter1)

## Using optional arguments
Scatter2 <- gvisScatterChart(women, options=list(legend="none",
lineWidth=2, pointSize=2,
title="Women", vAxis="{title: 'weight (lbs)'}",
crosshair="[ trigger: 'both' ]",
hAxis="{title: 'height (in)'}", width=500, height=400))
plot(Scatter2)

df=data.frame(x=sin(1:100/3),
    Circle=cos(1:100/3),
    Ellipse=cos(1:100/3)*0.5)

## Plot several variables as smooth curves
Scatter3 <- gvisScatterChart(df,
options=list(curveType="function",
    pointSize=0,
    lineWidth=2))
plot(Scatter3)
## Two series in the same plot with different x-values

```r
df <- data.frame(x=c(2,2,1,3,4),
y1=c(0,3,NA,NA,NA),
y2=c(NA,NA,0,3,2))
```

```r
Scatter4 <- gvisScatterChart(df, 
    options=list(lineWidth=2, 
                 pointSize=2))
plot(Scatter4)
```

## Customize points

```r
M <- matrix(nrow=6,ncol=6)
M[as.matrix(M)==1:6] <- 1:6
dat <- data.frame(X=1:6, M)
SC <- gvisScatterChart(dat, 
    options=list(
        title="Customizing points",
        legend="right",
        pointSize=30,
        series="[
            0: { pointShape: 'circle' },
            1: { pointShape: 'triangle' },
            2: { pointShape: 'square' },
            3: { pointShape: 'diamond' },
            4: { pointShape: 'star' },
            5: { pointShape: 'polygon' }
        ]")
)
plot(SC)
```

---

gvisSteppedAreaChart  Google Stepped Area Chart with R

### Description

The gvisSteppedAreaChart function reads a data.frame and creates text output referring to the Google Visualisation API, which can be included into a web page, or as a stand-alone page.

### Usage

```r
gvisSteppedAreaChart(data, xvar = "", yvar = ", options = list(), chartid)
```

### Arguments

- **data**
  - a `data.frame` to be displayed as a stepped area chart.
- **xvar**
  - name of the character column which contains the category labels for the x-axes.
- **yvar**
  - a vector of column names of the numerical variables to be plotted. Each column is displayed as a separate line.
- **options**
  - list of configuration options, see:
    - https://developers.google.com/chart/interactive/docs/gallery/steppedarechart#Configuration_Options
  - The parameters can be set via a named list. The parameters have to map those of the Google documentation.
• Boolean arguments are set to either TRUE or FALSE, using the R syntax.
• Google API parameters with a single value and with names that don’t include a “.” are set like one would do in R, that is options=list(width=200,height=300). Exceptions to this rule are the width and height options for gvisAnnotatedTimeLine and gvisAnnotationChart. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options=list(width="200px", height="300px").
• Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "["] and separated by commas, e.g.

options=list(colors="['#cbb69d', '#603913', '#c69c6e']")

• Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as 'true' or 'false'. For example the Google documentation states the formating options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as:

options=list(vAxis="{format:'#,###%'}")

• If several sub-options have to be set, e.g. titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize, then those can be combined in one list item such as:

options=list(titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}'")

parameters that can have more than one value per sub-options are wrapped in "["] . For example to set the labels for left and right axes use:

options=list(vAxes="[{title:'val1'},{title:'val2'}]")

• gvis.editor a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

chartid character. If missing (default) a random chart id will be generated based on chart type and tempfile

Details

The stepped area chart is rendered within the browser using SVG or VML and displays tips when hovering over points.

Value

gvisSteppedAreaChart returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

type Google visualisation type
chartid character id of the chart object. Unique chart ids are required to place several charts on the same page.

html a list with the building blocks for a page

header a character string of a html page header: <html>...<body>,
chart a named character vector of the chart’s building blocks:

jshoster Opening <script> tag and reference to Google’s JavaScript library.
jsData JavaScript function defining the input data as a JSON object.
jsDrawChart  JavaScript function combing the data with the visualisation API and user options.
sjsDisplayChart  JavaScript function calling the handler to display the chart.
sjsFooter  End tag </script>.
sjsChart  Call of the jsDisplayChart function.
divChart  <div> container to embed the chart into the page.
caption  character string of a standard caption, including data name and chart id.
footer  character string of a html page footer: </body>...</html>, including the used R and googleVis version and link to Google’s Terms of Use.

Author(s)
Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

References
Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/steppedarechart

See Also
See also print.gvis, plot.gvis for printing and plotting methods

Examples
## Please note that by default the googleVis plot command
## will open a browser window and requires an internet
## connection to display the visualisation.

df=data.frame(country=c("US", "GB", "BR"), val1=c(1,3,4), val2=c(23,12,32))

## Stepped Area chart
SteppedArea1 <- gvisSteppedAreaChart(df, xvar="country", yvar=c("val1", "val2"))
plot(SteppedArea1)

## Stacked chart
SteppedArea2 <- gvisSteppedAreaChart(df, xvar="country", yvar=c("val1", "val2"),
  options=list(isStacked=TRUE))
plot(SteppedArea2)

## Add a customised title
SteppedArea3 <- gvisSteppedAreaChart(df, xvar="country", yvar=c("val1", "val2"),
  options=list(title="Hello World",
               titleTextStyle=\"{color:'red',fontName:'Courier',fontSize:16}\"))
plot(SteppedArea3)

## Not run:
## Change y-axis to percentages
SteppedArea3 <- gvisSteppedAreaChart(df, xvar="country", yvar=c("val1", "val2"),
  options=list(vAxis=\"{format:'#,###%'}\"))
plot(SteppedArea3)

## End(Not run)
**gvisTable**

**Google Table Chart with R**

**Description**

The `gvisTable` function reads a `data.frame` and creates text output referring to the Google Visualization API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser.

**Usage**

```r
gvisTable(data, options = list(), chartid, formats = NULL)
```

**Arguments**

- `data`: a `data.frame` to be displayed as a table.
- `options`: list of configuration options, see:
  
  https://developers.google.com/chart/interactive/docs/gallery/table#Configuration_Options

  The parameters can be set via a named list. The parameters have to map those of the Google documentation.

  - Boolean arguments are set to either `TRUE` or `FALSE`, using the R syntax.
  - Google API parameters with a single value and with names that don’t include a ",." are set like one would do in R, that is `options=list(width=200, height=300)`. Exceptions to this rule are the width and height options for `gvisAnnotatedTimeLine` and `gvisAnnotationChart`. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, `options=list(width="200px", height="300px")`.
  - Google API parameters with names that don’t include a ",." but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g.
    ```
    options=list(colors="[#cbb69d,'#603913,'#c69c6e']")
    ```
  - Google API parameters with names that do include a ",." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as ‘true’ or ‘false’. For example the Google documentaion states the formating options for the vertical axis and states the parameter as `vAxis.format`. Then this paramter can be set in R as:
    ```
    options=list(vAxis="{"format:‘#,###%’}\")
    ```
  - If several sub-options have to be set, e.g.
    ```
    titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize,
    ```
    then those can be combined in one list item such as:
    ```
    options=list(titleTextStyle="{"color:‘red’,fontName:‘Courier’,fontSize:16”}\")
    ```
  - Parameters that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use:
    ```
    options=list(vAxes="{"title:‘val1’},{title:‘val2’}\")
    ```
• `gvis.editor` a character label for an on-page button that opens an in-page
dialog box enabling users to edit, change and customise the chart. By de-
fault no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

**chartid** character. If missing (default) a random chart id will be generated based on chart
type and **tempfile**

**formats** named list. If NULL (default) no specific format will be used. The named list
needs to contain the column names of the data and the specified format. The
format string is a subset of the ICU pattern set. For instance, pattern:
"#,###%" will result in output values "1.000%", "750%", and "50%" for values 10, 7.5,
and 0.5.

**Details**

A table that can be sorted and paged. Table cells can be formatted using format strings, or by directly
inserting HTML as cell values. Numeric values are right-aligned; boolean values are displayed as
check marks. Users can select single rows either with the keyboard or the mouse. Users can sort
rows by clicking on column headers. The header row remains fixed as the user scrolls. The table
fires a number of events corresponding to user interaction.

**Value**

gvisTable returns list of class "gvis" and "list". An object of class "gvis" is a list containing at
least the following components:

**type** Google visualisation type

**chartid** character id of the chart object. Unique chart ids are required to place several charts on
the same page.

**html** a list with the building blocks for a page

- **header** a character string of a html page header: `<html>...</body>`,
- **chart** a named character vector of the chart’s building blocks:
  - `jsHeader` Opening `<script>` tag and reference to Google’s JavaScript library.
  - `jsData` JavaScript function defining the input data as a JSON object.
  - `jsDrawChart` JavaScript function combing the data with the visualisation API and user
  options.
  - `jsDisplayChart` JavaScript function calling the handler to display the chart.
  - `jsFooter` End tag `</script>`,
  - `jsChart` Call of the `jsDisplayChart` function.
  - `divChart` `<div>` container to embed the chart into the page.
- **caption** character string of a standard caption, including data name and chart id.
- **footer** character string of a html page footer: `</body>...</html>`, including the used R
  and googleVis version and link to Google’s Terms of Use.

**Author(s)**

Markus Gesmann <markus.gesmann@gmail.com>,

Diego de Castillo <decastillo@gmail.com>
gvisTimeline

References

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/table

See Also

See also print.gvis, plot.gvis for printing and plotting methods.

Examples

## Please note that by default the googleVis plot command
## will open a browser window and requires Flash and Internet
## connection to display the visualisation.

## Table with links to wikipedia (flags)
tbl1 <- gvisTable(Population)
plot(tbl1)

## Table with enabled paging
.tbl2 <- gvisTable(Population, options=list(page='enable',
height='automatic',
width='automatic'))

plot(tbl2)

## Table with formating options
.tbl3 <- gvisTable(Population, formats=list(Population="#,###"))
Populations["% of World Population"] <- Populations["% of World Population"] / 100
.tbl4 <- gvisTable(Populations, formats=list(Populations="#,###",
' % of World Population'="#.#' ))

plot(tbl4)

---

gvisTimeline

Google Timeline Chart with R

Description

A timeline is a chart that depicts how a set of resources are used over time. One popular type of
timeline is the Gantt chart.

Usage

gvisTimeline(
  data,
  rowlabel = "", 
  barlabel = "", 
  start = "", 
  end = "", 
  options = list(), 
  chartid
)
Arguments

data data.frame that contains the data to be visualised
rowlabel a string that refers to the column name in data for the row labels to be used
barlabel a string that refers to the column name in data for the bar labels to be used
start number, date or datetime for the start dates
end number, date or datetime for the end dates
options list of configuration options. The options are documented in detail by Google online:
https://developers.google.com/chart/interactive/docs/gallery/timeline#Configuration_Options

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- Boolean arguments are set to either TRUE or FALSE, using the R syntax.
- Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is options=list(width=200, height=300).
  Exceptions to this rule are the width and height options for gvisAnnotatedTimeLine and gvisAnnotationChart. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options=list(width="200px", height="300px").
- Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "] [" and separated by commas, e.g.
  options=list(colors="['#cbb69d', '#603913', '#c69c6e']")
- Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as ‘true’ or ‘false’. For example the Google documentation states the formating options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as:
  options=list(vAxis="{format: '#,###%'}").
- If several sub-options have to be set, e.g.
  titleTextStyle.color, titleTextStyle.fontName and titleTextStyle.fontSize, then those can be combined in one list item such as:
  options=list(titleTextStyle="[color: ‘red’, fontName: ‘Courier’, fontSize:16]")
- parameters that can have more than one value per sub-options are wrapped in "[ "].
  For example to set the labels for left and right axes use:
  options=list(vAxes="[title: ‘val1’], [title: ‘val2’]"")

For more details see the Google API documentation and the R examples below.

Value

gvisTimeline returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:
type Google visualisation type
chartId character id of the chart object. Unique chart ids are required to place several charts on the same page.
html a list with the building blocks for a page
  header a character string of a html page header: <html>...<body>,
  chart a named character vector of the chart’s building blocks:
    jsHeader Opening <script> tag and reference to Google’s JavaScript library.
    jsData JavaScript function defining the input data as a JSON object.
    jsDrawChart JavaScript function combining the data with the visualisation API and user options.
    jsDisplayChart JavaScript function calling the handler to display the chart.
    jsFooter End tag </script>.
    jsChart Call of the jsDisplayChart function.
    divChart <div> container to embed the chart into the page.
  caption character string of a standard caption, including data name and chart id.
footer character string of a html page footer: </body>...</html>, including the used R and googleVis version and link to Google’s Terms of Use.

Author(s)
Markus Gesmann <markus.gesmann@gmail.com>

References
Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/timeline

Examples

dat <- data.frame(Term=c("1","2","3"),
                   President=c("Washington", "Adams", "Jefferson"),
                   start=as.Date(x=c("1789-03-29", "1797-02-03", "1801-02-03")),
                   end=as.Date(x=c("1797-02-03", "1801-02-03", "1809-02-03")))
tl <- gvisTimeline(data=dat[-1], rowlabel="President",
                    start="start", end="end")
plot(tl)

tl <- gvisTimeline(data=dat, barlabel="President",
                    start="start", end="end")
plot(tl)

tl <- gvisTimeline(data=dat, rowlabel="President",
                    start="start", end="end",
                    options=list(timeline="{showRowLabels:false}"))
plot(tl)

dat <- data.frame(Position=c(rep("President", 3), rep("Vice", 3)),
                   Name=c("Washington", "Adams", "Jefferson",
                          "Adams", "Jefferson", "Burr"),
                   start=as.Date(x=rep(c("1789-03-29", "1797-02-03", "1801-02-03"),2)),
                   end=as.Date(x=rep(c("1797-02-03", "1801-02-03", "1809-02-03"),2)))
gvisTreeMap

Google Tree Map with R

Description

The gvisTreeMap function reads a data.frame and creates text output referring to the Google Visualization API, which can be included into a web page, or as a stand-alone page. The actual chart is rendered by the web browser.

Usage

gvisTreeMap(
data,
idvar = "",
parentvar = "",
sizevar = "",
colorvar = "",
options = list())
Arguments

data a data.frame. The data has to have at least four columns. Each row in the data
table describes one node (a rectangle in the graph). Each node (except the root
node) has one or more parent nodes. Each node is sized and colored according
to its values relative to the other nodes currently shown.

idvar column name of data describing the ID for each node. It can be any valid
JavaScript string, including spaces, and any length that a string can hold. This
value is displayed as the node header.

parentvar column name of data that match to entries in idvar. If this is a root node, leave
this NA. Only one root is allowed per treemap.

sizevar column name of data with positive values to define the size of maps. Any
positive value is allowed. This value determines the size of the node, computed
relative to all other nodes currently shown. This value is ignored for non-leaf
nodes (it is actually calculated from the size of all its children).

colorvar column name of data with values to define range of color. The value is used to
calculate a color for this node. Any value, positive or negative, is allowed. The
color value is first recomputed on a scale from minColorValue to maxColorValue,
and then the node is assigned a color from the gradient between minColor and
maxColor.

options list of configuration options, see:
https://developers.google.com/chart/interactive/docs/gallery/treemap#Configuration_Options
The parameters can be set via a named list. The parameters have to map those
of the Google documentation.

• Boolean arguments are set to either TRUE or FALSE, using the R syntax.
• Google API parameters with a single value and with names that don’t in-
clude a "." are set like one would do in R, that is options=list(width=200,height=300).
Exceptions to this rule are the width and height options for gvisAnnotatedTimeLine
and gvisAnnotationChart. For those two functions, width and height
must be character strings of the format "Xpx", where X is a number, or
"automatic". For example, options=list(width="200px",height="300px").
• Google API parameters with names that don’t include a ".", but require mul-
tivalues are set as a character, wrapped in "[ ]" and separated by commas,
e.g.
  options=list(colors=["#cbb69d,'#683913,'#c69c6e'])
• Google API parameters with names that do include a "." present parameters
with several sub-options and have to be set as a character wrapped in " ".
  The values of those sub-options are set via parameter:value. Boolean
values have to be stated as ‘true’ or ‘false’. For example the Google
documentaion states the formatting options for the vertical axis and states
the parameter as vAxis.format. Then this paramter can be set in R as:
  options=list(vAxis="{format:’#,###’}").
• If several sub-options have to be set, e.g.
titleTextStyle.color,titleTextStyle.fontName and titleTextStyle.fontSize,
then those can be combined in one list item such as:
  options=list(titleTextStyle="{color:’red’,fontName:’Courier’,fontSize:16’}")
• parameters that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use:
  ```
  options=list(vAxes="[{title: 'val1'}, {title: 'val2'}]"
  ```

• gvis.editor a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

For more details see the Google API documentation and the R examples below.

chartid character. If missing (default) a random chart id will be generated based on chart type and tempfile

Details

A tree map is a visual representation of a data tree, where each node can have zero or more children, and one parent (except for the root, which has no parents). Each node is displayed as a rectangle, sized and colored according to values that you assign. Sizes and colors are valued relative to all other nodes in the graph. You can specify how many levels to display simultaneously, and optionally to display deeper levels in a hinted fashion. If a node is a leaf node, you can specify a size and color; if it is not a leaf, it will be displayed as a bounding box for leaf nodes. The default behavior is to move down the tree when a user left-clicks a node, and to move back up the tree when a user right-clicks the graph.

The total size of the graph is determined by the size of the containing element that you insert in your page. If you have leaf nodes with names too long to show, the name will be truncated with an ellipsis (...).

Value

`gvisTreeMap` returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

- `type` Google visualisation type
- `chartid` character id of the chart object. Unique chart ids are required to place several charts on the same page.
- `html` a list with the building blocks for a page
  - `header` a character string of a html page header: `<html>...<body>`,
  - `chart` a named character vector of the chart’s building blocks:
    - `jsHeader` Opening `<script>` tag and reference to Google’s JavaScript library.
    - `jsData` JavaScript function defining the input data as a JSON object.
    - `jsDrawChart` JavaScript function combing the data with the visualisation API and user options.
    - `jsDisplayChart` JavaScript function calling the handler to display the chart.
    - `jsFooter` End tag `</script>`.
    - `jsChart` Call of the `jsDisplayChart` function.
    - `divChart` `<div>` container to embed the chart into the page.
- `caption` character string of a standard caption, including data name and chart id.
- `footer` character string of a html page footer: `</body>...</html>`, including the used R and googleVis version and link to Google’s Terms of Use.

Warning

Tree maps display a tree like structure where every child has to have a unique parent.
Values in column `sizevar` should be greater than zero and finite.
Author(s)

Markus Gesmann <markus.gesmann@gmail.com>,
Diego de Castillo <decastillo@gmail.com>

References

Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/treemap

See Also

See also print.gvis, plot.gvis for printing and plotting methods.

Please note that the treemap package offers a static version of tree maps via its tmPlot function.

Examples

## Please note that by default the googleVis plot command
## will open a browser window and requires Internet
## connection to display the visualisation.

Tree <- gvisTreeMap(Regions, idvar="Region", parentvar="Parent",
                   sizevar="Val", colorvar="Fac")
plot(Tree)

Tree2 <- gvisTreeMap(Regions, "Region", "Parent", "Val", "Fac",
                    options=list(width=600, height=500,
                                  fontSize=16,
                                  minColor="#EDF8FB",
                                  midColor="#66C2A4",
                                  maxColor="#006D2C",
                                  headerHeight=20,
                                  fontColor='black',
                                  showScale=TRUE))
plot(Tree2)

## Simple static treemap with no drill down options based on US states
## and their area. However we still have to create a parent id to use
## gvisTreeMap

require(datasets)
states <- data.frame(state.name, state.area)

## Create parent variable

total=data.frame(state.area=sum(states$state.area), state.name="USA")

my.states <- rbind(total, states)
my.states$parent="USA"

## Set parent variable to NA at root level
my.states$parent[my.states$state.name=="USA"] <- NA

my.states$state.area.log=log(my.states$state.area)
statesTree <- gvisTreeMap(my.states, "state.name", "parent",
                         idvar="state.name", parentvar="parent",
                         sizevar="state.area", colorvar="state.area",
                         options=list(width=600, height=500,
                                       fontSize=16,
                                       minColor="#EDF8FB",
                                       midColor="#66C2A4",
                                       maxColor="#006D2C",
                                       headerHeight=20,
                                       fontColor='black',
                                       showScale=TRUE))
plot(statesTree)
"state.area", "state.area.log")

plot(statesTree)

## We add US regions to the above data set to enable drill down capabilities

states2 <- data.frame(state.region, state.name, state.area)

regions <- aggregate(list(region.area=states2$state.area),
                     list(region=state.region), sum)

my.states2 <- data.frame(regionid=c("USA",
                                  as.character(regions$region),
                                  as.character(states2$state.name)),
                          parentid=c(NA, rep("USA", 4),
                                     as.character(states2$state.region)),
                          state.area=c(sum(states2$state.area),
                                       regions$region.area, states2$state.area))

my.states2$state.area.log=log(my.states2$state.area)

statesTree2 <- gvisTreeMap(my.states2, "regionid", "parentid",
                           "state.area", "state.area.log")

plot(statesTree2)

## Now we add another layer with US divisions

states3 <- data.frame(state.region, state.division, state.name, state.area)

regions <- aggregate(list(region.area=states3$state.area),
                     list(region=state.region), sum)

divisions <- aggregate(list(division.area=states3$state.area),
                      list(division=state.division, region=state.region),
                      sum)

my.states3 <- data.frame(regionid=c("USA",
                                  as.character(regions$region),
                                  as.character(divisions$division),
                                  as.character(states3$state.name)),
                          parentid=c(NA, rep("USA", 4),
                                     as.character(divisions$region),
                                     as.character(states3$state.division)),
                          state.area=c(sum(states3$state.area),
                                       regions$region.area, divisions$division.area,
                                       states3$state.area))

my.states3$state.area.log=log(my.states3$state.area)

statesTree3 <- gvisTreeMap(my.states3, "regionid", "parentid",
                           "state.area", "state.area.log")

plot(statesTree3)
**gvisWordTree**  

**Google Word Tree with R**

**Description**

A word tree depicts multiple parallel sequences of words. It could be used to show which words most often follow or precede a target word (e.g., "Cats are...") or to show a hierarchy of terms (e.g., a decision tree).

**Usage**

```r
library(ggplot2)

# Example data
data <- data.frame(heading = c("Intro to R", "R basics", "R advanced"),
                   text = c("Learn the basics of R.", "Navigate the console.", "Graphs and data analysis."))

# Create a word tree
wt <- gvisWordTree(data = data, textvar = "heading", sizevar = "text")

# Display the word tree
wt$chart()
```

**Arguments**

- `data`: data.frame that contains the data to be visualised.
- `textvar`: a string that refers to the column name in `data` that contains the text to be used.
- `sizevar`: a string that refers to the column name in `data` that contains the size of the words in the word tree.
- `stylevar`: a string that refers to the column name in `data` that contains the style of the text. For no warning message in plot, should be called `<textvar>.style`.
- `idvar` (only when format is explicit): a string that refers to the column name in `data` that contains the unique ID of the text.
- `parentvar` (only when format is explicit): a string that refers to the column name in `data` that contains the ID of the parent of the text.
- `options`: list of configuration options, see: [https://developers.google.com/chart/interactive/docs/gallery/wordtree#Configuration_Options](https://developers.google.com/chart/interactive/docs/gallery/wordtree#Configuration_Options)

The parameters can be set via a named list. The parameters have to map those of the Google documentation.

- Boolean arguments are set to either TRUE or FALSE, using the R syntax.
- Google API parameters with a single value and with names that don’t include a "." are set like one would do in R, that is options=list(width=200,height=300).

Exceptions to this rule are the width and height options for `gvisAnnotatedTimeLine` and `gvisAnnotationChart`. For those two functions, width and height must be character strings of the format "Xpx", where X is a number, or "automatic". For example, options=list(width="200px",height="300px").
• Google API parameters with names that don’t include a ".", but require multivalues are set as a character, wrapped in "[ ]" and separated by commas, e.g.
  
  options=list(colors="[\#cb69d,\#603913,\#c69c6e]")

• Google API parameters with names that do include a "." present parameters with several sub-options and have to be set as a character wrapped in " ". The values of those sub-options are set via parameter:value. Boolean values have to be stated as 'true' or 'false'. For example the Google documentation states the formatting options for the vertical axis and states the parameter as vAxis.format. Then this parameter can be set in R as:
  
  options=list(vAxis="{format:\#,###%}")

• If several sub-options have to be set, e.g.:
  
  titleTextStyle.color,titleTextStyle.fontName and titleTextStyle.fontSize, then those can be combined in one list item such as:
  
  options=list(titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}")

• parameters that can have more than one value per sub-options are wrapped in "[ ]". For example to set the labels for left and right axes use:
  
  options=list(vAxes="[{title:'val1'}, {title:'val2'}]")

• gvis.editor a character label for an on-page button that opens an in-page dialog box enabling users to edit, change and customise the chart. By default no value is given and therefore no button is displayed.

  For more details see the Google API documentation and the R examples below.

method a string to say whether the word tree is either:

  • implicit - will weight and connect automatically, or
  
  • explicit - can specify weights and connections

chartid character. If missing (default) a random chart id will be generated based on chart type and tempfile

Value

gvisWordTree returns list of class "gvis" and "list". An object of class "gvis" is a list containing at least the following components:

type Google visualisation type
chartid character id of the chart object. Unique chart ids are required to place several charts on the same page.
html a list with the building blocks for a page

header a character string of a html page header: &lt;html&gt;...&lt;/body&gt;
chart a named character vector of the chart's building blocks:
  
  jsHeader Opening &lt;script&gt; tag and reference to Google’s JavaScript library.
  jsData JavaScript function defining the input data as a JSON object.
  jsDrawChart JavaScript function combing the data with the visualisation API and user options.
  jsDisplayChart JavaScript function calling the handler to display the chart.
  jsFooter End tag &lt;/script&gt;.
  jsChart Call of the jsDisplayChart function.
  divChart &lt;div&gt; container to embed the chart into the page.
caption character string of a standard caption, including data name and chart id.
footer character string of a html page footer: &lt;/body&gt;...&lt;/html&gt;, including the used R and googleVis version and link to Google’s Terms of Use.
Warning
The word tree chart may be undergoing substantial revisions in future Google Charts releases.

Author(s)
Ashley Baldry

References
Google Chart Tools API: https://developers.google.com/chart/interactive/docs/gallery/wordtree

Examples
## Please note that by default the googleVis plot command
## will open a browser window and requires Internet
## connection to display the visualisation.
wt1 <- gvisWordTree(Cats, textvar = "Phrase")
plot(wt1)

Cats2 <- Cats
Cats2$Phrase.style <- ifelse(Cats$Sentiment >= 7, "green",
ifelse(Cats$Sentiment <= 3, "red", "black"))

wt2 <- gvisWordTree(Cats2, textvar = "Phrase", stylevar = "Phrase.style",
options = list(fontName = "Times-Roman",
wordtree = "{word: 'cats'}",
backgroundColor = "#cba")

plot(wt2)

# Explicit word tree
exp.data <- data.frame(id = as.numeric(0:9),
label = letters[1:10],
parent = c(-1, 0, 0, 0, 2, 2, 4, 6, 1, 7),
size = c(10, 5, 3, 2, 2, 1, 1, 5, 1),
stringsAsFactors = FALSE)

wt3 <- gvisWordTree(exp.data, idvar = "id", textvar = "label",
parentvar = "parent", sizevar = "size",
options = list(wordtree = "{format: 'explicit'}"),
method = "explicit")

plot(wt3)

OpenClose

OpenClose: googleVis example data set

Description
Example data set to illustrate the use of the googleVis package.
**Usage**

data(OpenClose)

**Format**

A data frame with 5 observations on the following 5 variables.

- **Weekday**: a factor with levels Fri Mon Thurs Tues Wed
- **Low**: a numeric vector
- **Open**: a numeric vector
- **Close**: a numeric vector
- **High**: a numeric vector

**Source**


**Examples**

OpenClose
plot(gvisCandlestickChart(OpenClose, options=list(legend="none")))

---

**Population**

*Population: googleVis example data set*

**Description**

Example data set to illustrate the use of the googleVis package.

**Usage**

data(Population)

**Format**

A data frame with 195 observations on the following 7 variables.

- **Rank**: a numeric vector with population ranking
- **Country**: country name as character
- **Population**: population
- **% of World Population**: % of world population
- **Flag**: html image-tag to wikipedia with country flag
- **Mode**: logical test vector
- **Date**: date test vector

**Source**

Regions

Examples

data(Population)
tbl <- gvisTable(Population)

## Not run:
plot(tbl)

## End(Not run)

Regions

Regions: googleVis example data set

Description

Example data set to illustrate the use of the googleVis package.

Usage

data(Regions)

Format

A data frame with 11 observations on the following 4 variables.

Region a factor with levels America, Asia ...
Parent parent region identifier
Val a numeric vector
Fac a numeric vector

Examples

data(Regions)
Tree <- gvisTreeMap(Regions, "Region", "Parent", "Val", "Fac",
options=list(width=600, height=500,
showScale=TRUE, fontSize=16))

## Not run:
plot(Tree)

## End(Not run)
renderGvis

Description

This function lets you use googleVis charts as Shiny output. Use it to wrap a googleVis-generating function that you assign to an output element in server.R; then create an htmlOutput with the same name in ui.R.

Usage

renderGvis(expr, env = parent.frame(), quoted = FALSE)

Arguments

expr An expression that returns a gvis object.
env The environment in which to evaluate expr
quoted Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

Details

Use a googleVis Chart as Shiny Output

Value

Returns a function that can be assigned to a Shiny output element.

Details

More information about shiny is available online: https://shiny.rstudio.com/. You find further examples with googleVis on shiny on mages’ blog: https://magesblog.com/tags/shiny/

Author(s)

Joe Cheng, <joe@rstudio.com>

Examples

## Not run:
# To run this example:
shiny::runApp(system.file("shiny/", package="googleVis"))
# server.R
library(googleVis)

shinyServer(function(input, output) {
  datasetInput <- reactive{
    switch(input$dataset,
      "rock" = rock,
      "pressure" = pressure,
      "cars" = cars)
  }
})
Stock: googleVis example data set

Description
Example data set to illustrate the use of the googleVis package.

Usage
data(Stock)

Format
A data frame with 12 observations on the following 5 variables.

Date a Date
Device a character vector
Value a numeric vector
Title a factor with levels Bought pencils Out of stock
Annotation a factor with levels Bought 200k pencils Ran of stock on pens at 4pm

Source
Google Annotated Time Line API: https://developers.google.com/chart/interactive/docs/gallery/annotatedtimeline.html
## Create data as used by Google in their annotated time line example

```r
Date <- as.Date(paste("2008-1- ", 1:6, sep=""))
Pencils <- c(3000, 14045, 5502, 75284, 41476, 333222)
Pencils.titles <- c(rep(NA, 4), "Bought pencils", NA)
Pencils.annotation <- c(rep(NA, 4), "Bought 200k pencils", NA)
Pens <- c(40645, 20374, 50766, 14334, 66467, 39463)
Pens.titles <- c(rep(NA, 3), "Out of stock", NA, NA)
Pens.annotation <- c(rep(NA, 3), "Ran out of stock of pens at 4pm", NA, NA)

original.df <- data.frame(Date, Pencils, Pencils.titles,
                           Pencils.annotation, Pens, Pens.titles,
                           Pens.annotation)

Stock <- reshape(original.df, idvar="Date", times=c("Pencils", "Pens"),
                   timevar="Device",
                   varying=list(c("Pencils", "Pens"),
                                 c("Pencils.titles", "Pens.titles"),
                                 c("Pencils.annotation", "Pens.annotation")),
                   v.names=c("Value", "Title", "Annotation"),
                   direction="long")
```
Index

* **aplot**
  - gvisMerge, 60

* **datasets**
  - Andrew, 3
  - Cairo, 4
  - Cats, 5
  - CityPopularity, 5
  - dino, 7
  - Exports, 7
  - Fruits, 8
  - OpenClose, 91
  - Population, 92
  - Regions, 93
  - Stock, 95

* **intreface**
  - createGoogleGadget, 6

* **iplot**
  - gvis Methods, 8
  - gvisAnnotatedTimeLine, 13
  - gvisAnnotationChart, 17
  - gvisAreaChart, 20
  - gvisBarChart, 23
  - gvisBubbleChart, 26
  - gvisCalendar, 29
  - gvisCandlestickChart, 31
  - gvisColumnChart, 34
  - gvisComboChart, 37
  - gvisGantt, 39
  - gvisGauge, 43
  - gvisGeoChart, 45
  - gvisGeoMap, 49
  - gvisHistogram, 52
  - gvisLineChart, 54
  - gvisMap, 57
  - gvisMotionChart, 62
  - gvisOrgChart, 66
  - gvisPieChart, 69
  - gvisSankey, 71
  - gvisScatterChart, 73
  - gvisSteppedAreaChart, 76
  - gvisTable, 79
  - gvisTimeline, 81
  - gvisTreeMap, 84
  - gvisWordTree, 89

* **methods**
  - gvis Methods, 8

* **package**
  - googleVis-package, 2

* **print**
  - gvis Methods, 8

* **shiny**
  - renderGvis, 94

Andrew, 3
browseURL, 9, 10
Cairo, 4
cat, 6, 9, 10
Cats, 5
class, 14, 18
CityPopularity, 5
class, 15, 19, 22, 24, 27, 30, 33, 35, 38, 41, 44, 47, 50, 53, 55, 58, 60, 64, 68, 70, 72, 74, 77, 80, 82, 86, 90
createGoogleGadget, 6, 10
data.frame, 21, 23, 26, 32, 34, 37, 43, 52, 55, 69, 73, 76, 79
Date, 14, 17, 18, 29, 63
dino, 7
Exports, 7
factor, 14, 18
Fruits, 8
googleVis (googleVis-package), 2
googleVis-package, 2
gvis Methods, 8
gvisAnnotatedTimeLine, 13, 14, 18, 21, 23, 27, 29, 32, 35, 37, 40, 43, 46, 50, 52, 55, 58, 63, 67, 69, 72, 74, 77, 79, 82, 85, 89
gvisAnnotationChart, 14, 16, 17, 18, 21, 23, 27, 29, 32, 35, 37, 40, 43, 46, 50, 52, 55, 58, 63, 67, 69, 72, 74, 77, 79, 82, 85, 89

97
gvisAreaChart, 20
GvisBarChart, 23
GvisBubbleChart, 26
GvisCalendar, 29
GvisCandlestickChart, 31
GvisColumnChart, 34
GvisComboChart, 37
GvisGantt, 39
GvisGauge, 43
GvisGeoChart, 45, 59
GvisGeoMap, 49
GvisHistogram, 52
GvisLineChart, 54
GvisMap, 57
GvisMerge, 10, 60
GvisMotionChart, 28, 62
GvisOrgChart, 66
GvisPieChart, 69
GvisSankey, 71
GvisScatterChart, 73
GvisSteppedAreaChart, 76
GvisTable, 79
GvisTimeline, 81
GvisTreeMap, 84
GvisWordTree, 89

Numeric, 14, 18, 29

OpenClose, 91

Plot.gvis, 16, 19, 22, 25, 28, 31, 34, 36, 38, 44, 47, 54, 56, 59, 61, 65, 68, 70, 75, 78, 81, 87
Plot.gvis(gvis Methods), 8

Population, 92

Print.gvis, 6, 16, 19, 22, 25, 28, 31, 34, 36, 38, 44, 47, 54, 56, 59, 61, 65, 68, 70, 75, 78, 81, 87
Print.gvis(gvis Methods), 8

Regions, 93

RenderGvis, 94

Reshape, 16, 19

Sink, 9

Stock, 95

tempfile, 15, 18, 21, 24, 27, 30, 33, 35, 38, 41, 44, 46, 50, 53, 55, 58, 60, 64, 67, 70, 72, 74, 77, 80, 82, 86, 90