Package ‘googleVis’

August 4, 2012

Type Package

Title Interface between R and the Google Visualisation API

Version 0.2.17

Date 2012-08-02

Author Markus Gesmann, Diego de Castillo

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

Description The googleVis package provides an interface between R and
the Google Visualisation API. With the googleVis package users
can easily create web pages with interactive charts based on R
data frames and display them either via the local R HTTP help
server or within their own sites, without uploading the data to
Google. A browser with Flash and Internet connection is
required. Please visit the project web site for more information and examples.

Depends R (>= 2.11.0), RJSONIO (>= 0.3), methods, utils

Suggests R.rsp, brew, pscl, WDI, Hmisc

License GPL (>= 2)

URL http://code.google.com/p/google-motion-charts-with-r/

LazyLoad yes

LazyData yes

Repository CRAN

Date/Publication 2012-08-04 06:27:39

R topics documented:

`googleVis-package` *Using the Google Visualisation API with R*

Description

The googleVis package provides an interface between R and the Google Visualisation API. Users of the googleVis package can easily create web pages with interactive charts based on R data frames and display them either with the local R HTTP help server or within their own sites, without uploading the data to Google.

A browser with Flash and Internet connection is required. Please visit the project web site for more information and examples: `http://code.google.com/p/google-motion-charts-with-r/`.

Details

Package:	googleVis
Type:	Package
Version:	0.2.17
Date:	2012-08-02
License:	GPL version 2 or later

Note

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

Of course there are many alternative visualisation toolkits out there, e.g.

- d3js: `http://d3js.org`
- Many Eyes: `http://services.alphaworks.ibm.com/manyeyes/page/Create_a_Visualization.html`
- Open Flash Chart (Flash): `http://teethgrinder.co.uk/open-flash-chart/`
- OpenLayers (JavaScript): `http://www.openlayers.org/`
- Processing (Java): `http://processing.org/`
- simile (AJAX): `http://simile.mit.edu/`
- FLARE (ActionScript): `http://flare.prefuse.org/`

See the Google Public Data Explorer for examples using the Google Visualisation API: `http://www.google.com/publicdata/home`

Other R packages of interest to you might be:

- plotGoogleMaps: Plot HTML output with Google Maps API and your own data. `http://cran.r-project.org/web/packages/plotGoogleMaps/`
- RgoogleMaps: Overlays on Google map tiles in R. `http://cran.r-project.org/web/packages/RgoogleMaps/index.html`
- animation: A Gallery of Animations in Statistics and Utilities to Create Animations. `http://cran.r-project.org/web/packages/animation/`
- gridSVG: Export grid graphics as SVG/ `http://cran.r-project.org/web/packages/gridSVG/`
- SVGAnnotation: Tools for Post-Processing SVG Plots Created in R `http://www.omegahat.org/SVGAnnotation/`
- RSVGTipsDevice: An R SVG graphics device with dynamic tips and hyperlinks. `http://cran.r-project.org/web/packages/RSVGTipsDevice/`
- iWebPlots: Interactive web-based plots. `http://cran.r-project.org/web/packages/iWebPlots/`

Author(s)

Markus Gesmann, Diego de Castillo

References

- googleVis project site: http://code.google.com/p/google-motion-charts-with-r/
- Google Chart Tools API: https://developers.google.com/chart/
- Google Terms of Use: https://developers.google.com/terms/
- Google Maps API Terms of Service: https://developers.google.com/maps/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: http://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)
```

`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://google-developers.appspot.com/chart/interactive/docs/gallery/geomap.html

Examples

```
data(CityPopularity)

G <-  gvisGeoMap(CityPopularity, locationvar='City' ,numvar='Popularity',
       options=list(region='US',
     dataMode='markers',
     colors='[0xFF8747, 0xFFB581, 0xc06000]'))
## Not run:
plot(G)

## End(Not run)
```

---

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="http://example.com/gadget.xml" height="200" border="0" />`

You find examples on the googleVis project site: http://code.google.com/p/google-motion-charts-with-r/wiki/GadgetExamples

Author(s)

Markus Gesmann

References

For more information about Google Gadgets see: http://www.google.com/webmasters/gadgets/

See Also

See also as print.gvis, cat

Examples

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

`Exports` *Exports: googleVis example data set*

Description

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

Usage

```
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

```
data(Exports)
Exports
G <- gvisGeoMap(Exports, locationvar='Country',  numvar='Profit',
options=list(height=350, dataMode='regions'))
## Not run:
plot(G)

## End(Not run)
```

`Fruits` *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)
```

`gvis Methods` *Print and plot gvis objects*

Description

Methods to print and plot `gvis` objects

Usage

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

## S3 method for class 'gvis'
plot(x,...)
```

Arguments

`x`	An object of class `gvis`.
`tag`	name tag of the objects to be extracted from a gvis object. The default tag "html" will show a complete web page with the visualisation. The tag "chart" will present all code for the visualisation chart only. For more information see the details section.
`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.
  - `jsChart` Call of the `jsDisplayChart` function.
  - `jsFooter` End tag `</script>`.
  - `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 and R.rsp.

See Also

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

Examples

```
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, "chart")

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

## Not run:
```

```
## 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

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)

## Write the data and functions into separate files:
cat(M$html$chart['jsData'], file=file.path(wwwdir, "gvisData.js"))
cat(M$html$chart[c('jsDrawChart', 'jsDisplayChart', 'jsChart')],
     file=file.path(wwwdir, "gvisFunctions.js"))


## Create a html page with reference to the above
## JavaScript files

html <- sprintf('
<html>
<head>
<script type="text/javascript" src="http://www.google.com/jsapi">
</script>
<script type="text/javascript" src="%s/gvisFunctions.js"></script>
<script type="text/javascript" src="%s/gvisData.js"></script>
<script type="text/javascript">
displayChart%s()
</script>
</head>
<body>
<div id="%s" style="width: 600px; height: 500px;"></div>
</body>
</html>
', baseURL, baseURL, myChartID, myChartID)

## Write html scaffold into a file
cat(html, file=file.path(wwwdir, paste("Chart", myChartID, ".html", sep="")))
```

```
  ## Display the result via
  URL <- paste(baseURL,"/Chart", myChartID, ".html", sep="")
  browseURL(URL)

  ## Update the data, say the data should have shown North and South
  ## instead of East and West as a location
  FruitsUpdate <- Fruits
  levels(FruitsUpdate$Location)=c("North", "South")

  Mupdate=gvisMotionChart(FruitsUpdate, "Fruit", "Year", chartid=myChartID)

  ## Only update the file gvisData.js:
  cat(Mupdate$html$chart['jsData'], file=file.path(wwwdir, "gvisData.js"))

  ## Redisplay the chart with the updated data
  browseURL(URL)


## End(Not run)
```

---

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.

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

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 for Google Annotated Time Line.

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

`gvis.language` values may be 'ca', 'da', 'de', 'en', 'en_GB', 'en_IE', 'es', 'es_419', 'fi', 'fr', 'id', 'in', 'is', 'it', 'nl', 'no', 'pt', 'pt_BR', 'pt_PT', 'sv'. If not set the API detects the language settings of the browser.

Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/annotatedtimeline.html#Configuration_Options`:

`height` height of the chart in pixels.

`width` width of the chart in pixels.

`allowHtml` boolean. Default `FALSE`. If set to `TRUE`, any annotation text that includes HTML tags will be rendered as HTML.

`allowRedraw` boolean. Default `FALSE`. Enables a more efficient redrawing technique for the second and later calls to draw() on this visualization. It only redraws the chart area. To use this option, you must fulfill the following requirements:

- `allowRedraw` must be `TRUE`
- `displayAnnotations` must be `FALSE` (that is, you cannot show annotations)
- you must pass in the same options and values (except for the `allowRedraw` and `displayAnnotations`) as in your first call to draw().

`allValuesSuffix` string. Default none. A suffix to be added to all values in the scales and the legend.

`annotationsWidth` number. Default 25. The width (in percent) of the annotations area, out of the entire chart area. Must be a number in the range 5-80.

`colors` a JSON array of strings. Default colors. The colors to use for the chart lines and labels. An array of strings. Each element is a string in a valid HTML color format. For example 'red' or '#00cc00'.

`dateFormat` string. Either 'MMMM dd, yyyy' or 'HH:mm MMMM dd, yyyy', depending on the type of the first column (date, or datetime, respectively). The format used to display the date information in the top right corner. The format of this field is as specified by the java SimpleDateFormat class.

`displayAnnotations` boolean. Default `FALSE`. If set to `TRUE`, the chart will show annotations on top of selected values. When this option is set to `TRUE`, after every numeric column, two optional annotation string columns can be added, one for the annotation title and one for the annotation text.

`displayAnnotationsFilter` boolean. Default `FALSE`. If set to `TRUE`, the chart will display a filter contol to filter annotations. Use this option when there are many annotations.

`displayDateBarSeparator` boolean. Default `TRUE`. Whether to display a small bar separator (|) between the series values and the date in the legend, where `TRUE` means yes.

`displayExactValues` boolean. Default `FALSE`. Whether to display a shortened, rounded version of the values on the top of the graph, to save space; false indicates that it may. For example, if set to false, 56123.45 might be displayed as 56.12k.

`displayLegendDots` boolean. Default `TRUE`. Whether to display dots next to the values in the legend text, where `TRUE` means yes.

`displayLegendValues` boolean. Default `TRUE`. Whether to display the highlighted values in the legend, where `TRUE` means yes.

`displayRangeSelector` boolean. Default `TRUE`. Whether to show the zoom range selection area (the area at the bottom of the chart), where `FALSE` means no.

The outline in the zoom selector is a log scale version of the last series in the chart, scaled to fit the height of the zoom selector.

`displayZoomButtons` boolean. Default `TRUE`. Whether to show the zoom links ("1d 5d 1m" and so on), where `FALSE` means no.

`fill` number. Default 0. A number from 0-100 (inclusive) specifying the alpha of the fill below each line in the line graph. 100 means 100% opaque fill, 0 means no fill at all. The fill color is the same color as the line above it.

`highlightDot` string. Default 'nearest'. Which dot on the series to highlight, and corresponding values to show in the legend. Select from one of these values:

`'nearest'` The values closest along the X axis to the mouse.

`'last'` The next set of values to the left of the mouse.

`legendPosition` string. Default 'sameRow'. Whether to put the colored legend on the same row with the zoom buttons and the date ('sameRow'), or on a new row ('newRow').

`max` number. Default automatic. The maximum value to show on the Y axis. If the maximum data point exceeds this value, this setting will be ignored, and the chart will be adjusted to show the next major tick mark above the maximum data point. This will take precedence over the Y axis maximum determined by `scaleType`.

`min` number. Default automatic. The minimum value to show on the Y axis. If the minimum data point is less than this value, this setting will be ignored, and the chart will be adjusted to show the next major tick mark below the minimum data point. This will take precedence over the Y axis minimum determined by `scaleType`.

`numberFormats` string, or a map of number:String pairs. Default automatic. Specifies the number format patterns to be used to format the values at the top of the graph.

The patterns should be in the format as specified by the java DecimalFormat class.

- If not specified, the default format pattern is used.
- If a single string pattern is specified, it is used for all of the values.
- If a map is specified, the keys are (zero-based) indexes of series, and the values are the patterns to be used to format the specified series.
  You are not required to include a format for every series on the chart; any unspecified series will use the default format.
  If this option is specified, the `displayExactValues` option is ignored.

`scaleColumns` a JSON array of numbers. Default automatic. Specifies which values to show on the Y axis tick marks in the graph. The default is to have a single scale on the right side, which applies to both series; but you can have different sides of the graph scaled to different series values.
This option takes an array of zero to three numbers specifying the (zero-based) index of the series to use as the scale value. Where these values are shown depends on how many values you include in your array:

- If you specify an empty array, the chart will not show Y values next to the tick marks.
- If you specify one value, the scale of the indicated series will be displayed on the right side of the chart only.
- If you specify two values, a the scale for the second series will be added to the right of the chart.
- If you specify three values, a scale for the third series will be added to the middle of the chart.
- Any values after the third in the array will be ignored.

When displaying more than one scale, it is advisable to set the `scaleType` option to either ’allfixed’ or ’allmaximized’.

`scaleType` string. Default ’fixed’. Sets the maximum and minimum values shown on the Y axis. The following options are available:

`’maximized’` The Y axis will span the minimum to the maximum values of the series. If you have more than one series, use ’allmaximized’.

`’fixed’` **[default**] The Y axis varies, depending on the data values values:

- If all values are >=0, the Y axis will span from zero to the maximum data value.
- If all values are <=0, the Y axis will span from zero to the minimum data value.
- If values are both positive and negative, the Y axis will range from the series maximum to the series minimum.
  For multiple series, use ’allfixed’.

`’allmaximized’` Same as ’maximized,’ but used when multiple scales are displayed. Both charts will be maximized within the same scale, which means that one will be misrepresented against the Y axis, but hovering over each series will display it’s true value.

`’allfixed’` Same as ’fixed,’ but used when multiple scales are displayed. This setting adjusts each scale to the series to which it applies (use this in conjunction with `scaleColumns`).

If you specify the min and/or max options, they will take precedence over the minimum and maximum values determined by your scale type.

`thickness` number. Default 0. A number from 0-10 (inclusive) specifying the thickness of the lines, where 0 is the thinnest.

`wmode` string. Default 'window'. The Window Mode (wmode) parameter for the Flash chart. Available values are: 'opaque', 'window' or 'transparent'.

`zoomEndTime` Date. Default none. Sets the end date/time of the selected zoom range.

`zoomStartTime` Date. Default none. Sets the start date/time of the selected zoom range.

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

Details

From `https://google-developers.appspot.com/chart/interactive/docs/gallery/annotatedtimeline.html#Data_Format`:

You can display one or more lines on your chart. Each row represents an X position on the chart - that is, a specific time; each line is described by a set of one to three columns.

- The first column is of type date or datetime, and specifies the X value of the point on the chart. If this column is of type date (and not datetime) then the smallest time resolution on the X axis will be one day.
- Each data line is then described by a set of one to three additional columns as described here:
  - Y value - [Required, Number] The first column in each set describes the value of the line at the corresponding time from the first column. The column label is displayed on the chart as the title of that line.
  - Annotation title - [Optional, String] If a string column follows the value column, and the displayAnnotations option is true, this column holds a short title describing this point. For instance, if this line represents temperature in Brazil, and this point is a very high number, the title could be "Hottest day on record".
  - Annotation text - [Optional string] If a second string column exists for this series, the cell value will be used as additional descriptive text for this point. You must set the option displayAnnotations to true to use this column. You can use HTML tags, if you set allowHtml to true; there is essentially no size limit, but note that excessively long entries might overflow the display section. You are not required to have this column even if you have an annotation title column for this point. The column label is not used by the chart. For example, if this were the hottest day on record point, you might say something like "Next closest day was 10 degrees cooler!".

Value

`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, here 'AnnotatedTimeLine'

`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.
- `jsChart` Call of the `jsDisplayChart` function.
- `jsFooter` End tag `</script>`.
- `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. http://www.myhost.com/myviz.html). See the googleVis package vignette and the Macromedia web site (`http://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 Annotated Time Line API: `https://google-developers.appspot.com/chart/interactive/docs/gallery/annotatedtimeline.html`

Follow the link for Google's data policy.

See Also

See also `print.gvis`, `plot.gvis` for printing and plotting methods. Further see `reshape` for reshaping 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=600, height=350)
                           )
plot(A1)

## Two Y-axis
A2 <- gvisAnnotatedTimeLine(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)

## Zoom into the time window, no Y-axis ticks
A3 <- gvisAnnotatedTimeLine(Stock, datevar="Date",
                           numvar="Value", idvar="Device",
                           titlevar="Title", annotationvar="Annotation",
                           options=list(
                             width=600, height=350,
                            zoomStartTime=as.Date("2008-01-04"),
                            zoomEndTime=as.Date("2008-01-05"))
                          )
plot(A3)



## Colouring the area below the lines to create an area chart
A4 <- gvisAnnotatedTimeLine(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(A4)
```

```
## Data with POSIXct datetime variable
A5 <- gvisAnnotatedTimeLine(Andrew, datevar="Date/Time UTC",
                            numvar="Pressure_mb",
                            options=list(scaleType='maximized')
                            )

plot(A5)


## Not run:

## Plot Apple's monthly stock prices since 1984

## Get current date
d <- Sys.time()
current.year <- format(d, "%Y")
current.month <- format(d, "%m")
current.day <- format(d, "%d")

## Yahoo finance sets January to 00 hence:
month <- as.numeric(current.month)  - 1
month <- ifelse(month < 10, paste("0",month, sep=""), m)

## Get weekly stock prices from Apple Inc.
tckr <- 'AAPL'
fn <- sprintf('http://ichart.finance.yahoo.com/table.csv?s=%s&a=08&b=7&c=1984&d=%s&e=%s&f=%s&g=w&ignore=.csv',
      tckr, month, current.day, current.year)

## Get data from Yahoo! Finance
data <- read.csv(fn, colClasses=c("Date", rep("numeric",6)))

AAPL <- reshape(data[,c("Date", "Close", "Volume")], idvar="Date",
      times=c("Close", "Volume"),
                timevar="Type",
                varying=list(c("Close", "Volume")),
                v.names="Value",
                direction="long")

## Calculate previous two years for zoom start time
lyd <- as.POSIXlt(as.Date(d))
lyd$year <- lyd$year-2
lyd <- as.Date(lyd)

aapl <- gvisAnnotatedTimeLine(AAPL, datevar="Date",
                           numvar="Value", idvar="Type",
                          options=list(
                            colors="['blue', 'lightblue']",
                            zoomStartTime=lyd,
                            zoomEndTime=as.Date(d),
                            legendPosition='newRow',
                            width=600, height=400, scaleColumns='[0,1]',
                            scaleType='allmaximized')
```

```
                                )

plot(aapl)

## End(Not run)
```

`gvisAreaChart` *Google Area Chart with R*

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.

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

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 for Google Area Chart.

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

Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/areachart.html#Configuration_Options`:

`areaOpacity` a number between 0.0 - 1.0. Default 0.3. The default opacity of the colored area under an area chart series, where 0.0 is fully transparent and 1.0 is fully opaque. To specify opacity for an individual series, set the areaOpacity value in the series property.

`axisTitlesPosition` a string. Default 'out'. Where to place the axis titles, compared to the chart area. Supported values:

`'in'` Draw the axis titles inside the the chart area.

`'out'` Draw the axis titles outside the chart area.

`'none'` Omit the axis titles.

`backgroundColor` a string or object. Default `'white'`. The background color for the main area of the chart. Can be either a simple HTML color string, for example: `'red'` or `'#00cc00'`, or an object with the following properties.

`backgroundColor.stroke` a string. Default `'#666'`. The color of the chart border, as an HTML color string.

`backgroundColor.strokeWidth` a number. Default 0. The border width, in pixels.

`backgroundColor.fill` a string. Default `'white'`. The chart fill color, as an HTML color string.

`chartArea` a string. Default `'null'`. An object with members to configure the placement and size of the chart area (where the chart itself is drawn, excluding axis and legends). Two formats are supported: a number, or a number followed by `%`. A simple number is a value in pixels; a number followed by `%` is a percentage. Example:

`{left:20,top:0,width:\"50%\",height:\"75%\"}`

`chartArea.height` a number or string. Default auto. Chart area height.

`chartArea.left` a number or string. Default auto. How far to draw the chart from the left border.

`chartArea.top` a number or string. Default auto. How far to draw the chart from the top border.

`chartArea.width` a number or string. Default auto. Chart area width.

`colors` a JSON array of strings. Default 'colors'. The colors to use for the chart elements. An array of strings, where each element is an HTML color string, for example: `colors:[red','#004411']`.

`enableInteractivity` boolean. Default `TRUE`. Whether the chart throws user-based events or reacts to user interaction. If false, the chart will not throw 'select' or other interaction-based events (but will throw ready or error events), and will not display hovertext or otherwise change depending on user input.

`focusTarget` a string. Default `'datum'`. The type of the entity that receives focus on mouse hover. Also affects which entity is selected by mouse click, and which data table element is associated with events. Can be one of the following:

`'datum'` Focus on a single data point. Correlates to a cell in the data.

`'category'` Focus on a grouping of all data points along the major axis. Correlates to a row in the data table.

In focusTarget `'category'` the tooltip displays all the category values. This may be useful for comparing values of different series.

`fontSize` a number. Default automatic. The default font size, in pixels, of all text in the chart. You can override this using properties for specific chart elements.

`fontName` a string. Default `'Arial'`. The default font face for all text in the chart. You can override this using properties for specific chart elements.

`hAxis` a JSON object. Default `'null'`. An object with members to configure various horizontal axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`hAxis.baseline` a number. Default automatic. The baseline for the horizontal axis. This option is only supported for a continuous axis.

`hAxis.baselineColor` a string. Default `'black'`. The color of the baseline for the horizontal axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`. This option is only supported for a continuous axis.

`hAxis.direction` 1 or -1. Default 1. The direction in which the values along the vertical axis grow. Specify -1 to reverse the order of the values.

`hAxis.format` a string. Default auto. A format string for numeric or date axis labels.

For number axis labels, this is a subset of the decimal formatting ICU pattern set. For instance,

`{format:'#,###%'}.`

will display values `"1,000%"`, `"750%"`, and `"50%"` for values 10, 7.5, and 0.5.

For date axis labels, this is a subset of the date formatting ICU pattern set. For instance,

`{format:'MMM d, y'}.`

will display the value "Jul 1, 2011" for the date of July first in 2011.

The actual formatting applied to the label is derived from the locale the API has been loaded with. For more details, see loading charts with a specific locale.

This option is only supported for a continuous axis.

`hAxis.gridlines` a JSON object. Default null. An object with members to configure the gridlines on the horizontal axis. To specify properties of this object, you can use object literal notation, as shown here:

`{color: '#333', count: 4}`

This option is only supported for a continuous axis.

`hAxis.gridlines.color` a string. Default `'#CCC'`. The color of the vertical gridlines inside the chart area. Specify a valid HTML color string.

`hAxis.gridlines.count` a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

`hAxis.logScale` boolean. Default `FALSE`. `vAxis` property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to `TRUE` for yes. This option is only supported for a continuous axis.

`hAxis.textPosition` a string. Default `'out'` Position of the horizontal axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`hAxis.textStyle` a JSON object. Default

```
{color: 'black',
fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the horizontal axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.title` a string. Default `'null'`. `hAxis` property that specifies the title of the horizontal axis.

`hAxis.titleTextStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}.`
An object that specifies the horizontal axis title text style. The object has this format:
`{color: <string>, fontName: <string>, fontSize: <number>}`
The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.slantedText` boolean. Default automatic. If true, draw the horizontal axis text at an angle, to help fit more text along the axis; if false, draw horizontal axis text upright. Default behavior is to slant text if it cannot all fit when drawn upright. Notice that this option is available only when the `hAxis.textPosition` is set to `'out'` (which is the default).
This option is only supported for a discrete axis.

`hAxis.slantedTextAngle` a number, 1-90. Default 30. The angle of the horizontal axis text, if it's drawn slanted. Ignored if `hAxis.slantedText` is false, or is in auto mode, and the chart decided to draw the text horizontally.
This option is only supported for a discrete axis.

`hAxis.maxAlternation` a number. Default 2. Maximum number of levels of horizontal axis text. If axis text labels become too crowded, the server might shift neighboring labels up or down in order to fit labels closer together. This value specifies the most number of levels to use; the server can use fewer levels, if labels can fit without overlapping.
This option is only supported for a discrete axis.

`hAxis.showTextEvery` a number. Default automatic. How many horizontal axis labels to show, where 1 means show every label, 2 means show every other label, and so on. Default is to try to show as many labels as possible without overlapping.
This option is only supported for a discrete axis.

`hAxis.maxValue` a number. Default automatic. `hAxis` property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.
This option is only supported for a continuous axis.

`hAxis.minValue` a number. Default automatic. `hAxis` property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.
This option is only supported for a continuous axis.

`hAxis.viewWindowMode` a string. Default `"pretty"` if `hAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the horizontal axis to render the values within the chart area. The following string values are supported:

`'pretty'` Scale the horizontal values so that the maximum and minimum data values are rendered a bit inside the left and right of the chart area.

`'maximized'` Scale the horizontal values so that the maximum and minimum data values touch the left and right of the chart area.

'explicit' Specify the left and right scale values of the chart area. Data values outside these values will be cropped. You must specify a

hAxis.viewWindow

object describing the maximum and minimum values to show.

This option is only supported for a continuous axis.

hAxis.viewWindow Object. Default NULL. Specifies the cropping range of the horizontal axis.

hAxis.viewWindow.max A number. Default auto.

For a continuous axis The maximum horizontal data value to render. Has an effect only if hAxis.viewWindowMode='explicit'.

For a discrete axis The zero-based row index where the cropping window ends. Data points at this index and higher will be cropped out. In conjunction with vAxis.viewWindowMode.min, it defines a half-opened range [min, max) that denotes the element indices to display. In other words, every index such that min <= index < max will be displayed.

hAxis.viewWindow.min A number. Default auto.

For a continuous axis The minimum horizontal data value to render. Has an effect only if hAxis.viewWindowMode='explicit'.

For a discrete axis The zero-based row index where the cropping window begins. Data points at indices lower than this will be cropped out. In conjunction with vAxis.viewWindowMode.max, it defines a half-opened range [min, max) that denotes the element indices to display. In other words, every index such that min <= index < max will be displayed.

height a number. Default height of the containing element. Height of the chart, in pixels.

isStacked boolean. Default FALSE. If set to TRUE, bar values are stacked (accumulated).

legend a JSON object. Default NULL. An object with members to configure various aspects of the legend. To specify properties of this object, you can use object literal notation, as shown here:

```
{position: 'top', textStyle: {color: 'blue', fontSize: 16}}
```

legend.position a string. Default 'right'. Position of the legend. Can be one of the following:

'right' To the right of the chart.

'top' Above the chart.

'bottom' Below the chart.

'none' No legend is displayed.

legend.textStyle a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the legend text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`lineWidth` a number. Default 2. Line width in pixels. Use zero to hide all lines and show only the points.

`pointSize` a number. Default 0. Diameter of data points, in pixels. Use zero to hide all points.

`reverseCategories` boolean. Default `FALSE`. If set to true, will draw series from right to left. The default is to draw left-to-right.
This option is only supported for a discrete major axis.

`series` a JSON array of objects, or object with nested objects. Default `{}`.

An array of objects, each describing the format of the corresponding series in the chart. To use default values for a series, specify an empty object . If a series or a value is not specified, the global value will be used. Each object supports the following properties:

`color` The color to use for this series. Specify a valid HTML color string.

`targetAxisIndex` Which axis to assign this series to, where 0 is the default axis, and 1 is the opposite axis. Default value is 0; set to 1 to define a chart where different series are rendered against different axes. You can define a different scale for different axes.

`pointSize` Overrides the global pointSize value for this series.

`lineWidth` Overrides the global lineWidth value for this series.

`curveType` Overrides the global curveType value for this series.

`visibleInLegend` A boolean value, where true means that the series should have a legend entry, and false means that it should not. Default is `TRUE`.

You can specify either an array of objects, each of which applies to the series in the order given, or you can specify an object where each child has a numeric key indicating which series it applies to. For example, the following two declarations are identical, and declare the first series as black and absent from the legend, and the fourth as red and absent from the legend:

```
series: [{color: 'black', visibleInLegend: false},{},
{}, {color: 'red', visibleInLegend: false}]
```

```
series: {0:{color: 'black', visibleInLegend: false},
3:{color: 'red', visibleInLegend: false}}
```

`theme` a string. Default `NULL`. A theme is a set of predefined option values that work together to achieve a specific chart behavior or visual effect. Currently only one theme is available:

`maximized` Maximizes the area of the chart, and draws the legend and all of the labels inside the chart area. Sets the following options:

```
chartArea: {width: '100%', height: '100%'},
legend: {position: 'in'},
titlePosition: 'in', axisTitlesPosition: 'in',
hAxis: {textPosition: 'in'}, vAxis: {textPosition: 'in'}
```

`title` a string. Default no title. Text to display above the chart.

`titlePosition` a string. Default `'out'`. Where to place the chart title, compared to the chart area. Supported values:

`'in'` Draw the title inside the chart area.

`'out'` Draw the title outside the chart area.

`'none'` Omit the title.

`titleTextStyle` a JSON object. Default

```
{color:'black',
  fontName:<global-font-name>,fontSize:<global-font-size>}.
```

An object that specifies the title text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip` a JSON object. Default `NULL`. An object with members to configure various tooltip elements. To specify properties of this object, you can use object literal notation, as shown here:

```
{textStyle: {color: '#FF0000'}, showColorCode: true}
```

`tooltip.showColorCode` boolean. Default automatic. If true, show colored squares next to the series information in the tooltip. The default is true when `focusTarget` is set to `'category'`, otherwise the default is `FALSE`.

`tooltip.TextStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the tooltip text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize: <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip.trigger` The user interaction that causes the tooltip to be displayed:

`'hover'` The tooltip will be displayed when the user hovers over an element.

`'none'` The tooltip will not be displayed.

`vAxes` a JSON array of objects, or object with child objects null. Specifies properties for individual vertical axes, if the chart has multiple vertical axes. Each child object is a `vAxis` object, and can contain all the properties supported by `vAxes`. These property values override any global settings for the same property.
To specify a chart with multiple vertical axes, first define a new axis using `series.targetAxisIndex`, then configure the axis using `vAxes`. The following example assigns series 2 to the right axis and specifies a custom title and text style for it:

```
series:{2:{targetAxisIndex:1}},
  vAxes:{1:{title:'Losses',textStyle:{color: 'red'}}}
```

This property can be either an object or an array: the object is a collection of objects, each with a numeric label that specifies the axis that it defines–this is the format shown above; the array is an array of objects, one per axis. For example, the following array-style notation is identical to the `vAxis` object shown above:

```
vAxes:[{}, // Nothing specified for axis 0
{title:'Losses',textStyle:{color: 'red'}} // Axis 1
]
```

`vAxis` a JSON object. Default `'null'`. An object with members to configure various vertical axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`vAxis.baseline` a number. Default automatic. `vAxis` property that specifies the baseline for the vertical axis. If the baseline is smaller than the highest grid line or smaller than the lowest grid line, it will be rounded tothe closest gridline.

`vAxis.baselineColor` a string. Default `'black'`. `vAxis` property that specifies the color of the baseline for the vertical axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`.

`vAxis.direction` 1 or -1. Default 1. The direction in which the values along the vertical axis grow. Specify -1 to reverse the order of the values.

`vAxis.format` a string. Default auto. A format string for numeric axis labels. This is a subset of the ICU pattern set. For instance,

`{format:'#,###%'}`.

will display values 1,000%, 750%, and 50% for values 10, 7.5, and 0.5.

`vAxis.gridlines` a JSON object. Default `NULL`. An object with members to configure the gridlines on the vertical axis. To specify properties of this object, you can use object literal notation, as shown here:

`{color: '#333', count: 4}`

`vAxis.gridlines.color` a string. Default `'#CCC'`. The color of the vertical gridlines inside the chart area. Specify a valid HTML color string.

`vAxis.gridlines.count` a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

`vAxis.logScale` boolean. Default `FALSE`. `vAxis` property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to `TRUE` for yes.

`vAxis.textPosition` a string. Default `'out'`. Position of the vertical axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`vAxis.textStyle` a JSON object. Default

`{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`.

An object that specifies the vertical axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.title` a string. Default no title. `vAxis` property that specifies a title for the vertical axis.

`vAxis.titleTextStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}.
```

An object that specifies the vertical axis title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.maxValue` a number. Default automatic. vAxis property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.

`vAxis.minValue` a number. Default automatic. `vAxis` property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.

`vAxis.viewWindowMode` a string. Default `"pretty"` if `vAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the vertical axis to render the values within the chart area. The following string values are supported:

- `'pretty'` Scale the vertical values so that the maximum and minimum data values are rendered a bit inside the top and bottom of the chart area.
- `'maximized'` Scale the vertical values so that the maximum and minimum data values touch the top and bottom of the chart area.
- `'explicit'` Specify the top and bottom scale values of the chart area. Data values outside these values will be cropped. You must specify a `vAxis.viewWindow` object describing the maximum and minimum values to show.

`vAxis.viewWindow` a JSON object. Specifies the cropping range of the vertical axis.

`vAxis.viewWindow.max` A number. Default 0. The maximum vertical data value to render.
Has an effect only if vAxis.viewWindowMode='explicit'.

`vAxis.viewWindow.min` A number. Default 0. The minimum vertical data value to render.
Has an effect only if vAxis.viewWindowMode='explicit'.

`width` a number. Default width of the containing element. Width of the chart, in pixels.

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

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, here 'AreaChart'
`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.
  - `jsChart` Call of the `jsDisplayChart` function.
  - `jsFooter` End tag `</script>`.
  - `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

Google Visualisation API: You cannot load both areachart and corechart packages at the same time on the same page.

Author(s)

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

Diego de Castillo `<decastillo@gmail.com>`

References

Google Area Chart API: `http://code.google.com/apis/chart/interactive/docs/gallery/areachart.html`

Follow the link for Google's data policy.

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))

## 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))
plot(Area2)


## 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)

## Not run:
## 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 Bar Chart with R*

Description

The gvisBarChart 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

```
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 for Google Bar Chart.

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

Further possible components are, taken from https://google-developers.appspot.com/chart/interactive/docs/gallery/barchart.html#Configuration_Options:

`axisTitlesPosition` a string. Default 'out'. Where to place the axis titles, compared to the chart area. Supported values:

`'in'` Draw the axis titles inside the the chart area.

`'out'` Draw the axis titles outside the chart area.

`'none'` Omit the axis titles.

`backgroundColor` a string or object. Default `'white'`. The background color for the main area of the chart. Can be either a simple HTML color string, for example: `'red'` or `'#00cc00'`, or an object with the following properties.

`backgroundColor.stroke` a string. Default `'#666'`. The color of the chart border, as an HTML color string.

`backgroundColor.strokeWidth` a number. Default 0. The border width, in pixels.

`backgroundColor.fill` a string. Default `'white'`. The chart fill color, as an HTML color string.

`chartArea` a string. Default `'null'`. An object with members to configure the placement and size of the chart area (where the chart itself is drawn, excluding axis and legends). Two formats are supported: a number, or a number followed by `%`. A simple number is a value in pixels; a number followed by `%` is a percentage. Example:

`{left:20,top:0,width:\"50%\",height:\"75%\"}`

`chartArea.height` a number or string. Default auto. Chart area height.

`chartArea.left` a number or string. Default auto. How far to draw the chart from the left border.

`chartArea.top` a number or string. Default auto. How far to draw the chart from the top border.

`chartArea.width` a number or string. Default auto. Chart area width.

`colors` A JSON array of strings. Default 'colors'. The colors to use for the chart elements. An array of strings, where each element is an HTML color string, for example: `colors:['red','#004411']`.

`enableInteractivity` boolean. Default `TRUE`. Whether the chart throws user-based events or reacts to user interaction. If false, the chart will not throw 'select' or other interaction-based events (but will throw ready or error events), and will not display hovertext or otherwise change depending on user input.

`focusTarget` a string. Default `'datum'`. The type of the entity that receives focus on mouse hover. Also affects which entity is selected by mouse click, and which data table element is associated with events. Can be one of the following:

`'datum'` Focus on a single data point. Correlates to a cell in the data.

`'category'` Focus on a grouping of all data points along the major axis. Correlates to a row in the data table.

In focusTarget `'category'` the tooltip displays all the category values. This may be useful for comparing values of different series.

`fontSize` a number. Default automatic. The default font size, in pixels, of all text in the chart. You can override this using properties for specific chart elements.

`fontName` a string. Default `'Arial'`. The default font face for all text in the chart. You can override this using properties for specific chart elements.

`hAxes` a JSON array of objects, or object with child objects null. Specifies properties for individual horizontal axes, if the chart has multiple horizontal axes. Each child object is a `hAxis` object, and can contain all the properties supported by `hAxis`. These property values override any global settings for the same property.
To specify a chart with multiple horizontal axes, first define a new axis using `series.targetAxisIndex`, then configure the axis using `vAxes`. The following example assigns series 2 to the bottom axis and specifies a custom title and text style for it:

```
series:{2:{targetAxisIndex:1}},
  vAxes:{1:{title:'Losses',textStyle:{color: 'red'}}}
```

This property can be either an object or an array: the object is a collection of objects, each with a numeric label that specifies the axis that it defines–this is the format shown above; the array is an array of objects, one per axis. For example, the following array-style notation is identical to the `hAxis` object shown above:

```
hAxes:[
{}, // Nothing specified for axis 0
{title:'Losses',textStyle:{color: 'red'}} // Axis 1
]
```

`hAxis` a JSON object. Default `'null'`. An object with members to configure various horizontal axis elements. To specify properties of this object, you can use object literal notation, as shown here:

```
{title: 'Hello', titleTextStyle: {color: '#FF0000'}}
```

`hAxis.baseline` a number. Default automatic. `hAxis` property that specifies the baseline for the horizontal axis. If the baseline is smaller than the highest grid line or smaller than the lowest grid line, it will be rounded to the closest gridline.

`hAxis.baselineColor` a string. Default 'black'. `hAxis` property that specifies the color of the baseline for the horizontal axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`.

`hAxis.direction` 1 or -1. Default 1. The direction in which the values along the horizontal axis grow. Specify -1 to reverse the order of the values.

`hAxis.format` a string. Default auto. A format string for numeric axis labels. This is a subset of the ICU pattern set. For instance,

`{format:'#,###%'}`.

will display values 1,000%, 750%, and 50% for values 10, 7.5, and 0.5.

`hAxis.gridlines` a JSON object. Default `NULL`. An object with members to configure the gridlines on the vertical axis. To specify properties of this object, you can use object literal notation, as shown here:

```
{color: '#333', count: 4}
```

`hAxis.gridlines.color` a string. Default `'#CCC'`. The color of the horizontal gridlines inside the chart area. Specify a valid HTML color string.

`hAxis.gridlines.count` a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

`hAxis.logScale` boolean. Default `FALSE`. `vAxis` property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to `TRUE` for yes.

`hAxis.textPosition` a string. Default `'out'` Position of the horizontal axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`hAxis.textStyle` a JSON object. Default

```
{color: 'black',
fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the horizontal axis text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize: <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.title` a string. Default `'null'`. `hAxis` property that specifies the title of the horizontal axis.

`hAxis.titleTextStyle` a JSON object. Default

```
{color: 'black',
fontName: <global-font-name>, fontSize: <global-font-size>}.
```

An object that specifies the horizontal axis title text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize: <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.maxValue` a number. Default automatic. `hAxis` property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.

`hAxis.minValue` a number. Default automatic. `hAxis` property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.

`hAxis.viewWindowMode` a string. Default `"pretty"` if `hAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the horizontal axis to render the values within the chart area. The following string values are supported:

`'pretty'` Scale the horizontal values so that the maximum and minimum data values are rendered a bit inside the left and right of the chart area.

`'maximized'` Scale the horizontal values so that the maximum and minimum data values touch the left and right of the chart area.

`'explicit'` Specify the left and right scale values of the chart area. Data values outside these values will be cropped. You must specify a

`hAxis.viewWindow`

object describing the maximum and minimum values to show.

`hAxis.viewWindow` JSON object. Default `NULL`. Specifies the maximum and minimum data values to show on the horizontal axis. Present only if

`vAxis.viewWindowMode = 'explicit'`

`hAxis.viewWindow.max` number. Default 0. The maximum vertical data value to render.

`hAxis.viewWindow.min` number. Default 0. The minimum vertical data value to render.

`height` number. Default height of the containing element. Height of the chart, in pixels.

`isStacked` boolean. Default `FALSE`. If set to `TRUE`, bar values are stacked (accumulated).

`legend` a JSON object. Default `NULL`. An object with members to configure various aspects of the legend. To specify properties of this object, you can use object literal notation, as shown here:

`{position: 'top', textStyle: {color: 'blue', fontSize: 16}}`

`legend.position` a string. Default `'right'`. Position of the legend. Can be one of the following:

`'right'` To the right of the chart.

`'top'` Above the chart.

`'bottom'` Below the chart.

`'none'` No legend is displayed.

`legend.textStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the legend text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`reverseCategories` boolean. Default `FALSE`. If set to true, will draw series from right to left. The default is to draw left-to-right.

`series` a JSON array of objects, or object with nested objects. Default {}. An array of objects, each describing the format of the corresponding series in the chart. To use default values for a series, specify an empty object . If a series or a value is not specified, the global value will be used. Each object supports the following properties:

`color` The color to use for this series. Specify a valid HTML color string.

`targetAxisIndex` Which axis to assign this series to, where 0 is the default axis, and 1 is the opposite axis. Default value is 0; set to 1 to define a chart where different series are rendered against different axes. You can define a different scale for different axes.

`pointSize` Overrides the global pointSize value for this series.

`lineWidth` Overrides the global lineWidth value for this series.

`curveType` Overrides the global curveType value for this series.

`visibleInLegend` A boolean value, where true means that the series should have a legend entry, and false means that it should not. Default is `TRUE`.

You can specify either an array of objects, each of which applies to the series in the order given, or you can specify an object where each child has a numeric key indicating which series it applies to. For example, the following two declarations are identical, and declare the first series as black and absent from the legend, and the fourth as red and absent from the legend:

```
series: [{color: 'black', visibleInLegend: false},{},
{}, {color: 'red', visibleInLegend: false}]

series: {0:{color: 'black', visibleInLegend: false},
3:{color: 'red', visibleInLegend: false}}
```

`theme` a string. Default `NULL`. A theme is a set of predefined option values that work together to achieve a specific chart behavior or visual effect. Currently only one theme is available:

`maximized` Maximizes the area of the chart, and draws the legend and all of the labels inside the chart area. Sets the following options:

```
chartArea: {width: '100%', height: '100%'},
legend: {position: 'in'},
titlePosition: 'in', axisTitlesPosition: 'in',
hAxis: {textPosition: 'in'}, vAxis: {textPosition: 'in'}
```

`title` a string. Default no title. Text to display above the chart.

`titlePosition` a string. Default `'out'`. Where to place the chart title, compared to the chart area. Supported values:

`'in'` Draw the title inside the chart area.

`'out'` Draw the title outside the chart area.

`'none'` Omit the title.

`titleTextStyle` a JSON object. Default

```
{color:'black',
  fontName:<global-font-name>,fontSize:<global-font-size>}.
```

An object that specifies the title text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip` a JSON object. Default `NULL`. An object with members to configure various tooltip elements. To specify properties of this object, you can use object literal notation, as shown here:

`{textStyle: {color: '#FF0000'}, showColorCode: true}`

`tooltip.showColorCode` boolean. Default automatic. If true, show colored squares next to the series information in the tooltip. The default is true when `focusTarget` is set to `'category'`, otherwise the default is `FALSE`.

`tooltip.TextStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the tooltip text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip.trigger` The user interaction that causes the tooltip to be displayed:

`'hover'` The tooltip will be displayed when the user hovers over an element.

`'none'` The tooltip will not be displayed.

`vAxis` a JSON object. Default `'null'`. An object with members to configure various vertical axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`vAxis.direction` 1 or -1. Default 1. The direction in which the values along the vertical axis grow. Specify -1 to reverse the order of the values.

`vAxis.textPosition` a string. Default `'out'`. Position of the vertical axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`vAxis.textStyle` a JSON object. Default

```
{color: 'black', fontName: <global-font-name>,
  fontSize: <global-font-size>}.
```

An object that specifies the vertical axis text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.title` a string. Default no title. `vAxis` property that specifies a title for the vertical axis.

`vAxis.titleTextStyle` a JSON object. Default

```
{color: 'black',
fontName: <global-font-name>, fontSize: <global-font-size>}.
```

An object that specifies the vertical axis title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

	The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.
	`width` a number. Default width of the containing element. Width of the chart, in pixels.
`chartid`	character. If missing (default) a random chart id will be generated based on chart type and `tempfile`

Value

`gvisBarChart` returns a 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 'BarChart'
`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.
  - `jsChart` Call of the `jsDisplayChart` function.
  - `jsFooter` End tag `</script>`.
  - `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

Google Visualisation API: You cannot load both barchart/columnchart and corechart packages at the same time on the same page.

Author(s)

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

Diego de Castillo <`decastillo@gmail.com`>

References

Google Bar Chart API: `http://code.google.com/apis/chart/interactive/docs/gallery/barchart.html`

Follow the link for Google's data policy.

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 smoothed curve
Bar3 <- gvisBarChart(df, xvar="country", yvar=c("val1", "val2"),
             options=list(title="Hello World",
                          titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}",
                          curveType='function'))
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 readHMLTable 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("http://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:
df[,2]=as.numeric(gsub(",", "", as.character(df[,2])))

## Finally we can create a nice bar chart:
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.

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.

Usage

```
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.
`options`	list of configuration options for Google Bubble Chart. `gvis.editor` a character label for an on-page button which opens an in-page dialog box that enables users to edit, change and customise the chart. By default no value is given and therefore no button is displayed. Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/bubblechart.html#Configuration_Options`:

`axisTitlesPosition` a string. Default 'out'. Where to place the axis titles, compared to the chart area. Supported values:

  `'in'` Draw the axis titles inside the the chart area.

  `'out'` Draw the axis titles outside the chart area.

  `'none'` Omit the axis titles.

`backgroundColor` a string or object. Default `'white'`. The background color for the main area of the chart. Can be either a simple HTML color string, for example: `'red'` or `'#00cc00'`, or an object with the following properties.

`backgroundColor.stroke` a string. Default `'#666'`. The color of the chart border, as an HTML color string.

`backgroundColor.strokeWidth` a number. Default 0. The border width, in pixels.

`backgroundColor.fill` a string. Default `'white'`. The chart fill color, as an HTML color string.

`bubble` a JSON object. Default `NULL`. An object with members to configure the visual properties of the bubbles.

`bubble.opacity` a number between 0.0 - 1.0. Default 0.8. The opacity of the bubbles, where 0 is fully transparent and 1 is fully opaque.

`bubble.stroke` a string. Default `'#ccc'`. The color of the bubbles' stroke.

`bubble.textStyle` a JSON object. Default

```
{color: 'black', fontName: <global-font-name>, fontSize:
  <global-font-size>}.
```

  An object that specifies the bubble text style. The object has this format:

```
  {color: <string>, fontName: <string>, fontSize: <number>}
```

  The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`chartArea` a string. Default `'null'`. An object with members to configure the placement and size of the chart area (where the chart itself is drawn, excluding axis and legends). Two formats are supported: a number, or a number followed by `%`. A simple number is a value in pixels; a number followed by `%` is a percentage. Example:

```
{left:20,top:0,width:\"50%\",height:\"75%\"}
```

`chartArea.height` a number or string. Default auto. Chart area height.

`chartArea.left` a number or string. Default auto. How far to draw the chart from the left border.

`chartArea.top` a number or string. Default auto. How far to draw the chart from the top border.

`chartArea.width` a number or string. Default auto. Chart area width.

`colors` a JSON array of strings. Default 'colors'. The colors to use for the chart elements. An array of strings, where each element is an HTML color string, for example: `colors:[red','#004411']`.

`colorAxis` a string. Default `'null'` An object that specifies a mapping between colors and color column values. To specify properties of this object, you can use object literal notation, as shown here:

`{minValue: 0,  colors: ['#FF0000', '#00FF00']}`

`colorAxis.minValue` a number. Default minimum value of color column in chart data. If present, specifies a minimum value for chart color data. Color data values of this value and lower will be rendered as the first color in the `colorAxis.colors` range.

`colorAxis.maxValue` a number. Default maximum value of color column in chart data If present, specifies a maximum value for chart color data. Color data values of this value and higher will be rendered as the last color in the `colorAxis.colors` range.

`colorAxis.values` a JSON array of numbers. Default `'null'`. Controls how values are associated with colors. Each value is associated with the corresponding color in the `colorAxis.colors` array. These values apply to the color value for a region or marker. Regions are colored according to a gradient of the values specified here. Not specifying a value for this option is equivalent to specifying `[minValue,  maxValue]`.

`colorAxis.colors` a JSON array of color strings. Default `'null'`. Colors to assign to values in the visualization. An array of strings, where each element is an HTML color string, for example: colorAxis:

`{colors:['red','#004411']}.`

You must have at least two values; the gradient will include all your values, plus calculated intermediary values, with the first color as the smallest value, and the last color as the highest.

`colorAxis.legend` an object. Default `null`. An object that specifies the style of the gradient color legend.

`colorAxis.legend.position` a string. Default `'top'`. Position of the legend. Can be one of the following:

`'top'` Above the chart.

`'bottom'` Below the chart.

`'in'` Inside the chart, at the top.

`'none'` No legend is displayed.

`colorAxis.legend.textStyle` an object. Default

`{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`

An object that specifies the legend text style. The object has this format:

```
  {color: <string>, fontName: <string>, fontSize: <number>}
```

The color can be any HTML color string, for example:

`'red'`

or

`'#00cc00'`

. Also see `fontName` and `fontSize`.

`colorAxis.legend.numberFormat` a string. Default `'auto'`. A format string for numeric labels. This is a subset of the ICU pattern set. For instance,

`{numberFormat:'.##'}`

will display values

`"10.66", "10.6"`

, and

`"10.0"`

for values 10.666, 10.6, and 10.

`enableInteractivity` boolean. Default `TRUE`. Whether the chart throws user-based events or reacts to user interaction. If false, the chart will not throw 'select' or other interaction-based events (but will throw ready or error events), and will not display hovertext or otherwise change depending on user input.

`fontSize` a number. Default automatic. The default font size, in pixels, of all text in the chart. You can override this using properties for specific chart elements.

`fontName` a string. Default `'Arial'`. The default font face for all text in the chart. You can override this using properties for specific chart elements.

`hAxis` a JSON object. Default `'null'`. An object with members to configure various horizontal axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`hAxis.baseline` a number. Default automatic. The baseline for the horizontal axis. This option is only supported for a continuous axis.

`hAxis.baselineColor` a string. Default `'black'`. The color of the baseline for the horizontal axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`. This option is only supported for a continuous axis.

`hAxis.direction` 1 or -1. Default 1. The direction in which the values along the vertical axis grow. Specify -1 to reverse the order of the values.

`hAxis.format` a string. Default auto. A format string for numeric or date axis labels.
For number axis labels, this is a subset of the decimal formatting ICU pattern set. For instance,

`{format:'#,###%'}.`

will display values \code"1,000%", "750%", and `"50%"` for values 10, 7.5, and 0.5.
For date axis labels, this is a subset of the date formatting ICU pattern set. For instance,

`{format:'MMM d, y'}.`

will display the value "Jul 1, 2011" for the date of July first in 2011.
The actual formatting applied to the label is derived from the locale the API has been loaded with. For more details, see loading charts with a specific locale.
This option is only supported for a continuous axis.

`hAxis.gridlines` a JSON object. Default null. An object with members to configure the gridlines on the horizontal axis. To specify properties of this object, you can use object literal notation, as shown here:

`{color: '#333', count: 4}`

This option is only supported for a continuous axis.

`hAxis.gridlines.color` a string. Default `'#CCC'`. The color of the vertical gridlines inside the chart area. Specify a valid HTML color string.

`hAxis.gridlines.count` a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

`hAxis.logScale` boolean. Default `FALSE`. `vAxis` property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to `TRUE` for yes. This option is only supported for a continuous axis.

`hAxis.textPosition` a string. Default `'out'` Position of the horizontal axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`hAxis.textStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}`

An object that specifies the horizontal axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.title` a string. Default `'null'`. `hAxis` property that specifies the title of the horizontal axis.

`hAxis.titleTextStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}.`

An object that specifies the horizontal axis title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.slantedText` boolean. Default automatic. If true, draw the horizontal axis text at an angle, to help fit more text along the axis; if false, draw horizontal axis text upright. Default behavior is to slant text if it cannot all fit when drawn upright. Notice that this option is available only when the `hAxis.textPosition` is set to `'out'` (which is the default).
This option is only supported for a discrete axis.

`hAxis.slantedTextAngle` a number, 1-90. Default 30. The angle of the horizontal axis text, if it's drawn slanted. Ignored if `hAxis.slantedText` is false, or is in auto mode, and the chart decided to draw the text horizontally.
This option is only supported for a discrete axis.

`hAxis.maxAlternation` a number. Default 2. Maximum number of levels of horizontal axis text. If axis text labels become too crowded, the server might shift neighboring labels up or down in order to fit labels closer together. This value specifies the most number of levels to use; the server can use fewer levels, if labels can fit without overlapping.
This option is only supported for a discrete axis.

`hAxis.showTextEvery` a number. Default automatic. How many horizontal axis labels to show, where 1 means show every label, 2 means show every other label, and so on. Default is to try to show as many labels as possible without overlapping.

This option is only supported for a discrete axis.

`hAxis.maxValue` a number. Default automatic. `hAxis` property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.

This option is only supported for a continuous axis.

`hAxis.minValue` a number. Default automatic. `hAxis` property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.

This option is only supported for a continuous axis.

`hAxis.viewWindowMode` a string. Default `"pretty"` if `hAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the horizontal axis to render the values within the chart area. The following string values are supported:

`'pretty'` Scale the horizontal values so that the maximum and minimum data values are rendered a bit inside the left and right of the chart area.

`'maximized'` Scale the horizontal values so that the maximum and minimum data values touch the left and right of the chart area.

`'explicit'` Specify the left and right scale values of the chart area. Data values outside these values will be cropped. You must specify a `hAxis.viewWindow` object describing the maximum and minimum values to show.

This option is only supported for a continuous axis.

`hAxis.viewWindow` Object. Default `NULL`. Specifies the cropping range of the horizontal axis.

`hAxis.viewWindow.max` A number. Default auto.

For a continuous axis The maximum horizontal data value to render. Has an effect only if `hAxis.viewWindowMode='explicit'`.

For a discrete axis The zero-based row index where the cropping window ends. Data points at this index and higher will be cropped out. In conjunction with `vAxis.viewWindowMode.min`, it defines a half-opened range `[min, max)` that denotes the element indices to display. In other words, every index such that `min <= index < max` will be displayed.

`hAxis.viewWindow.min` A number. Default auto.

For a continuous axis The minimum horizontal data value to render. Has an effect only if `hAxis.viewWindowMode='explicit'`.

For a discrete axis The zero-based row index where the cropping window begins. Data points at indices lower than this will be cropped out. In conjunction with `vAxis.viewWindowMode.max`, it defines a half-opened range `[min, max)` that denotes the element indices to display. In other words, every index such that `min <= index < max` will be displayed.

`height` a number. Default height of the containing element. Height of the chart, in pixels.

`legend` a JSON object. Default `NULL`. An object with members to configure various aspects of the legend. To specify properties of this object, you can use object literal notation, as shown here:

`{position: 'top', textStyle: {color: 'blue', fontSize: 16}}`

`legend.position` a string. Default `'right'`. Position of the legend. Can be one of the following:

`'right'` To the right of the chart.

`'top'` Above the chart.

`'bottom'` Below the chart.

`'none'` No legend is displayed.

`legend.textStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the legend text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`sizeAxis` a JSON object. Default `NULL`. An object with members to configure how values are associated with bubble size. To specify properties of this object, you can use object literal notation, as shown here:

`{minValue: 0,  maxSize: 20}`

`sizeAxis.maxSize` a number. Default 30. Maximum radius of the largest possible bubble, in pixels.

`sizeAxis.maxValue` a number. Default set to maximum value of size column in chart data. The size value (as appears in the chart data) to be mapped to `sizeAxis.maxSize`. Larger values will be cropped to this value.

`sizeAxis.minSize` a number. Default 5. Minimum radius of the largest possible bubble, in pixels.

`sizeAxis.minValue` a number. Default set to minimum value of size column in chart data. The size value (as appears in the chart data) to be mapped to `sizeAxis.minSize`. Smaller values will be cropped to this value.

`sortBubblesBySize` boolean. Default `TRUE`. If true, sorts the bubbles by size so the smaller bubbles appear above the larger bubbles. If false, bubbles are sorted according to their order in `data`.

`theme` a string. Default `NULL`. A theme is a set of predefined option values that work together to achieve a specific chart behavior or visual effect. Currently only one theme is available:

`maximized` Maximizes the area of the chart, and draws the legend and all of the labels inside the chart area. Sets the following options:

```
chartArea: {width: '100%', height: '100%'},
legend: {position: 'in'},
titlePosition: 'in', axisTitlesPosition: 'in',
hAxis: {textPosition: 'in'}, vAxis: {textPosition: 'in'}
```

`title` a string. Default no title. Text to display above the chart.

`titlePosition` a string. Default `'out'`. Where to place the chart title, compared to the chart area. Supported values:

`'in'` Draw the title inside the chart area.

`'out'` Draw the title outside the chart area.

`'none'` Omit the title.

`titleTextStyle` a JSON object. Default

`{color:'black', fontName:<global-font-name>,fontSize:<global-font-size>}.`

An object that specifies the title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip` a JSON object. Default `NULL`. An object with members to configure various tooltip elements. To specify properties of this object, you can use object literal notation, as shown here:

`{textStyle: {color: '#FF0000'}, showColorCode: true}`

`tooltip.showColorCode` boolean. Default automatic. If true, show colored squares next to the series information in the tooltip. The default is true when `focusTarget` is set to `'category'`, otherwise the default is `FALSE`.

`tooltip.textStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}`

An object that specifies the tooltip text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip.trigger` The user interaction that causes the tooltip to be displayed:

`'hover'` The tooltip will be displayed when the user hovers over an element.

`'none'` The tooltip will not be displayed.

`vAxis` a JSON object. Default `'null'`. An object with members to configure various vertical axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`vAxis.baseline` a number. Default automatic. `vAxis` property that specifies the baseline for the vertical axis. If the baseline is smaller than the highest grid line or smaller than the lowest grid line, it will be rounded tothe closest gridline.

`vAxis.baselineColor` a string. Default `'black'`. `vAxis` property that specifies the color of the baseline for the vertical axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`.

`vAxis.direction` 1 or -1. Default 1. The direction in which the values along the vertical axis grow. Specify -1 to reverse the order of the values.

`vAxis.format` a string. Default auto. A format string for numeric axis labels. This is a subset of the ICU pattern set. For instance, {format:'#,###%'}.

will display values 1,000%, 750%, and 50% for values 10, 7.5, and 0.5.

vAxis.gridlines a JSON object. Default NULL. An object with members to configure the gridlines on the vertical axis. To specify properties of this object, you can use object literal notation, as shown here:

{color: '#333', count: 4}

vAxis.gridlines.color a string. Default '#CCC'. The color of the vertical gridlines inside the chart area. Specify a valid HTML color string.

vAxis.gridlines.count a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

vAxis.logScale boolean. Default FALSE. vAxis property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to TRUE for yes.

vAxis.textPosition a string. Default 'out'. Position of the vertical axis text, relative to the chart area. Supported values: 'out', 'in', 'none'.

vAxis.textStyle a JSON object. Default

{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}.

An object that specifies the vertical axis text style. The object has this format:

{color: <string>, fontName: <string>, fontSize: <number>}

The color can be any HTML color string, for example: 'red' or '#00cc00'. Also see fontName and fontSize.

vAxis.title a string. Default no title. vAxis property that specifies a title for the vertical axis.

vAxis.titleTextStyle a JSON object. Default

{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}.

An object that specifies the vertical axis title text style. The object has this format:

{color: <string>, fontName: <string>, fontSize: <number>}

The color can be any HTML color string, for example: 'red' or '#00cc00'. Also see fontName and fontSize.

vAxis.maxValue a number. Default automatic. vAxis property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.

vAxis.minValue a number. Default automatic. vAxis property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.

vAxis.viewWindowMode a string. Default "pretty" if vAxis.viewWindow is null, "explicit" otherwise. Specifies how to scale the vertical axis to render the values within the chart area. The following string values are supported:

- `'pretty'` Scale the vertical values so that the maximum and minimum data values are rendered a bit inside the top and bottom of the chart area.
- `'maximized'` Scale the vertical values so that the maximum and minimum data values touch the top and bottom of the chart area.
- `'explicit'` Specify the top and bottom scale values of the chart area. Data values outside these values will be cropped. You must specify a `vAxis.viewWindow` object describing the maximum and minimum values to show.

`vAxis.viewWindow` a JSON object. Specifies the cropping range of the vertical axis.

`vAxis.viewWindow.max` A number. Default 0. The maximum vertical data value to render.
Has an effect only if vAxis.viewWindowMode='explicit'.

`vAxis.viewWindow.min` A number. Default 0. The minimum vertical data value to render.
Has an effect only if vAxis.viewWindowMode='explicit'.

`width` a number. Default width of the containing element. Width of the chart, in pixels.

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

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, here 'BubbleChart'
`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.
- `jsChart` Call of the `jsDisplayChart` function.
- `jsFooter` End tag `</script>`.
- `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

Google Visualisation API: You cannot load both bubblechart and corechart packages at the same time on the same page.

Author(s)

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

Diego de Castillo <`decastillo@gmail.com`>

References

Google Bubble Chart API: `http://code.google.com/apis/chart/interactive/docs/gallery/bubblechart.html`

Follow the link for Google's data policy.

See Also

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

Examples

```
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='{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='{minValue:75, maxValue:125}'))
plot(bubble3)

## Gradient colour example
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)
```

gvisCandlestickChart *Google Candlestick chart with R*

Description

An interactive candlestick chart.

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.

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 for Google Combo Chart. `gvis.editor` a character label for an on-page button which opens an in-page dialog box that enables users to edit, change and customise the chart. By default no value is given and therefore no button is displayed. Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/candlestickchart.html#Configuration_Options`:

`axisTitlesPosition` a string. Default 'out'. Where to place the axis titles, compared to the chart area. Supported values:

`'in'` Draw the axis titles inside the the chart area.

`'out'` Draw the axis titles outside the chart area.

`'none'` Omit the axis titles.

`backgroundColor` a string or object. Default `'white'`. The background color for the main area of the chart. Can be either a simple HTML color string, for example: `'red'` or `'#00cc00'`, or an object with the following properties.

`backgroundColor.stroke` a string. Default `'#666'`. The color of the chart border, as an HTML color string.

`backgroundColor.strokeWidth` a number. Default 0. The border width, in pixels.

`backgroundColor.fill` a string. Default `'white'`. The chart fill color, as an HTML color string.

`chartArea` a string. Default `'null'`. An object with members to configure the placement and size of the chart area (where the chart itself is drawn, excluding axis and legends). Two formats are supported: a number, or a number followed by `%`. A simple number is a value in pixels; a number followed by `%` is a percentage. Example:

`{left:20,top:0,width:\"50%\",height:\"75%\"}`

`chartArea.left` a number or string. Default auto. How far to draw the chart from the left border.

`chartArea.top` a number or string. Default auto. How far to draw the chart from the top border.

`chartArea.width` a number or string. Default auto. Chart area width.

`chartArea.height` a number or string. Default auto. Chart area height.

`colors` a JSON array of strings. Default 'colors'. The colors to use for the chart elements. An array of strings, where each element is an HTML color string, for example: `colors:['red','#004411']`.

`enableInteractivity` boolean. Default `TRUE`. Whether the chart throws user-based events or reacts to user interaction. If false, the chart will not throw 'select' or other interaction-based events (but will throw ready or error events), and will not display hovertext or otherwise change depending on user input.

`focusTarget` a string. Default `'datum'`. The type of the entity that receives focus on mouse hover. Also affects which entity is selected by mouse click, and which data table element is associated with events. Can be one of the following:

`'datum'` Focus on a single data point. Correlates to a cell in the data.

`'category'` Focus on a grouping of all data points along the major axis. Correlates to a row in the data table.

In focusTarget `'category'` the tooltip displays all the category values. This may be useful for comparing values of different series.

`fontSize` a number. Default automatic. The default font size, in pixels, of all text in the chart. You can override this using properties for specific chart elements.

`fontName` a string. Default `'Arial'`. The default font face for all text in the chart. You can override this using properties for specific chart elements.

`hAxis` a JSON object. Default `'null'`. An object with members to configure various horizontal axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`hAxis.direction` 1 or -1. Default 1. The direction in which the values along the horizontal axis grow. Specify -1 to reverse the order of the values.

`hAxis.textPosition` a string. Default `'out'` Position of the horizontal axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`hAxis.textStyle` a JSON object. Default

```
{color: 'black',
fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the horizontal axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.title` a string. Default `'null'`. `hAxis` property that specifies the title of the horizontal axis.

`hAxis.titleTextStyle` a JSON object. Default

```
{color: 'black',
fontName: <global-font-name>, fontSize: <global-font-size>}.
```

An object that specifies the horizontal axis title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.slantedText` Boolean. Default automatic. If `TRUE`, draw the horizontal axis text at an angle, to help fit more text along the axis; if false, draw horizontal axis text upright. Default behavior is to slant text if it cannot all fit when drawn upright.

`hAxis.slantedTextAngle` a number, 1-90. Default 30. The angle of the horizontal axis text, if it's drawn slanted. Ignored if `hAxis.slantedText` is false, or is in auto mode, and the chart decided to draw the text horizontally.

`hAxis.maxAlternation` a number. Default 2. Maximum number of levels of horizontal axis text. If axis text labels become too crowded, the server might shift neighboring labels up or down in order to fit labels closer together. This value specifies the most number of levels to use; the server can use fewer levels, if labels can fit without overlapping.

`hAxis.showTextEvery` a number. Default automatic. How many horizontal axis labels to show, where 1 means show every label, 2 means show every other label, and so on. Default is to try to show as many labels as possible without overlapping.

`hAxis.viewWindowMode` a string. Default `"pretty"` if `hAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the horizontal axis to render the values within the chart area. The following string values are supported:

`'pretty'` Scale the horizontal values so that the maximum and minimum data values are rendered a bit inside the left and right of the chart area.

`'maximized'` Scale the horizontal values so that the maximum and minimum data values touch the left and right of the chart area.

`'explicit'` Specify the left and right scale values of the chart area. Data values outside these values will be cropped. You must specify a `hAxis.viewWindow` object describing the maximum and minimum values to show.

`hAxis.viewWindow` JSON object. Default `NULL`. Specifies the maximum and minimum data values to show on the horizontal axis. Present only if `vAxis.viewWindowMode='expl`

`hAxis.viewWindow.max` number. Default 0. The maximum vertical data value to render.

`hAxis.viewWindow.min` number. Default 0. The minimum vertical data value to render.

`height` a number. Default height of the containing element. Height of the chart, in pixels.

`legend` a string. Default `'right'`. Position of the legend. Can be one of the following:

`'right'` To the right of the chart.

`'top'` Above the chart.

`'bottom'` Below the chart.

`'none'` No legend is displayed.

`legend` a JSON object. Default `NULL`. An object with members to configure various aspects of the legend. To specify properties of this object, you can use object literal notation, as shown here:

```
{position: 'top', textStyle: {color: 'blue', fontSize: 16}}
```

`legend.position` a string. Default `'right'`. Position of the legend. Can be one of the following:

`'right'` To the right of the chart.

`'top'` Above the chart.

`'bottom'` Below the chart.

`'none'` No legend is displayed.

`legend.textStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the legend text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`reverseCategories` boolean. Default `FALSE`. If set to true, will draw series from right to left. The default is to draw left-to-right.

`series` a JSON array of objects, or object with nested objects. Default {}. An array of objects, each describing the format of the corresponding series in the chart. To use default values for a series, specify an empty object . If a series or a value is not specified, the global value will be used. Each object supports the following properties:

`color` The color to use for this series. Specify a valid HTML color string.

`targetAxisIndex` Which axis to assign this series to, where 0 is the default axis, and 1 is the opposite axis. Default value is 0; set to 1 to define a chart where different series are rendered against different axes. You can define a different scale for different axes.

`pointSize` Overrides the global pointSize value for this series.

`lineWidth` Overrides the global lineWidth value for this series.

`curveType` Overrides the global curveType value for this series.

`visibleInLegend` A boolean value, where true means that the series should have a legend entry, and false means that it should not. Default is `TRUE`.

You can specify either an array of objects, each of which applies to the series in the order given, or you can specify an object where each child has a numeric key indicating which series it applies to. For example, the following two declarations are identical, and declare the first series as black and absent from the legend, and the fourth as red and absent from the legend:

```
series: [{color: 'black', visibleInLegend: false},{}, {}, {color:
'red', visibleInLegend: false}]

series: {0:{color: 'black', visibleInLegend: false}, 3:{color: 'red',
visibleInLegend: false}}
```

`theme` a string. Default `NULL`. A theme is a set of predefined option values that work together to achieve a specific chart behavior or visual effect. Currently only one theme is available:

`maximized` Maximizes the area of the chart, and draws the legend and all of the labels inside the chart area. Sets the following options:

```
chartArea: {width: '100%', height: '100%'},
legend: {position: 'in'},
titlePosition: 'in', axisTitlesPosition: 'in',
hAxis: {textPosition: 'in'}, vAxis: {textPosition: 'in'}
```

`title` a string. Default no title. Text to display above the chart.

`titlePosition` a string. Default `'out'`. Where to place the chart title, compared to the chart area. Supported values:

`'in'` Draw the title inside the chart area.

`'out'` Draw the title outside the chart area.

`'none'` Omit the title.

`titleTextStyle` a JSON object. Default

`{color:'black', fontName:<global-font-name>,fontSize:<global-font-size>}`.

An object that specifies the title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip` a JSON object. Default `NULL`. An object with members to configure various tooltip elements. To specify properties of this object, you can use object literal notation, as shown here:

`{textStyle: {color: '#FF0000'}, showColorCode: true}`

`tooltip.showColorCode` boolean. Default automatic. If true, show colored squares next to the series information in the tooltip. The default is true when `focusTarget` is set to `'category'`, otherwise the default is `FALSE`.

`tooltip.TextStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the tooltip text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip.trigger` The user interaction that causes the tooltip to be displayed:

`'hover'` The tooltip will be displayed when the user hovers over an element.

`'none'` The tooltip will not be displayed.

`vAxes` a JSON array of objects, or object with child objects null. Specifies properties for individual vertical axes, if the chart has multiple vertical axes. Each child object is a `vAxis` object, and can contain all the properties supported by `vAxis`. These property values override any global settings for the same property.
To specify a chart with multiple vertical axes, first define a new axis using `series.targetAxisIndex`, then configure the axis using `vAxes`. The following example assigns series 2 to the right axis and specifies a custom title and text style for it:

```
series:{2:{targetAxisIndex:1}},
  vAxes:{1:{title:'Losses',textStyle:{color: 'red'}}}
```

This property can be either an object or an array: the object is a collection of objects, each with a numeric label that specifies the axis that it defines–this is the format shown above; the array is an array of objects, one per axis. For example, the following array-style notation is identical to the `vAxis` object shown above:

```
vAxes:[
{}, // Nothing specified for axis 0
{title:'Losses',textStyle:{color: 'red'}} // Axis 1
]
```

`vAxis` a JSON object. Default `'null'`. An object with members to configure various vertical axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`vAxis.baseline` a number. Default automatic. `vAxis` property that specifies the baseline for the vertical axis. If the baseline is smaller than the highest grid line or smaller than the lowest grid line, it will be rounded tothe closest gridline.

`vAxis.baselineColor` a string. Default `'black'`. `vAxis` property that specifies the color of the baseline for the vertical axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`.

`vAxis.direction` 1 or -1. Default 1. The direction in which the values along the vertical axis grow. Specify -1 to reverse the order of the values.

`vAxis.format` a string. Default auto. A format string for numeric axis labels. This is a subset of the ICU pattern set. For instance,

`{format:'#,###%'}`.

will display values 1,000%, 750%, and 50% for values 10, 7.5, and 0.5.

`vAxis.gridlines` a JSON object. Default `NULL`. An object with members to configure the gridlines on the vertical axis. To specify properties of this object, you can use object literal notation, as shown here:

`{color: '#333', count: 4}`

`vAxis.gridlines.color` a string. Default `'#CCC'`. The color of the vertical gridlines inside the chart area. Specify a valid HTML color string.

`vAxis.gridlines.count` a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

`vAxis.logScale` boolean. Default `FALSE`. `vAxis` property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to `TRUE` for yes.

`vAxis.textPosition` a string. Default `'out'`. Position of the vertical axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`vAxis.textStyle` a JSON object. Default

`{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`.

An object that specifies the vertical axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.title` a string. Default no title. `vAxis` property that specifies a title for the vertical axis.

`vAxis.titleTextStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}`.

An object that specifies the vertical axis title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.maxValue` a number. Default automatic. vAxis property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.

`vAxis.minValue` a number. Default automatic. `vAxis` property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.

`vAxis.viewWindowMode` a string. Default `"pretty"` if `vAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the vertical axis to render the values within the chart area. The following string values are supported:

- `'pretty'` Scale the vertical values so that the maximum and minimum data values are rendered a bit inside the top and bottom of the chart area.
- `'maximized'` Scale the vertical values so that the maximum and minimum data values touch the top and bottom of the chart area.
- `'explicit'` Specify the top and bottom scale values of the chart area. Data values outside these values will be cropped. You must specify a `vAxis.viewWindow` object describing the maximum and minimum values to show.

`vAxis.viewWindow` Object. Default `NULL`. Specifies the maximum and minimum data values to show on the vertical axis. Present only if `vAxis.viewWindowMode='explicit'`

`vAxis.viewWindow.max` A number. Default 0. The maximum vertical data value to render.

`vAxis.viewWindow.min` A number. Default 0. The minimum vertical data value to render.

`width` a number. Default width of the containing element. Width of the chart, in pixels.

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

Details

From `http://code.google.com/apis/chart/interactive/docs/gallery/candlestickchart.html#Overview`:

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, here 'CandlestickChart'

`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.
  - `jsChart` Call of the `jsDisplayChart` function.
  - `jsFooter` End tag `</script>`.
  - `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 Candlestick Chart API: `http://code.google.com/apis/chart/interactive/docs/gallery/candlestickchart.html`

Follow the link for Google's data policy.

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 for Google Column Chart.

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

Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/columnchart.html#Configuration_Options`:

`axisTitlesPosition` a string. Default 'out'. Where to place the axis titles, compared to the chart area. Supported values:

`'in'` Draw the axis titles inside the the chart area.

`'out'` Draw the axis titles outside the chart area.

`'none'` Omit the axis titles.

`backgroundColor` a string or object. Default `'white'`. The background color for the main area of the chart. Can be either a simple HTML color string, for example: `'red'` or `'#00cc00'`, or an object with the following properties.

`backgroundColor.stroke` a string. Default `'#666'`. The color of the chart border, as an HTML color string.

`backgroundColor.strokeWidth` a number. Default 0. The border width, in pixels.

`backgroundColor.fill` a string. Default `'white'`. The chart fill color, as an HTML color string.

`chartArea` a string. Default `'null'`. An object with members to configure the placement and size of the chart area (where the chart itself is drawn, excluding axis and legends). Two formats are supported: a number, or a number followed by `%`. A simple number is a value in pixels; a number followed by `%` is a percentage. Example:

`{left:20,top:0,width:\"50%\",height:\"75%\"}`

`chartArea.left` a number or string. Default auto. How far to draw the chart from the left border.

`chartArea.top` a number or string. Default auto. How far to draw the chart from the top border.

`chartArea.width` a number or string. Default auto. Chart area width.

`chartArea.height` a number or string. Default auto. Chart area height.

`colors` An array of strings. Default 'colors'. The colors to use for the chart elements. An array of strings, where each element is an HTML color string, for example: `colors:['red','#004411']`.

`enableInteractivity` boolean. Default `TRUE`. Whether the chart throws user-based events or reacts to user interaction. If false, the chart will not throw 'select' or other interaction-based events (but will throw ready or error events), and will not display hovertext or otherwise change depending on user input.

`focusTarget` a string. Default `'datum'`. The type of the entity that receives focus on mouse hover. Also affects which entity is selected by mouse click, and which data table element is associated with events. Can be one of the following:

`'datum'` Focus on a single data point. Correlates to a cell in the data.

`'category'` Focus on a grouping of all data points along the major axis. Correlates to a row in the data table.

In focusTarget `'category'` the tooltip displays all the category values. This may be useful for comparing values of different series.

`fontSize` a number. Default automatic. The default font size, in pixels, of all text in the chart. You can override this using properties for specific chart elements.

`fontName` a string. Default `'Arial'`. The default font face for all text in the chart. You can override this using properties for specific chart elements.

`hAxis` a JSON object. Default `'null'`. An object with members to configure various horizontal axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`hAxis.direction` 1 or -1. Default 1. The direction in which the values along the horizontal axis grow. Specify -1 to reverse the order of the values.

`hAxis.textPosition` a string. Default `'out'` Position of the horizontal axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`hAxis.textStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the horizontal axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.title` a string. Default `'null'`. `hAxis` property that specifies the title of the horizontal axis.

`hAxis.titleTextStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}.
```

An object that specifies the horizontal axis title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.slantedText` boolean. Default automatic. If `TRUE`, draw the horizontal axis text at an angle, to help fit more text along the axis; if false, draw horizontal axis text upright. Default behavior is to slant text if it cannot all fit when drawn upright.

`hAxis.slantedTextAngle` a number, 1-90. Default 30. The angle of the horizontal axis text, if it's drawn slanted. Ignored if `hAxis.slantedText` is false, or is in auto mode, and the chart decided to draw the text horizontally.

`hAxis.maxAlternation` a number. Default 2. Maximum number of levels of horizontal axis text. If axis text labels become too crowded, the server might shift neighboring labels up or down in order to fit labels closer together. This value specifies the most number of levels to use; the server can use fewer levels, if labels can fit without overlapping.

`hAxis.showTextEvery` a number. Default automatic. How many horizontal axis labels to show, where 1 means show every label, 2 means show every other label, and so on. Default is to try to show as many labels as possible without overlapping.

`hAxis.viewWindow` JSON object. Default `NULL`. Specifies the maximum and minimum data values to show on the horizontal axis. Present only if `vAxis.viewWindowMode='expl`

`hAxis.viewWindow.max` number. Default 0. The maximum vertical data value to render.

`hAxis.viewWindow.min` number. Default 0. The minimum vertical data value to render.

`height` a number. Default height of the containing element. Height of the chart, in pixels.

`isStacked` Boolean. Default `FALSE`. If set to `TRUE`, column values are stacked (accumulated).

`legend` a JSON object. Default `NULL`. An object with members to configure various aspects of the legend. To specify properties of this object, you can use object literal notation, as shown here:

`{position: 'top', textStyle: {color: 'blue', fontSize: 16}}` `legend.position` a string. Default `'right'`. Position of the legend. Can be one of the following:

`'right'` To the right of the chart.

`'top'` Above the chart.

`'bottom'` Below the chart.

`'none'` No legend is displayed.

`legend.textStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the legend text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`reverseCategories` boolean. Default `FALSE`. If set to true, will draw series from right to left. The default is to draw left-to-right.

`series` a JSON array of objects, or object with nested objects. Default {}. An array of objects, each describing the format of the corresponding series in the chart. To use default values for a series, specify an empty object . If a series or a value is not specified, the global value will be used. Each object supports the following properties:

`color` The color to use for this series. Specify a valid HTML color string.

`targetAxisIndex` Which axis to assign this series to, where 0 is the default axis, and 1 is the opposite axis. Default value is 0; set to 1 to define a chart where different series are rendered against different axes. You can define a different scale for different axes.

`pointSize` Overrides the global pointSize value for this series.

`lineWidth` Overrides the global lineWidth value for this series.

`curveType` Overrides the global curveType value for this series.

`visibleInLegend` A boolean value, where true means that the series should have a legend entry, and false means that it should not. Default is `TRUE`.

You can specify either an array of objects, each of which applies to the series in the order given, or you can specify an object where each child has a numeric key indicating which series it applies to. For example, the following two declarations are identical, and declare the first series as black and absent from the legend, and the fourth as red and absent from the legend:

```
series: [{color: 'black', visibleInLegend: false},{}, {}, {color:
'red', visibleInLegend: false}]

series: {0:{color: 'black', visibleInLegend: false}, 3:{color: 'red',
visibleInLegend: false}}
```

`theme` a string. Default `NULL`. A theme is a set of predefined option values that work together to achieve a specific chart behavior or visual effect. Currently only one theme is available:

`maximized` Maximizes the area of the chart, and draws the legend and all of the labels inside the chart area. Sets the following options:

```
chartArea: {width: '100%', height: '100%'},
legend: {position: 'in'},
titlePosition: 'in', axisTitlesPosition: 'in',
hAxis: {textPosition: 'in'}, vAxis: {textPosition: 'in'}
```

`title` a string. Default no title. Text to display above the chart.

`titlePosition` a string. Default `'out'`. Where to place the chart title, compared to the chart area. Supported values:

- `'in'` Draw the title inside the chart area.
- `'out'` Draw the title outside the chart area.
- `'none'` Omit the title.

`titleTextStyle` a JSON object. Default

`{color:'black', fontName:<global-font-name>,fontSize:<global-font-size>}`.

An object that specifies the title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip` a JSON object. Default `NULL`. An object with members to configure various tooltip elements. To specify properties of this object, you can use object literal notation, as shown here:

`{textStyle: {color: '#FF0000'}, showColorCode: true}`

`tooltip.showColorCode` boolean. Default automatic. If true, show colored squares next to the series information in the tooltip. The default is true when `focusTarget` is set to `'category'`, otherwise the default is `FALSE`.

`tooltip.TextStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the tooltip text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip.trigger` The user interaction that causes the tooltip to be displayed:

- `'hover'` The tooltip will be displayed when the user hovers over an element.
- `'none'` The tooltip will not be displayed.

`vAxes` Array of object, or object with child objects null. Specifies properties for individual vertical axes, if the chart has multiple vertical axes. Each child object is a `vAxis` object, and can contain all the properties supported by `vAxis`. These property values override any global settings for the same property.

To specify a chart with multiple vertical axes, first define a new axis using `series.targetAxisIndex`, then configure the axis using `vAxes`. The following example assigns series 2 to the right axis and specifies a custom title and text style for it:

```
series:{2:{targetAxisIndex:1}},
vAxes:{1:{title:'Losses',textStyle:{color: 'red'}}}
```

This property can be either an object or an array: the object is a collection of objects, each with a numeric label that specifies the axis that it defines–this is the format shown above; the array is an array of objects, one per axis. For example, the following array-style notation is identical to the `vAxis` object shown above:

```
vAxes:[
{}, // Nothing specified for axis 0
{title:'Losses',textStyle:{color: 'red'}} // Axis 1
]
```

`vAxis` a JSON object. Default `'null'`. An object with members to configure various vertical axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`vAxis.baseline` a number. Default automatic. `vAxis` property that specifies the baseline for the vertical axis. If the baseline is smaller than the highest grid line or smaller than the lowest grid line, it will be rounded tothe closest gridline.

`vAxis.baselineColor` a string. Default `'black'`. `vAxis` property that specifies the color of the baseline for the vertical axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`.

`vAxis.direction` 1 or -1. Default 1. The direction in which the values along the vertical axis grow. Specify -1 to reverse the order of the values.

`vAxis.format` a string. Default auto. A format string for numeric axis labels. This is a subset of the ICU pattern set. For instance,

`{format:'#,###%'}`.

will display values 1,000%, 750%, and 50% for values 10, 7.5, and 0.5.

`vAxis.gridlines.color` a string. Default `'#CCC'`. The color of the vertical gridlines inside the chart area. Specify a valid HTML color string.

`vAxis.gridlines.count` a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

`vAxis.logScale` boolean. Default `FALSE`. `vAxis` property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to `TRUE` for yes.

`vAxis.textPosition` a string. Default `'out'`. Position of the vertical axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`vAxis.textStyle` a JSON object. Default

`{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`.

An object that specifies the vertical axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.title` a string. Default no title. `vAxis` property that specifies a title for the vertical axis.

`vAxis.titleTextStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}.`

An object that specifies the vertical axis title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.maxValue` a number. Default automatic. vAxis property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.

`vAxis.minValue` a number. Default automatic. `vAxis` property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.

`vAxis.viewWindowMode` a string. Default `"pretty"` if `vAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the vertical axis to render the values within the chart area. The following string values are supported:

- `'pretty'` Scale the vertical values so that the maximum and minimum data values are rendered a bit inside the top and bottom of the chart area.
- `'maximized'` Scale the vertical values so that the maximum and minimum data values touch the top and bottom of the chart area.
- `'explicit'` Specify the top and bottom scale values of the chart area. Data values outside these values will be cropped. You must specify a `vAxis.viewWindow` object describing the maximum and minimum values to show.

`vAxis.viewWindow` Object. Default `NULL`. Specifies the maximum and minimum data values to show on the vertical axis. Present only if `vAxis.viewWindowMode='explicit'`

`vAxis.viewWindow.max` A number. Default 0. The maximum vertical data value to render.

`vAxis.viewWindow.min` A number. Default 0. The minimum vertical data value to render.

`width` a number. Default width of the containing element. Width of the chart, in pixels.

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

Value

`gvisColumnChart` returns a 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 'ColumnChart'
`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.
- `jsChart` Call of the `jsDisplayChart` function.
- `jsFooter` End tag `</script>`.
- `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

Google Visualisation API: You cannot load both barchart/columnchart and corechart packages at the same time on the same page.

Author(s)

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

Diego de Castillo `<decastillo@gmail.com>`

References

Google Column Chart API: `http://code.google.com/apis/chart/interactive/docs/gallery/columnchart.html`

Follow the link for Google's data policy.

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 smoothed curve
Col3 <- gvisColumnChart(df, xvar="country", yvar=c("val1", "val2"),
             options=list(title="Hello World",
                          titleTextStyle="{color:'red',fontName:'Courier',fontSize:16}",
                          curveType='function'))
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.

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.

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 for Google Combo Chart.

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

Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/combochart.html#Configuration_Options`:

`areaOpacity` a number, 0.0 - 1.0. Default 0.3. The default opacity of the colored area under an area chart series, where 0.0 is fully transparent and 1.0 is fully opaque. To specify opacity for an individual series, set the areaOpacity value in the series property.

`axisTitlesPosition` a string. Default 'out'. Where to place the axis titles, compared to the chart area. Supported values:

`'in'` Draw the axis titles inside the the chart area.

`'out'` Draw the axis titles outside the chart area.

`'none'` Omit the axis titles.

`backgroundColor` a string or object. Default `'white'`. The background color for the main area of the chart. Can be either a simple HTML color string, for example: `'red'` or `'#00cc00'`, or an object with the following properties.

`backgroundColor.stroke` a string. Default `'#666'`. The color of the chart border, as an HTML color string.

`backgroundColor.strokeWidth` a number. Default 0. The border width, in pixels.

`backgroundColor.fill` a string. Default `'white'`. The chart fill color, as an HTML color string.

`chartArea` a string. Default `'null'`. An object with members to configure the placement and size of the chart area (where the chart itself is drawn, excluding axis and legends). Two formats are supported: a number, or a number followed by `%`. A simple number is a value in pixels; a number followed by `%` is a percentage. Example:

`{left:20,top:0,width:\"50%\",height:\"75%\"}`

`chartArea.left` a number or string. Default auto. How far to draw the chart from the left border.

`chartArea.top` a number or string. Default auto. How far to draw the chart from the top border.

`chartArea.width` a number or string. Default auto. Chart area width.

`chartArea.height` a number or string. Default auto. Chart area height.

`colors` a JSON array of strings. Default 'colors'. The colors to use for the chart elements. An array of strings, where each element is an HTML color string, for example: `colors:['red','#004411']`.

`curveType` [Line and area series only] a string. Default `'none'`. Controls the curve of the lines. Can be one of the following:

`'none'` Straight lines without curve.

`'function'` The angles of the line will be smoothed.

`enableInteractivity` boolean. Default `TRUE`. Whether the chart throws user-based events or reacts to user interaction. If false, the chart will not throw 'select' or other interaction-based events (but will throw ready or error events), and will not display hovertext or otherwise change depending on user input.

`focusTarget` a string. Default `'datum'`. The type of the entity that receives focus on mouse hover. Also affects which entity is selected by mouse click, and which data table element is associated with events. Can be one of the following:

- `'datum'` Focus on a single data point. Correlates to a cell in the data.
- `'category'` Focus on a grouping of all data points along the major axis. Correlates to a row in the data table.

In focusTarget `'category'` the tooltip displays all the category values. This may be useful for comparing values of different series.

`fontSize` a number. Default automatic. The default font size, in pixels, of all text in the chart. You can override this using properties for specific chart elements.

`fontName` a string. Default `'Arial'`. The default font face for all text in the chart. You can override this using properties for specific chart elements.

`hAxis.baseline` a number. Default automatic. The baseline for the horizontal axis. This option is only supported for a continuous axis.

`hAxis.baselineColor` a string. Default `'black'`. The color of the baseline for the horizontal axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`. This option is only supported for a continuous axis.

`hAxis.direction` 1 or -1. Default 1. The direction in which the values along the vertical axis grow. Specify -1 to reverse the order of the values.

`hAxis.format` a string. Default auto. A format string for numeric or date axis labels.
For number axis labels, this is a subset of the decimal formatting ICU pattern set. For instance,

`{format:'#,###%'}.`

will display values \code"1,000%", "750%", and `"50%"` for values 10, 7.5, and 0.5.
For date axis labels, this is a subset of the date formatting ICU pattern set. For instance,

`{format:'MMM d, y'}.`

will display the value "Jul 1, 2011" for the date of July first in 2011.
The actual formatting applied to the label is derived from the locale the API has been loaded with. For more details, see loading charts with a specific locale.
This option is only supported for a continuous axis.

`hAxis.gridlines` a JSON object. Default null. An object with members to configure the gridlines on the horizontal axis. To specify properties of this object, you can use object literal notation, as shown here:

`{color: '#333', count: 4}`

This option is only supported for a continuous axis.

`hAxis.gridlines.color` a string. Default `'#CCC'`. The color of the vertical gridlines inside the chart area. Specify a valid HTML color string.

`hAxis.gridlines.count` a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

`hAxis.logScale` boolean. Default `FALSE`. `vAxis` property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to `TRUE` for yes. This option is only supported for a continuous axis.

`hAxis.textPosition` a string. Default `'out'` Position of the horizontal axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`hAxis.textStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}`

An object that specifies the horizontal axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.title` a string. Default `'null'`. `hAxis` property that specifies the title of the horizontal axis.

`hAxis.titleTextStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}.`

An object that specifies the horizontal axis title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.slantedText` boolean. Default automatic. If true, draw the horizontal axis text at an angle, to help fit more text along the axis; if false, draw horizontal axis text upright. Default behavior is to slant text if it cannot all fit when drawn upright. Notice that this option is available only when the `hAxis.textPosition` is set to `'out'` (which is the default).
This option is only supported for a discrete axis.

`hAxis.slantedTextAngle` a number, 1-90. Default 30. The angle of the horizontal axis text, if it's drawn slanted. Ignored if `hAxis.slantedText` is false, or is in auto mode, and the chart decided to draw the text horizontally.
This option is only supported for a discrete axis.

`hAxis.maxAlternation` a number. Default 2. Maximum number of levels of horizontal axis text. If axis text labels become too crowded, the server might shift neighboring labels up or down in order to fit labels closer together. This value specifies the most number of levels to use; the server can use fewer levels, if labels can fit without overlapping.
This option is only supported for a discrete axis.

`hAxis.showTextEvery` a number. Default automatic. How many horizontal axis labels to show, where 1 means show every label, 2 means show every other label, and so on. Default is to try to show as many labels as possible without overlapping.

This option is only supported for a discrete axis.

`hAxis.maxValue` a number. Default automatic. `hAxis` property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.

This option is only supported for a continuous axis.

`hAxis.minValue` a number. Default automatic. `hAxis` property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.

This option is only supported for a continuous axis.

`hAxis.viewWindowMode` a string. Default `"pretty"` if `hAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the horizontal axis to render the values within the chart area. The following string values are supported:

`'pretty'` Scale the horizontal values so that the maximum and minimum data values are rendered a bit inside the left and right of the chart area.

`'maximized'` Scale the horizontal values so that the maximum and minimum data values touch the left and right of the chart area.

`'explicit'` Specify the left and right scale values of the chart area. Data values outside these values will be cropped. You must specify a `hAxis.viewWindow` object describing the maximum and minimum values to show.

This option is only supported for a continuous axis.

`hAxis.viewWindow` Object. Default `NULL`. Specifies the cropping range of the horizontal axis.

`hAxis.viewWindow.max` A number. Default auto.

For a continuous axis The maximum horizontal data value to render. Has an effect only if `hAxis.viewWindowMode='explicit'`.

For a discrete axis The zero-based row index where the cropping window ends. Data points at this index and higher will be cropped out. In conjunction with `vAxis.viewWindowMode.min`, it defines a half-opened range `[min, max)` that denotes the element indices to display. In other words, every index such that `min <= index < max` will be displayed.

`hAxis.viewWindow.min` A number. Default auto.

For a continuous axis The minimum horizontal data value to render. Has an effect only if `hAxis.viewWindowMode='explicit'`.

For a discrete axis The zero-based row index where the cropping window begins. Data points at indices lower than this will be cropped out. In conjunction with `vAxis.viewWindowMode.max`, it defines a half-opened range `[min, max)` that denotes the element indices to display. In other words, every index such that `min <= index < max` will be displayed.

`height` a number. Default height of the containing element. Height of the chart, in pixels.

`interpolateNulls` boolean. Default `FALSE`. Whether to guess the value of missing points. If true, it will guess the value of any missing data based on neighboring points. If false, it will leave a break in the line at the unknown point.

`legend` a JSON object. Default `NULL`. An object with members to configure various aspects of the legend. To specify properties of this object, you can use object literal notation, as shown here:

```
{position: 'top', textStyle: {color: 'blue', fontSize: 16}}
```

`legend.position` a string. Default `'right'`. Position of the legend. Can be one of the following:

`'right'` To the right of the chart.

`'top'` Above the chart.

`'bottom'` Below the chart.

`'none'` No legend is displayed.

`legend.textStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the legend text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`lineWidth` a number. Default 2. Line width in pixels. Use zero to hide all lines and show only the points.

`reverseCategories` Boolean. Default `FALSE`. If set to true, will draw series from right to left. The default is to draw left-to-right.

`series` a JSON array of objects, or object with nested objects. Default `'{}'`. An array of objects, each describing the format of the corresponding series in the chart. To use default values for a series, specify an empty object . If a series or a value is not specified, the default value will be used. Each object supports the following properties:

- `type` The type of marker for this series. Valid values are `'line'`, `'bars'`, and `'area'`. Note that bars are actually vertical bars (columns). The default value is specified by the chart's seriesType option.
- `color` A valid HTML color string.
- `pointSize` [line and area series only] The size of the circle on each data point, in pixels.
- `lineWidth` [line and area series only] The width of the line, in pixels.

You can either specify an array of objects, each of which applies to the series in the order given, or you can specify an object where each child has a numeric key indicating which series it applies to. For example, the following two declarations are identical, and apply to the first and fourth series:

```
series:[{color: 'black', type:    'bars'},{}, {}, {color: 'teal', lineWidth: 7}]
series:{0:{color: 'black', type: 'bars'},  3:{color: 'teal', lineWidth: 7}}
```

The following example sets the first series to a black line with 3 pixel points, the fourth series to a teal, seven pixel wide area line, and all other series are default values:
`series: [{type: 'line', color: 'black',  pointSize:3},{}, {}, {type:'area', colo`

`seriesType` a string. Default `'line'`. The default line type for any series not specified in the series property. Available values are `'line'`, `'bars'`, and `'area'`.

`theme` a string. Default `NULL`. A theme is a set of predefined option values that work together to achieve a specific chart behavior or visual effect. Currently only one theme is available:

`maximized` Maximizes the area of the chart, and draws the legend and all of the labels inside the chart area. Sets the following options:

```
chartArea: {width: '100%', height: '100%'},
legend: {position: 'in'},
titlePosition: 'in', axisTitlesPosition: 'in',
hAxis: {textPosition: 'in'}, vAxis: {textPosition: 'in'}
```

`title` a string. Default no title. Text to display above the chart.

`titlePosition` a string. Default `'out'`. Where to place the chart title, compared to the chart area. Supported values:

`'in'` Draw the title inside the chart area.

`'out'` Draw the title outside the chart area.

`'none'` Omit the title.

`titleTextStyle` a JSON object. Default
`{color:'black', fontName:<global-font-name>,fontSize:<global-font-size>}`.
An object that specifies the title text style. The object has this format:
`{color: <string>, fontName: <string>, fontSize: <number>}`
The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip` a JSON object. Default `NULL`. An object with members to configure various tooltip elements. To specify properties of this object, you can use object literal notation, as shown here:
`{textStyle: {color: '#FF0000'}, showColorCode: true}`

`tooltip.showColorCode` boolean. Default automatic. If true, show colored squares next to the series information in the tooltip. The default is true when `focusTarget` is set to `'category'`, otherwise the default is `FALSE`.

`tooltip.TextStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the tooltip text style. The object has this format:
`{color: <string>, fontName: <string>, fontSize: <number>}`
The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip.trigger` The user interaction that causes the tooltip to be displayed:

`'hover'` The tooltip will be displayed when the user hovers over an element.

`'none'` The tooltip will not be displayed.

`vAxes` a JSON array of objects, or object with child objects null. Specifies properties for individual vertical axes, if the chart has multiple vertical axes. Each child object is a `vAxis` object, and can contain all the properties supported by `vAxes`. These property values override any global settings for the same property.
To specify a chart with multiple vertical axes, first define a new axis using `series.targetAxisIndex`, then configure the axis using `vAxes`. The following example assigns series 2 to the right axis and specifies a custom title and text style for it:

```
series:{2:{targetAxisIndex:1}},
  vAxes:{1:{title:'Losses',textStyle:{color: 'red'}}}
```

This property can be either an object or an array: the object is a collection of objects, each with a numeric label that specifies the axis that it defines–this is the format shown above; the array is an array of objects, one per axis. For example, the following array-style notation is identical to the `vAxis` object shown above:

```
vAxes:[
{}, // Nothing specified for axis 0
{title:'Losses',textStyle:{color: 'red'}} // Axis 1
]
```

`vAxis` a JSON object. Default `'null'`. An object with members to configure various vertical axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`vAxis.baseline` a number. Default automatic. `vAxis` property that specifies the baseline for the vertical axis. If the baseline is smaller than the highest grid line or smaller than the lowest grid line, it will be rounded tothe closest gridline.

`vAxis.baselineColor` a string. Default `'black'`. `vAxis` property that specifies the color of the baseline for the vertical axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`.

`vAxis.direction` 1 or -1. Default 1. The direction in which the values along the vertical axis grow. Specify -1 to reverse the order of the values.

`vAxis.format` a string. Default auto. A format string for numeric axis labels. This is a subset of the ICU pattern set. For instance,

`{format:'#,###%'}`.

will display values 1,000%, 750%, and 50% for values 10, 7.5, and 0.5.

`vAxis.gridlines.color` a string. Default `'#CCC'`. The color of the vertical gridlines inside the chart area. Specify a valid HTML color string.

`vAxis.gridlines.count` a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

`vAxis.logScale` boolean. Default `FALSE`. `vAxis` property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to `TRUE` for yes.

`vAxis.textPosition` a string. Default `'out'`. Position of the vertical axis text, relative to the chart area. Supported values: `'out'`, `'in'`, `'none'`.

`vAxis.textStyle` a JSON object. Default

`{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}.`

An object that specifies the vertical axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.title` a string. Default no title. `vAxis` property that specifies a title for the vertical axis.

`vAxis.titleTextStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}.
```

An object that specifies the vertical axis title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.maxValue` a number. Default automatic. vAxis property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.

`vAxis.minValue` a number. Default automatic. `vAxis` property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.

`vAxis.viewWindowMode` a string. Default `"pretty"` if `vAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the vertical axis to render the values within the chart area. The following string values are supported:

`'pretty'` Scale the vertical values so that the maximum and minimum data values are rendered a bit inside the top and bottom of the chart area.

`'maximized'` Scale the vertical values so that the maximum and minimum data values touch the top and bottom of the chart area.

`'explicit'` Specify the top and bottom scale values of the chart area. Data values outside these values will be cropped. You must specify a `vAxis.viewWindow` object describing the maximum and minimum values to show.

`vAxis.viewWindow` a JSON object. Specifies the cropping range of the vertical axis.

`vAxis.viewWindow.max` A number. Default 0. The maximum vertical data value to render.
Has an effect only if vAxis.viewWindowMode='explicit'.

`vAxis.viewWindow.min` A number. Default 0. The minimum vertical data value to render.
Has an effect only if vAxis.viewWindowMode='explicit'.

`width` a number. Default width of the containing element. Width of the chart, in pixels.

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

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, here 'ComboChart'
`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.
- `jsChart` Call of the `jsDisplayChart` function.
- `jsFooter` End tag `</script>`.
- `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 Combo Chart API: `http://code.google.com/apis/chart/interactive/docs/gallery/combochart.html`

Follow the link for Google's data policy.

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.

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)
```

`gvisGauge` *Google Gauge with R*

Description

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 for Google Gauge. `gvis.editor` a character label for an on-page button which opens an in-page dialog box that enables users to edit, change and customise the chart. By default no value is given and therefore no button is displayed. Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/gauge.html#Configuration_Options`:

- `greenColor` a string. Default `'#109618'`. The color to use for the green section, in HTML color notation.
- `greenFrom` a number. Default none. The lowest value for a range marked by a green color.
- `greenTo` a number. Default none. The highest value for a range marked by a green color.
- `height` a number. Default container's width. Height of the chart in pixels.
- `majorTicks` an JSON array of strings. Default none. Labels for major tick marks. The number of labels define the number of major ticks in all gauges. The default is five major ticks, with the labels of the minimal and maximal gauge value.
- `max` a number. Default 100. The maximal value of a gauge.
- `min` a number. Default 0. The minimal value of a gauge.
- `minorTicks` a number. Default 2. The number of minor tick section in each major tick section.
- `redColor` a string. Default `'#DC3912'`. The color to use for the red section, in HTML color notation.
- `redFrom` a number. Default none. The lowest value for a range marked by a red color.
- `redTo` a number. Default none. The highest value for a range marked by a red color.
- `width` a number. Default container's width. Width of the chart in pixels.
- `yellowColor` a string. Default '#FF9900'. The color to use for the yellow section, in HTML color notation.
- `yellowFrom` a number. Default none. The lowest value for a range marked by a yellow color.
- `yellowTo` a number/ Default none. The highest value for a range marked by a yellow color.

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

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, here 'Gauge'

`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.

jsChart Call of the jsDisplayChart function.

jsFooter End tag </script>.

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 Gauge API: http://code.google.com/apis/chart/interactive/docs/gallery/gauge.html

Follow the link for Google's data policy.

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.

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.

Usage

```
gvisGeoChart(data, locationvar = "", colorvar="", sizevar="", 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 `http://code.google.com/apis/adwords/docs/developer/adwords_api_us_metros.html`. 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").

`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 valuesare not allowed.

`options` list of configuration options for Google Geo Chart.

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

Further possible components are, taken from https://google-developers.appspot.com/chart/interactive/docs/gallery/geochart.html#Configuration_Options:

`backgroundColor` a string or object. Default `white`. The background color for the main area of the chart. Can be either a simple HTML color string, for example: `'red'` or `'#00cc00'`, or an object with the following properties.

`backgroundColor.fill` a string. Default `white`. The chart fill color, as an HTML color string.

`backgroundColor.stroke` a string. Default `'#666'`. The color of the chart border, as an HTML color string.

`backgroundColor.strokeWidth` a number. Default 0. The border width, in pixels.

`colorAxis` a string. Default `'null'` An object that specifies a mapping between colors and color column values. To specify properties of this object, you can use object literal notation, as shown here:

`{minValue: 0,  colors: ['#FF0000', '#00FF00']}`

`colorAxis.minValue` a number. Default minimum value of color column in chart data. If present, specifies a minimum value for chart color data. Color data values of this value and lower will be rendered as the first color in the `colorAxis.colors` range.

`colorAxis.maxValue` a number. Default maximum value of color column in chart data If present, specifies a maximum value for chart color data. Color data values of this value and higher will be rendered as the last color in the `colorAxis.colors` range.

`colorAxis.values` a JSON array of numbers. Default `'null'`. Controls how values are associated with colors. Each value is associated with the corresponding color in the `colorAxis.colors` array. These values apply to the color value for a region or marker. Regions are colored according to a gradient of the values specified here. Not specifying a value for this option is equivalent to specifying `[minValue,  maxValue]`.

`colorAxis.colors` a JSON array of color strings. Default `'null'`. Colors to assign to values in the visualization. An array of strings, where each element is an HTML color string, for example: colorAxis:

`{colors:['red','#004411']}`.

You must have at least two values; the gradient will include all your values, plus calculated intermediary values, with the first color as the smallest value, and the last color as the highest.

`datalessRegionColor` a string. Default `'F5F5F5'`. Colors to assign to regions with no associated data.

`displayMode` a string. Default `'auto'`. Which type of map this is. The DataTable format must match the value specified. The following values are supported:

`'auto'`: Choose based on the format of the DataTable. `'regions'`: This is a region map `'markers'`: This is a marker map.

`enableRegionInteractivity` boolean. Default automatic. If true, enable region interactivity, including focus and tool-tip elaboration on mouse hover,

and region selection and firing of regionClick and select events on mouse click.
The default is `TRUE` in region mode, and `FALSE` in marker mode.

`height` number. The default height is 347 pixels, unless the width option is specified and `keepAspectRatio` is set to true - in which case the height is calculated accordingly.

`keepAspectRatio` boolean. Default `TRUE`. If true, the map will be drawn at the largest size that can fit inside the chart area at its natural aspect ratio. If only one of the width and height options is specified, the other one will be calculated according to the aspect ratio.
If false, the map will be stretched to the exact size of the chart as specified by the width and height options.

`legend` a JSON object / `'none'`. Default `'null'`. An object with members to configure various aspects of the legend, or `'none'`, if no legend should appear. To specify properties of this object, you can use object literal notation, as shown here:

`{textStyle: {color: 'blue', fontSize: 16}}`

`legend.numberFormat` a string. Default `'auto'`. A format string for numeric axis labels. This is a subset of the ICU pattern set. For instance,

`{numberFormat:'.##'}`.

will display values "10.66", "10.6", and "10.0" for values 10.666, 10.6, and 10.

`legend.textStyle` a JSON object. Default

```
{color:
  'black', fontName: <global-font-name>, fontSize:
  <global-font-size>}.
```

An object that specifies the legend text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`region` string, default `'world'`. The area to display on the map. (Surrounding areas will be displayed as well.) Can be one of the following:

- `'world'` A map of the entire world.
- A continent or a sub-continent, specified by its 3-digit code, e.g., '011' for Western Africa.
- A country, specified by its ISO 3166-1 alpha-2 code, e.g., `'AU'` for Australia.
- A state in the United States, specified by its ISO 3166-2:US code, e.g., `'US-AL'` for Alabama. Note that the resolution option must beset to either `'provinces'` or `'metros'`.

For more information see: http://code.google.com/apis/chart/interactive/docs/gallery/geochart.html#Configuration_Options

`magnifyingGlass` an object. Default

`{enable: true, zoomFactor: 5.0}`

An object with members to configure various aspects of the magnifying glass. To specify properties of this object, you can use object literal notation, as shown here:

`{enable: true, zoomFactor: 7.5}`

`magnifyingGlass.enable` boolean. Default `TRUE`. If true, when the user lingers over a cluttered marker, a magnifiying glass will be opened.

Note: this feature is not supported in browsers that do not support SVG, i.e. Internet Explorer version 8 or earlier.

`magnifyingGlass.zoomFactor` a number. Default 5.0. The zoom factor of the magnifying glass. Can be any number greater than 0.

`markerOpacity` number, between 0.0 - 1.0. Default 1.0.

`resolution` a string. Default `'countries'` The resolution of the map borders. Choose one of the following values:

`'countries'`

`'provinces'` Not supported for all countries; please test a country to see whether this option is supported.

`'metros'` Supported for the US country region and US state regions only.

`sizeAxis` a JSON object. Default `'null'`. An object with members to configure how values are associated with bubble size. To specify properties of this object, you can use object literal notation, as shown here:

`{minValue: 0,  maxSize: 20}`

`sizeAxis.maxSize` a number. Default 30. Maximum size of the largest marker, in pixels.

`sizeAxis.maxValue` a number. Default maximum value of size column in chart data Maximum size column value. Larger values will be cropped to this value.

`sizeAxis.minSize` a number. Default 2. Mininum size of the smallest marker, in pixels.

`sizeAxis.minValue` a number. Default minimum value of size column in chart data Minimum size column value. Smaller values will be clamped to this value.

`tooltip` a JSON object. Default `'null'`. An object with members to configure various tooltip elements. To specify properties of this object, you can use object literal notation, as shown here:

`{textStyle: {color: '#FF0000'}, showColorCode: true}`

`tooltip.textStyle` a JSON object. Default

```
{color:
  'black', fontName: <global-font-name>, fontSize:
  <global-font-size>}
```

An object that specifies the tooltip text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`.

`width` number, default 556. Width of the visualization. If no units are given, the default unit is pixels.

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

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, here 'GeoChart'
`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.
  - `jsChart` Call of the `jsDisplayChart` function.
  - `jsFooter` End tag `</script>`.
  - `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.

Note

This is the new version of the GeoChart chart.

Author(s)

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

Diego de Castillo <`decastillo@gmail.com`>

References

Google Geo Chart API: `https://google-developers.appspot.com/chart/interactive/docs/gallery/geochart.html`

Follow the link for Google's data policy.

See Also

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

Examples

```
## 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

G1 <- gvisGeoChart(Exports, locationvar='Country', colorvar='Profit')

plot(G1)

## 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)

## Markers Example
## A marker style map renders bubble-shaped markers at specified
## locations with the color and size that you specify.

G4 <- gvisGeoChart(CityPopularity, locationvar='City', colorvar='Popularity',
                      options=list(region='US', height=350,
                                   displayMode='markers',
   colorAxis="{values:[200,400,600,800],
                                   colors:[\'red', \'pink\', \'orange',\'green']}")
                      )
plot(G4)

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


G6 <- gvisGeoChart(Andrew, "LatLong", sizevar='Speed_kt',
                   colorvar="Pressure_mb", options=list(region="US"))
plot(G6)

## 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=":")

G7 <- gvisGeoChart(quakes, "latlong", "depth", "mag",
                   options=list(displayMode="Markers", region="009",
                   colorAxis="{colors:['red', 'grey']}",
                   backgroundColor="lightblue"))

plot(G7)

## Not run:
## Plot world wide earth quakes of the last 30 days with magnitude >= 4.0
library(XML)
## Get earthquake data of the last 30 days
eq <- readHTMLTable(readLines("http://www.iris.edu/seismon/last30.html"),
colClasses=c("factor", rep("numeric", 4), "factor"))
eq <- eq[[2]] ## extract the eq table
eq$loc=paste(eq$LAT, eq$LON, sep=":") ## create a lat:long location variable

G8 <- gvisGeoChart(eq, "loc", "DEPTH km", "MAG",
                   options=list(displayMode="Markers",
                   colorAxis="{colors:['purple', 'red', 'orange', 'grey']}",
                   backgroundColor="lightblue"))
plot(G8)

## End(Not run)
```

gvisGeoMap *Google Geo Map with R*

Description

The gvisGeoMap 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.

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

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

Arguments

`data`	a `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 `http://code.google.com/apis/adwords/docs/developer/adwords_api_us_metros.html`. 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 for Google Geo Map. `gvis.editor` a character label for an on-page button which opens an in-page dialog box that enables users to edit, change and customise the chart. By default no value is given and therefore no button is displayed. Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/geomap.html#Configuration_Options`: `region` string, default `'world'`. The area to display on the map. (Surrounding areas will be displayed as well.) Can be either a country code (in uppercase ISO-3166 format), or a one of the following strings

world	(Whole world)
us_metro	(United States, metro areas)
005	(South America)
013	(Central America)
021	(North America)
002	(All of Africa)
017	(Central Africa)
015	(Northern Africa)
018	(Southern Africa)
030	(Eastern Asia)
034	(Southern Asia)
035	(Asia/Pacific region)
143	(Central Asia)
145	(Middle East)
151	(Northern Asia)

154	(Northern Europe)
155	(Western Europe)
039	(Southern Europe)

Geomap does not enable scrolling or dragging behavior, and only limited zooming behavior. A basic zoom out can be enabled by setting the showZoomOut property.

`dataMode` string, default 'regions'. How to display values on the map. Two values are supported:
`regions` - Colors a whole region with the appropriate color. This option cannot be used with latitude/longitude addresses. See Regions Example.
`markers` - Displays a dot over a region, with the color and size indicating the value. See Markers Example.

`width` string, default '556px'. Width of the visualization. If no units are given, the default unit is pixels.

`height` default, string '347px'. Height of the visualization. If no units are given, the default unit is pixels.

`colors` a JSON array of RGB numbers in the format 0xRRGGBB [0xE0FFD4, 0xA5EF63, 0x50AA00, 0x267114]. Color gradient to assign to values in the visualization. You must have at least two values; the gradient will include all your values, plus calculated intermediary values, with the lightest color as the smallest value, and the darkest color as the highest.

`showLegend` boolean, default `TRUE`. If true, display a legend for the map.

`showZoomOut` boolean, default `FALSE`. If true, display a button with the label specified by the zoomOutLabel property. Note that this button does nothing when clicked, except throw the zoomOut event. To handle zooming, catch this event and change the region option. You can only specify `showZoomOut` if the region option is smaller than the world view. One way of enabling zoom in behavior would be to listen for the regionClick event, change the region property to the appropriate region, and reload the map.

`zoomOutLabel` string, default 'Zoom Out'. Label for the zoom button.

`gvis.listener.jscode` character string which will be placed inside select event. A valid value is `alert('a region was selected');`. You may also use the method `getSelection`.

`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, here 'GeoMap'
`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.
  - `jsChart` Call of the `jsDisplayChart` function.
  - `jsFooter` End tag `</script>`.
  - `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. http://www.myhost.com/myviz.html). See the googleVis package vignette and the Macromedia web site (`http://www.macromedia.com/support/documentation/en/flashplayer/help/`) for more details.

Note

A map can display a maximum of 400 entries; if your DataTable or DataView holds more than 400 rows, only the first 400 will be shown. The fastest modes are `dataMode='regions'` with locations specified as ISO codes, and `dataMode='markers'` with lat/long entries. The slowest mode is `dataMode='markers'` with a string address.

`gvisGeoMap` requires Flash, see `gvisGeoChart` for a geo map based on SVG.

Author(s)

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

Diego de Castillo <`decastillo@gmail.com`>

References

Google Geo Map API: `https://google-developers.appspot.com/chart/interactive/docs/gallery/geomap.html`

Follow the link for Google's data policy.

See Also

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

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)

G3 <- gvisGeoMap(states, "state.name", "Illiteracy",
                 options=list(region="US", dataMode="regions",
 width=600, height=400))
plot(G3)

## Example with latitude and longitude information
## Show Hurricane Andrew (1992) storm track
G4 <- gvisGeoMap(Andrew, locationvar="LatLong", numvar="Speed_kt",
        hovervar="Category",
                options=list(height=350, region="US", dataMode="markers"))

plot(G4)

## World population
WorldPopulation=data.frame(Country=Population$Country,
   Population.in.millions=round(Population$Population/1e6,0),
   Rank=paste(Population$Country, "Rank:", Population$Rank))

G5 <- gvisGeoMap(WorldPopulation, "Country", "Population.in.millions", "Rank",
             options=list(dataMode="regions", width=600, height=300))
plot(G5)
```

```
## Not run:
## Plot world wide earth quakes of the last 30 days with magnitude >= 4.0
library(XML)
## Get earthquake data of the last 30 days
eq <- readHTMLTable(readLines("http://www.iris.edu/seismon/last30.html"),
colClasses=c("factor", rep("numeric", 4), "factor"))
eq <- eq[[2]] ## extract the eq table
eq$loc=paste(eq$LAT, eq$LON, sep=":") ## create a lat:long location variable
plot(gvisGeoMap(eq, "loc", "MAG","DATE", options=list(dataMode="markers")))


## The demo 'AnimatedGeoMap' shows how a Geo Map can be animated
## with additional JavaScript.
## Thanks to Manoj Ananthapadmanabhan and Anand Ramalingam, who
## provided the idea and initial code.
## Please note: This demo requires the package 'pscl' for its data set
## 'presidentalElections'.

demo(AnimatedGeoMap)

## End(Not run)
```

`gvisIntensityMap` *Google Intensity Map with R*

Description

An intensity map highlights regions or countries based on relative values.

The gvisIntensityMap 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

```
gvisIntensityMap(data, locationvar = "", numvar = "",
                       options = list(), chartid)
```

Arguments

`data`	a `data.frame`. The data has to have at least two columns with location name (`locationvar`) and any number of numeric columns (`numvar`) to be mapped.
`locationvar`	column name of `data` with the geo locations to be analysed. The location has to contain country ISO codes or USA state codes.
`numvar`	column names of `data` with the numeric values to be displayed.
`options`	list of configuration options for Google Intensity Map.

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

Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/intensitymap.html#Configuration_Options`:

`colors` a JSON array of strings. The colors to use for each tab. An array of strings. Each element is a string in the format #rrggbb. For example `'#00cc00'`.

`height` a number. Default 220. Height of the map in pixels. The maximum height of the visualization is 220. Note that this height assumes a one-row tab. If your tab text is long, it will wrap the tab to multiple lines, and the extra lines will exceed the specified height.

`region` a string. Default `'world'`. The required region. Possible values are: `'world', 'africa', 'asia', 'europe', 'middle_east', 'south_america', and 'usa'.`

`showOneTab` boolean. Default `FALSE`. The intensity map can display one or more numeric columns. Each column is displayed as a separate map, and tabs on top enable selection of which map to show. When the data table contains only one numeric column, the tabs are not displayed. To display tabs even for a single numeric column, set this option to `TRUE`.

`width` a number. Default 440. Width of the map in pixels. Note: The maximum width of the visualization is 440.

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

Value

`gvisIntensityMap` 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 'IntensityMap'
`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.
- `jsChart` Call of the `jsDisplayChart` function.
- `jsFooter` End tag `</script>`.
- `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.

Note

Map images are generated using the Google Charts API (`http://code.google.com/apis/chart/image/docs/gallery/map_charts.html`). Please refer to the Chart API logging policy (`http://code.google.com/apis/chart/interactive/faq.html#logging`).

Author(s)

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

Diego de Castillo <`decastillo@gmail.com`>

References

Google Intensity Map API: `https://google-developers.appspot.com/chart/interactive/docs/gallery/intensitymap.html`

Follow the link for Google's data policy.

See Also

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

Examples

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

df=data.frame(country=c("US", "GB", "BR"), val1=c(1,3,4), val2=c(23,12,32))
Intensity1 <- gvisIntensityMap(df, locationvar="country", numvar=c("val1", "val2"))
plot(Intensity1)

## Set colours for each tab
Intensity2 <- gvisIntensityMap(df,
              options=list(colors="['#4682b4', '#0073CF']"))
plot(Intensity2)
```

gvisLineChart *Google Line Chart with R*

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

```
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 for Google Line Chart.

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

Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/linechart.html#Configuration_Options`:

`axisTitlesPosition` a string. Default 'out'. Where to place the axis titles, compared to the chart area. Supported values:

`'in'` Draw the axis titles inside the the chart area.

`'out'` Draw the axis titles outside the chart area.

`'none'` Omit the axis titles.

`backgroundColor` a string or object. Default `'white'`. The background color for the main area of the chart. Can be either a simple HTML color string, for example: `'red'` or `'#00cc00'`, or an object with the following properties.

`backgroundColor.stroke` a string. Default `'#666'`. The color of the chart border, as an HTML color string.

`backgroundColor.strokeWidth` a number. Default 0. The border width, in pixels.

`backgroundColor.fill` a string. Default `'white'`. The chart fill color, as an HTML color string.

`chartArea` a string. Default `'null'`. An object with members to configure the placement and size of the chart area (where the chart itself is drawn, excluding axis and legends). Two formats are supported: a number, or a number followed by `%`. A simple number is a value in pixels; a number followed by `%` is a percentage. Example:

`{left:20,top:0,width:\"50%\",height:\"75%\"}`

`chartArea.left` a number or string. Default auto. How far to draw the chart from the left border.

`chartArea.top` a number or string. Default auto. How far to draw the chart from the top border.

`chartArea.width` a number or string. Default auto. Chart area width.

`chartArea.height` a number or string. Default auto. Chart area height.

`colors` An array of strings. Default 'colors'. The colors to use for the chart elements. An array of strings, where each element is an HTML color string, for example: `colors:['red','#004411']`.

`curveType` a string. Default `'none'`. Controls the curve of the lines. Can be one of the following:

`'none'` Straight lines without curve.

`'function'` The angles of the line will be smoothed.

`enableInteractivity` boolean. Default `TRUE`. Whether the chart throws user-based events or reacts to user interaction. If false, the chart will not throw 'select' or other interaction-based events (but will throw ready or error events), and will not display hovertext or otherwise change depending on user input.

`focusTarget` a string. Default `'datum'`. The type of the entity that receives focus on mouse hover. Also affects which entity is selected by mouse click, and which data table element is associated with events. Can be one of the following:

`'datum'` Focus on a single data point. Correlates to a cell in the data.

`'category'` Focus on a grouping of all data points along the major axis. Correlates to a row in the data table.

In focusTarget `'category'` the tooltip displays all the category values. This may be useful for comparing values of different series.

`fontSize` a number. Default automatic. The default font size, in pixels, of all text in the chart. You can override this using properties for specific chart elements.

`fontName` a string. Default `'Arial'`. The default font face for all text in the chart. You can override this using properties for specific chart elements.

`hAxis` a JSON object. Default `'null'`. An object with members to configure various horizontal axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`hAxis.baseline` a number. Default automatic. The baseline for the horizontal axis. This option is only supported for a continuous axis.

`hAxis.baselineColor` a string. Default `'black'`. The color of the baseline for the horizontal axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`. This option is only supported for a continuous axis.

`hAxis.direction` 1 or -1. Default 1. The direction in which the values along the vertical axis grow. Specify -1 to reverse the order of the values.

`hAxis.format` a string. Default auto. A format string for numeric or date axis labels.

For number axis labels, this is a subset of the decimal formatting ICU pattern set. For instance,

`{format:'#,###%'}.`

will display values \code"1,000%", "750%", and `"50%"` for values 10, 7.5, and 0.5.

For date axis labels, this is a subset of the date formatting ICU pattern set. For instance,

`{format:'MMM d, y'}.`

will display the value "Jul 1, 2011" for the date of July first in 2011.

The actual formatting applied to the label is derived from the locale the API has been loaded with. For more details, see loading charts with a specific locale.

This option is only supported for a continuous axis.

`hAxis.gridlines` a JSON object. Default null. An object with members to configure the gridlines on the horizontal axis. To specify properties of this object, you can use object literal notation, as shown here:

`{color: '#333', count: 4}`

This option is only supported for a continuous axis.

`hAxis.gridlines.color` a string. Default `'#CCC'`. The color of the vertical gridlines inside the chart area. Specify a valid HTML color string.

`hAxis.gridlines.count` a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

`hAxis.logScale` boolean. Default `FALSE`. `vAxis` property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to `TRUE` for yes. This option is only supported for a continuous axis.

`hAxis.textPosition` a string. Default `'out'` Position of the horizontal axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`hAxis.textStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}`

An object that specifies the horizontal axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.title` a string. Default `'null'`. `hAxis` property that specifies the title of the horizontal axis.

`hAxis.titleTextStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}.`

An object that specifies the horizontal axis title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.slantedText` boolean. Default automatic. If true, draw the horizontal axis text at an angle, to help fit more text along the axis; if false, draw horizontal axis text upright. Default behavior is to slant text if it cannot all fit when drawn upright. Notice that this option is available only when the `hAxis.textPosition` is set to `'out'` (which is the default).
This option is only supported for a discrete axis.

`hAxis.slantedTextAngle` a number, 1-90. Default 30. The angle of the horizontal axis text, if it's drawn slanted. Ignored if `hAxis.slantedText` is false, or is in auto mode, and the chart decided to draw the text horizontally.
This option is only supported for a discrete axis.

`hAxis.maxAlternation` a number. Default 2. Maximum number of levels of horizontal axis text. If axis text labels become too crowded, the server might shift neighboring labels up or down in order to fit labels closer together. This value specifies the most number of levels to use; the server can use fewer levels, if labels can fit without overlapping.
This option is only supported for a discrete axis.

`hAxis.showTextEvery` a number. Default automatic. How many horizontal axis labels to show, where 1 means show every label, 2 means show every other label, and so on. Default is to try to show as many labels as possible without overlapping.
This option is only supported for a discrete axis.

`hAxis.maxValue` a number. Default automatic. `hAxis` property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.
This option is only supported for a continuous axis.

`hAxis.minValue` a number. Default automatic. `hAxis` property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.
This option is only supported for a continuous axis.

`hAxis.viewWindowMode` a string. Default `"pretty"` if `hAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the horizontal axis to render the values within the chart area. The following string values are supported:

`'pretty'` Scale the horizontal values so that the maximum and minimum data values are rendered a bit inside the left and right of the chart area.

`'maximized'` Scale the horizontal values so that the maximum and minimum data values touch the left and right of the chart area.

`'explicit'` Specify the left and right scale values of the chart area. Data values outside these values will be cropped. You must specify a `hAxis.viewWindow` object describing the maximum and minimum values to show.

This option is only supported for a continuous axis.

`hAxis.viewWindow` Object. Default `NULL`. Specifies the cropping range of the horizontal axis.

`hAxis.viewWindow.max` A number. Default auto.

For a continuous axis The maximum horizontal data value to render. Has an effect only if `hAxis.viewWindowMode='explicit'`.

For a discrete axis The zero-based row index where the cropping window ends. Data points at this index and higher will be cropped out. In conjunction with `vAxis.viewWindowMode.min`, it defines a half-opened range `[min, max)` that denotes the element indices to display. In other words, every index such that `min <= index < max` will be displayed.

`hAxis.viewWindow.min` A number. Default auto.

For a continuous axis The minimum horizontal data value to render. Has an effect only if `hAxis.viewWindowMode='explicit'`.

For a discrete axis The zero-based row index where the cropping window begins. Data points at indices lower than this will be cropped out. In conjunction with `vAxis.viewWindowMode.max`, it defines a half-opened range `[min, max)` that denotes the element indices to display. In other words, every index such that `min <= index < max` will be displayed.

`height` a number. Default height of the containing element. Height of the chart, in pixels.

`interpolateNulls` boolean. Default `FALSE`. Whether to guess the value of missing points. If true, it will guess the value of any missing data based on neighbouring points. If false, it will leave a break in the line at the unknown point.

`legend` a JSON object. Default `NULL`. An object with members to configure various aspects of the legend. To specify properties of this object, you can use object literal notation, as shown here:

```
{position: 'top', textStyle: {color: 'blue', fontSize: 16}}
```

`legend.position` a string. Default `'right'`. Position of the legend. Can be one of the following:

`'right'` To the right of the chart.

`'top'` Above the chart.

`'bottom'` Below the chart.

`'none'` No legend is displayed.

`legend.textStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the legend text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`lineWidth` a number. Default 2. Line width in pixels. Use zero to hide all lines and show only the points.

`pointSize` a number. Default 0. Diameter of data points, in pixels. Use zero to hide all points.

`reverseCategories` boolean. Default `FALSE`. If set to true, will draw series from right to left. The default is to draw left-to-right.

`series` a JSON array of objects, or object with nested objects. Default {}. An array of objects, each describing the format of the corresponding series in the chart. To use default values for a series, specify an empty object . If a series or a value is not specified, the global value will be used. Each object supports the following properties:

- `color` The color to use for this series. Specify a valid HTML color string.
- `targetAxisIndex` Which axis to assign this series to, where 0 is the default axis, and 1 is the opposite axis. Default value is 0; set to 1 to define a chart where different series are rendered against different axes. You can define a different scale for different axes.
- `pointSize` Overrides the global pointSize value for this series.
- `lineWidth` Overrides the global lineWidth value for this series.
- `curveType` Overrides the global curveType value for this series.
- `visibleInLegend` A boolean value, where true means that the series should have a legend entry, and false means that it should not. Default is `TRUE`.

You can specify either an array of objects, each of which applies to the series in the order given, or you can specify an object where each child has a numeric key indicating which series it applies to. For example, the following two declarations are identical, and declare the first series as black and absent from the legend, and the fourth as red and absent from the legend:

```
series: [{color: 'black', visibleInLegend: false},{}, {}, {color:
'red', visibleInLegend: false}]

series: {0:{color: 'black', visibleInLegend: false}, 3:{color: 'red',
visibleInLegend: false}}
```

`theme` a string. Default `NULL`. A theme is a set of predefined option values that work together to achieve a specific chart behavior or visual effect. Currently only one theme is available:

- `maximized` Maximizes the area of the chart, and draws the legend and all of the labels inside the chart area. Sets the following options:

```
chartArea: {width: '100%', height: '100%'},
legend: {position: 'in'},
titlePosition: 'in', axisTitlesPosition: 'in',
hAxis: {textPosition: 'in'}, vAxis: {textPosition: 'in'}
```

`title` a string. Default no title. Text to display above the chart.

`titlePosition` a string. Default `'out'`. Where to place the chart title, compared to the chart area. Supported values:

- `'in'` Draw the title inside the chart area.
- `'out'` Draw the title outside the chart area.
- `'none'` Omit the title.

`titleTextStyle` a JSON object. Default

`{color:'black', fontName:<global-font-name>,fontSize:<global-font-size>}`.

An object that specifies the title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip` a JSON object. Default `NULL`. An object with members to configure various tooltip elements. To specify properties of this object, you can use object literal notation, as shown here:

`{textStyle: {color: '#FF0000'}, showColorCode: true}`

`tooltip.showColorCode` boolean. Default automatic. If true, show colored squares next to the series information in the tooltip. The default is true when `focusTarget` is set to `'category'`, otherwise the default is `FALSE`.

`tooltip.TextStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the tooltip text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip.trigger` The user interaction that causes the tooltip to be displayed:

- `'hover'` The tooltip will be displayed when the user hovers over an element.
- `'none'` The tooltip will not be displayed.

`vAxes` a JSON array of objects, or object with child objects null. Specifies properties for individual vertical axes, if the chart has multiple vertical axes. Each child object is a `vAxis` object, and can contain all the properties supported by `vAxis`. These property values override any global settings for the same property.

To specify a chart with multiple vertical axes, first define a new axis using `series.targetAxisIndex`, then configure the axis using `vAxes`. The following example assigns series 2 to the right axis and specifies a custom title and text style for it:

```
series:{2:{targetAxisIndex:1}},
  vAxes:{1:{title:'Losses',textStyle:{color: 'red'}}}
```

This property can be either an object or an array: the object is a collection of objects, each with a numeric label that specifies the axis that it defines–this is the format shown above; the array is an array of objects, one per axis. For example, the following array-style notation is identical to the `vAxis` object shown above:

```
vAxes:[
{}, // Nothing specified for axis 0
{title:'Losses',textStyle:{color: 'red'}} // Axis 1
]
```

`vAxis` a JSON object. Default `'null'`. An object with members to configure various vertical axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`vAxis.baseline` a number. Default automatic. `vAxis` property that specifies the baseline for the vertical axis. If the baseline is smaller than the highest grid line or smaller than the lowest grid line, it will be rounded tothe closest gridline.

`vAxis.baselineColor` a string. Default `'black'`. `vAxis` property that specifies the color of the baseline for the vertical axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`.

`vAxis.direction` 1 or -1. Default 1. The direction in which the values along the vertical axis grow. Specify -1 to reverse the order of the values.

`vAxis.format` a string. Default auto. A format string for numeric axis labels. This is a subset of the ICU pattern set. For instance,

`{format:'#,###%'}`.

will display values 1,000%, 750%, and 50% for values 10, 7.5, and 0.5.

`vAxis.gridlines` a JSON object. Default `NULL`. An object with members to configure the gridlines on the vertical axis. To specify properties of this object, you can use object literal notation, as shown here:

`{color: '#333', count: 4}`

`vAxis.gridlines.color` a string. Default `'#CCC'`. The color of the vertical gridlines inside the chart area. Specify a valid HTML color string.

`vAxis.gridlines.count` a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

`vAxis.logScale` boolean. Default `FALSE`. `vAxis` property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to `TRUE` for yes.

`vAxis.textPosition` a string. Default `'out'`. Position of the vertical axis text, relative to the chart area. Supported values: `'out'`, `'in'`, `'none'`.

`vAxis.textStyle` a JSON object. Default

`{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`.

An object that specifies the vertical axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.title` a string. Default no title. `vAxis` property that specifies a title for the vertical axis.

`vAxis.titleTextStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}`.

An object that specifies the vertical axis title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.maxValue` a number. Default automatic. vAxis property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.

`vAxis.minValue` a number. Default automatic. `vAxis` property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.

`vAxis.viewWindowMode` a string. Default `"pretty"` if `vAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the vertical axis to render the values within the chart area. The following string values are supported:

- `'pretty'` Scale the vertical values so that the maximum and minimum data values are rendered a bit inside the top and bottom of the chart area.
- `'maximized'` Scale the vertical values so that the maximum and minimum data values touch the top and bottom of the chart area.
- `'explicit'` Specify the top and bottom scale values of the chart area. Data values outside these values will be cropped. You must specify a `vAxis.viewWindow` object describing the maximum and minimum values to show.

`vAxis.viewWindow` Object. Default `NULL`. Specifies the maximum and minimum data values to show on the vertical axis. Present only if `vAxis.viewWindowMode='explicit'`

`vAxis.viewWindow.max` A number. Default 0. The maximum vertical data value to render.

`vAxis.viewWindow.min` A number. Default 0. The minimum vertical data value to render.

`width` a number. Default width of the containing element. Width of the chart, in pixels.

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

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, here 'LineChart'

`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.

`jsChart` Call of the `jsDisplayChart` function.

`jsFooter` End tag `</script>`.

`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

Google Visualisation API: You cannot load both linechart and corechart packages at the same time on the same page.

Author(s)

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

Diego de Castillo `<decastillo@gmail.com>`

References

Google Line Chart API: http://code.google.com/apis/chart/interactive/docs/gallery/linechart.html

Follow the link for Google's data policy.

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(vAxis="{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)
```

`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.

The maps are the well known Google Maps.

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.

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

Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/map.html#Configuration_Options`:

`enableScrollWheel` boolean. Default `FALSE`. If set to `TRUE`, enables zooming in and out using the mouse scroll wheel.

`showTip` boolean. Default `FALSE`. If set to `TRUE`, shows the location description as a tool-tip when the mouse is positioned above a point marker.

`showLine` boolean. Default `FALSE`. If set to `TRUE`, shows a Google Maps polyline through all the points.

`lineColor` string default color. If showLine is `TRUE`, defines the line color. For example: '#800000'.

`lineWidth` number. Default 10. If showLine is true, defines the line width (in pixels).

`mapType` string. Default: 'hybrid'. The type of map to show. Possible values are 'normal', 'terrain', 'satellite' or 'hybrid'.

`useMapTypeControl` boolean. Default `FALSE`. Show a map type selector that enables the viewer to switch between [map, satellite, hybrid, terrain]. When `useMapTypeControl` is `FALSE` (default) no selector is presented and the type is determined by the `mapType` option.

`zoomLevel` number/ Default automatic. An integer indicating the initial zoom level of the map, where 0 is completely zoomed out (whole world) and 19 is the maximum zoom level. (See "Zoom Levels" in the Google Maps API.)

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

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, here 'Map'
`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.

`jsChart` Call of the `jsDisplayChart` function.

`jsFooter` End tag `</script>`.

`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.

Note

The Lat-Long pairs option loads maps much faster, especially with large data. We recommend that you use this option for large data sets. Please visit Google Maps API to find out how to transform your addresses to lat-long points. The map can display a maximum of 400 entries; if your data holds more than 400 rows, only the first 400 will be shown.

Author(s)

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

Diego de Castillo `<decastillo@gmail.com>`

References

Google Map API: `https://google-developers.appspot.com/chart/interactive/docs/gallery/map.html`

Follow the link for Google's data policy.

See Also

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

Further the packages:

- R2GoogleMaps: Provides a mechanism to generate JavaScript code from R that displays data using Google Maps, `http://www.omegahat.org/R2GoogleMaps/`.
- RgoogleMaps: Overlays on Google map tiles in R, `http://cran.r-project.org/web/packages/RgoogleMaps/index.html`,
- plotGoogleMaps: Plot HTML output with Google Maps API and your own data, `http://cran.r-project.org/web/packages/plotGoogleMaps/`.

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

df <- data.frame(Postcode=c("EC3M 7HA", "EC2P 2EJ"),
                 Tip=c("Lloyd's", "Guildhall"))

M2 <- gvisMap(df, "Postcode", "Tip",
              options=list(showTip=TRUE, mapType='normal',
              enableScrollWheel=TRUE))

plot(M2)
```

---

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.
  - `jsChart` Call of the `jsDisplayChart` function.
  - `jsFooter` End tag `</script>`.
  - `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: `http://code.google.com/apis/chart/index.html`

Follow the link for Google's data policy.

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.


G <- gvisGeoChart(Exports, "Country", "Profit",
                  options=list(width=200, height=100))
T <- gvisTable(Exports,
                  options=list(width=200, height=260))

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

M <- gvisMotionChart(Fruits, "Fruit", "Year",
                     options=list(width=400, height=360))

GTM <- gvisMerge(GT, M, horizontal=TRUE,
                     tableOptions="bgcolor=\"#CCCCCC\" 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), TRUE)

plot(LBCA)


plot(gvisMerge(GTM, LBCA, tableOptions="bgcolor=\"#AABBCC\""))

## 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

```
gvisMotionChart(data, idvar = "id", timevar = "time",
      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.
`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).
`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. `gvis.editor` a character label for an on-page button which opens an in-page dialog box that enables users to edit, change and customise the chart. By default no value is given and therefore no button is displayed. `gvis.language` values may be 'ca', 'da', 'de', 'en', 'en_GB', 'en_IE', 'es', 'es_419', 'fi', 'fr', 'id', 'in', 'is', 'it', 'nl', 'no', 'pt', 'pt_BR', 'pt_PT', 'sv'. If not set the API detects the language settings of the browser. Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/motionchart.html#Configuration_Options`: `height` height of the chart in pixels. `width` width of the chart in pixels. `state` An initial display state for the chart. This is a serialised JSON object that describes zoom level, selected dimensions, selected bubbles/entities, and other state descriptions. For more details see `https://google-developers.appspot.com/chart/interactive/docs/gallery/motionchart.html#Motion_Chart_initial_state` `showChartButtons` logical, default=`TRUE`. `FALSE` hides the buttons that control the chart type (bubbles / lines / columns) at top right corner. `showHeader` logical, default=`TRUE`. `FALSE` hides the title label of the entities (derived from the label of the first column in the data table). `showSelectListComponent` logical, default=`TRUE`. `FALSE` hides the list of visible entities. `showSidePanel` logical, default=`TRUE`. `FALSE` hides the right hand panel.

- `showXMetricPicker` logical, default=`TRUE`. `FALSE` hides the metric picker for x.
- `showYMetricPicker` logical, default=`TRUE`. `FALSE` hides the metric picker for y.
- `showXScalePicker` logical, default=`TRUE`. `FALSE` hides the scale picker for x.
- `showYScalePicker` logical, default=`TRUE`. `FALSE` hides the scale picker for y.
- `showAdvancedPanel` logical, default=`TRUE`. `FALSE` hides disables the options compartment in the settings panel.

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

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, here '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.
  - `jsChart` Call of the `jsDisplayChart` function.
  - `jsFooter` End tag `</script>`.
  - `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. http://www.myhost.com/myviz.html). See the googleVis package vignette and the Macromedia web site (`http://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 Motion Chart API: https://google-developers.appspot.com/chart/interactive/docs/gallery/motionchart.html

Follow the link for Google's data policy.

In 2006 Hans Rosling gave an inspiring talk at TED http://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen.html about social and economic developments in the world over the last 50 years, which challenged the views and perceptions of many listeners. Rosling had used extensive data analysis to reach his conclusions. To visualise his talk, he and his team at Gapminder http://www.gapminder.org had developed animated bubble charts, aka motion charts.

In March 2007 Google acquired Trendalyzer from the Gapminder Foundation and the Gapminder team of developers joined Google in California in April 2007.

Yihui Xie, the maintainer of the animation-package showed in an early blog entry the usage of the Google Motion Chart API with R:

http://yihui.name/en/2008/11/brownian-motion-using-google-visualization-api-and-r/

Further examples of displaying data with motion charts are available via the Google Public Data Explorer: http://www.google.com/publicdata/home.

Stephen Thompson at Lloyd's developed an Excel version that mimics much of the functionality of the Google motion charts: http://www.lloyds.com/The-Market/Tools-and-Resources/Resources/Statistics-Relating-to-Lloyds/Visualisation

See Also

See also print.gvis, plot.gvis for printing and plotting methods, and also the animation-package and its function Rosling.bubbles.

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.

Fruits
## timevar Year
M1 <- gvisMotionChart(Fruits, idvar="Fruit", timevar="Year")
str(M1)
## print.gvis, will concatenate the list into a one long string
M1

plot(M1)

## Combine with another chart, e.g. table
```

```
tbl <- gvisTable(Fruits, options=list(height=220))
Mtbl <- gvisMerge(M1, tbl)
plot(Mtbl)

## 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.

myStateSettings <-'
{"xZoomedDataMin":1199145600000,"colorOption":"2",
"duration":{"timeUnit":"Y","multiplier":1},"yLambda":1,
"yAxisOption":"4","sizeOption":"_UNISIZE",
"iconKeySettings":[],"xLambda":1,"nonSelectedAlpha":0,
"xZoomedDataMax":1262304000000,"iconType":"LINE",
"dimensions":{"iconDimensions":["dim0"]},
"showTrails":false,"uniColorForNonSelected":false,
"xAxisOption":"_TIME","orderedByX":false,"playDuration":15000,
"xZoomedIn":false,"time":"2010","yZoomedDataMin":0,
"yZoomedIn":false,"orderedByY":false,"yZoomedDataMax":100}
'
M6 <- gvisMotionChart(Fruits, "Fruit", "Year", options=list(state=myStateSettings))
plot(M6)
## For more information see:
## http://code.google.com/apis/chart/interactive/docs/gallery/motionchart.html#Motion_Chart_initial_state

## See also the demo(WorldBank). It demonstrates how you can access
## country level data from the World Bank to create Gapminder-like
## plots.
```

`gvisOrgChart` *Google Org Chart with R*

Description

An organizational chart that supports selection.

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.

Usage

```
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 for Google Org Chart. `gvis.editor` a character label for an on-page button which opens an in-page dialog box that enables users to edit, change and customise the chart. By default no value is given and therefore no button is displayed. Further possible components are, taken from https://google-developers.appspot.com/chart/interactive/docs/gallery/orgchart.html#Configuration_Options: `allowCollapse` Boolean. Default `FALSE`. Determines if double click will collapse a node. `allowHtml` Boolean. Default `FALSE`. If set to `TRUE`, names that includes HTML tags will be rendered as HTML.

- `nodeClass` A string. Default default class name. A class name to assign to node elements. Apply CSS to this class name to specify colors or styles for the chart elements.
- `selectedNodeClass` A string. Default class name. A class name to assign to selected node elements. Apply CSS to this class name to specify colors or styles for selected chart elements.

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

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, here 'OrgChart'

`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.
  - `jsChart` Call of the `jsDisplayChart` function.
  - `jsFooter` End tag `</script>`.
  - `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 Org Chart API: `https://google-developers.appspot.com/chart/interactive/docs/gallery/orgchart.html`

Follow the link for Google's data policy.

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

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

Further possible components are, taken from https://google-developers.appspot.com/chart/interactive/docs/gallery/piechart.html#Configuration_Options:

`backgroundColor` a string or object. Default `'white'`. The background color for the main area of the chart. Can be either a simple HTML color string, for example: `'red'` or `'#00cc00'`, or an object with the following properties.

`backgroundColor.stroke` a string. Default `'#666'`. The color of the chart border, as an HTML color string.

`backgroundColor.strokeWidth` a number. Default 0. The border width, in pixels.

`backgroundColor.fill` a string. Default `'white'`. The chart fill color, as an HTML color string.

`chartArea` A string. Default `'null'`. An object with members to configure the placement and size of the chart area (where the chart itself is drawn, excluding axis and legends). Two formats are supported: a number, or a number followed by `%`. A simple number is a value in pixels; a number followed by `%` is a percentage. Example:

`{left:20,top:0,width:\"50%\",height:\"75%\"}`

`chartArea.left` A number or string. Default auto. How far to draw the chart from the left border.

`chartArea.top` A number or string. Default auto. How far to draw the chart from the top border.

`chartArea.width` A number or string. Default auto. Chart area width.

`chartArea.height` A number or string. Default auto. Chart area height.

`colors` An array of strings. Default 'colors'. The colors to use for the chart elements. An array of strings, where each element is an HTML color string, for example: `colors:['red','#004411']`.

`fontSize` A number. Default automatic. The default font size, in pixels, of all text in the chart. You can override this using properties for specific chart elements.

`fontName` A string. Default `'Arial'`. The default font face for all text in the chart. You can override this using properties for specific chart elements.

`height` A number. Default height of the containing element. Height of the chart, in pixels.

`is3D` Boolean. Default `FALSE`. If set to `TRUE`, displays a three-dimensional chart.

`legend` a JSON object. Default `NULL`. An object with members to configure various aspects of the legend. To specify properties of this object, you can use object literal notation, as shown here:

`{position: 'top', textStyle: {color: 'blue', fontSize: 16}}`

`legend.position` a string. Default `'right'`. Position of the legend. Can be one of the following:

`'right'` To the right of the chart.

`'top'` Above the chart.

`'bottom'` Below the chart.

`'none'` No legend is displayed.

`legend.textStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the legend text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`slices` a JOSN array of objects, or object with nested objects. Default

`{}`.

A JSON array of objects, each describing the format of the corresponding slice in the pie. To use default values for a slice, specify an empty object

`{}`.

If a slice or a value is not specified, the global value will be used. Each object supports the following properties:

`color` The color to use for this slice. Specify a valid HTML color string.

`textStyle` Overrides the global pieSliceTextSlice for this slice.

You can specify either an array of objects, each of which applies to the slice in the order given, or you can specify an object where each child has a numeric key indicating which slice it applies to. For example, the following two declarations are identical, and declare the first slice as black and the fourth as red:

```
  slices: [{color: 'black', {}, {}, {color: 'red'}]
  slices: {0: {color: 'black'}, 3: {color: 'red'}}
```

`pieSliceBorderColor` a string. Default `'white'`. The color of the slices border.

`pieSliceText` A string. Default 'percentage'. The content of the text displayed on the slice. Can be one of the following:

`'percentage'` The percentage of the slice size out of the total.

`'value'` The quantitative value of the slice.

`'label'` The name of the slice.

`'none'` No text is displayed.

`pieSliceTextStyle` A json object. Default

```
{color: 'black', fontName: <global-font-name>, fontSize:
  <global-font-size>}.
```

An object that specifies the slice text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}.
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`reverseCategories` Boolean. Default `FALSE`. If set to `TRUE`, will draw slices counterclockwise. The default is to draw clockwise.

`sliceVisibilityThreshold` A number. Default 1/720. The slice relative part, below which a slice will not show individually. All slices that have not passed this threshold will be combined to a single slice, whose size is the sum of all their sizes. Default is not to show individually any slice which is smaller than half a degree.

`pieResidueSliceColor` A string. Default `'#ccc'`. Color for the combination slice that holds all slices below `sliceVisibilityThreshold`.

`pieResidueSliceLabel` A string. Default `'Other'`. A label for the combination slice that holds all slices below `sliceVisibilityThreshold`.

`title` A string. Default no title. Text to display above the chart.

`titleTextStyle` A json object. Default

`{color:'black', fontName:<global-font-name>,fontSize:<global-font-size>}`.

An object that specifies the title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip` a JSON object. Default `NULL`. An object with members to configure various tooltip elements. To specify properties of this object, you can use object literal notation, as shown here:

`{textStyle: {color: '#FF0000'}, showColorCode: true}`

`tooltip.showColorCode` boolean. Default automatic. If true, show colored squares next to the series information in the tooltip. The default is true when `focusTarget` is set to `'category'`, otherwise the default is `FALSE`.

`tooltip.textStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}`

An object that specifies the tooltip text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip.trigger` The user interaction that causes the tooltip to be displayed:

`'hover'` The tooltip will be displayed when the user hovers over an element.

`'none'` The tooltip will not be displayed.

`width` A number. Default width of the containing element. Width of the chart, in pixels.

`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, here 'PieChart'

`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.
  - `jsChart` Call of the `jsDisplayChart` function.
  - `jsFooter` End tag `</script>`.
  - `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

Google Visualisation API: You cannot load both piechart and corechart packages at the same time on the same page.

Author(s)

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

Diego de Castillo <decastillo@gmail.com>

References

Google Pie Chart API: http://code.google.com/apis/chart/interactive/docs/gallery/piechart.html

Follow the link for Google's data policy.

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.

Pie1 <- gvisPieChart(CityPopularity)
plot(Pie1)
```

gvisScatterChart *Google Scatter Chart with R*

Description

The 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.

`options` list of configuration options for Google Scatter Chart.

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

Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/scatterchart.html#Configuration_Options`:

`axisTitlesPosition` a string. Default 'out'. Where to place the axis titles, compared to the chart area. Supported values:

`'in'` Draw the axis titles inside the the chart area.

`'out'` Draw the axis titles outside the chart area.

`'none'` Omit the axis titles.

`backgroundColor` a string or object. Default `'white'`. The background color for the main area of the chart. Can be either a simple HTML color string, for example: `'red'` or `'#00cc00'`, or an object with the following properties.

`backgroundColor.stroke` a string. Default `'#666'`. The color of the chart border, as an HTML color string.

`backgroundColor.strokeWidth` a number. Default 0. The border width, in pixels.

`backgroundColor.fill` a string. Default `'white'`. The chart fill color, as an HTML color string.

`chartArea` a string. Default `'null'`. An object with members to configure the placement and size of the chart area (where the chart itself is drawn, excluding axis and legends). Two formats are supported: a number, or a number followed by `%`. A simple number is a value in pixels; a number followed by `%` is a percentage. Example:

`{left:20,top:0,width:\"50%\",height:\"75%\"}`

`chartArea.left` a number or string. Default auto. How far to draw the chart from the left border.

`chartArea.top` a number or string. Default auto. How far to draw the chart from the top border.

`chartArea.width` a number or string. Default auto. Chart area width.

`chartArea.height` a number or string. Default auto. Chart area height.

`colors` a JSON array of strings. Default 'colors'. The colors to use for the chart elements. An array of strings, where each element is an HTML color string, for example: `colors:['red','#004411']`.

`curveType` a string. Default `'none'`. Controls the curve of the lines. Can be one of the following:

`'none'` Straight lines without curve.

`'function'` The angles of the line will be smoothed.

`enableInteractivity` boolean. Default `TRUE`. Whether the chart throws user-based events or reacts to user interaction. If false, the chart will not throw 'select' or other interaction-based events (but will throw ready or error events), and will not display hovertext or otherwise change depending on user input.

`fontSize` a number. Default automatic. The default font size, in pixels, of all text in the chart. You can override this using properties for specific chart elements.

`fontName` a string. Default `'Arial'`. The default font face for all text in the chart. You can override this using properties for specific chart elements.

`gridlineColor` a string. Default `'#CCC'`. The color of the gridlines inside the chart area. Specify a valid HTML color string.

`hAxis` a JSON object. Default `'null'`. An object with members to configure various horizontal axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`hAxis.baseline` a number. Default automatic. `hAxis` property that specifies the baseline for the horizontal axis. If the baseline is smaller than the highest grid line or smaller than the lowest grid line, it will be rounded to the closest gridline.

`hAxis.baselineColor` a string. Default 'black'. `hAxis` property that specifies the color of the baseline for the horizontal axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`.

`hAxis.direction` 1 or -1. Default 1. The direction in which the values along the horizontal axis grow. Specify -1 to reverse the order of the values.

`hAxis.format` a string. Default auto. a format string for numeric axis labels. This is a subset of the ICU pattern set. For instance,

`{format:'#,###%'}`.

will display values 1,000%, 750%, and 50% for values 10, 7.5, and 0.5.

`hAxis.gridlines` a JSON object. Default null. An object with members to configure the gridlines on the horizontal axis. To specify properties of this object, you can use object literal notation, as shown here:

{color: '#333', count: 4}

This option is only supported for a continuous axis.

`hAxis.gridlines.color` a string. Default `'#CCC'`. The color of the horizontal gridlines inside the chart area. Specify a valid HTML color string.

`hAxis.gridlines.count` a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

`hAxis.logScale` boolean. Default `FALSE`. `vAxis` property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to `TRUE` for yes.

`hAxis.textPosition` a string. Default `'out'` Position of the horizontal axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`hAxis.textStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}`

An object that specifies the horizontal axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.title` a string. Default `'null'`. `hAxis` property that specifies the title of the horizontal axis.

`hAxis.titleTextStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}`.

An object that specifies the horizontal axis title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.maxValue` a number. Default automatic. `hAxis` property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.

`hAxis.minValue` a number. Default automatic. `hAxis` property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.

`hAxis.viewWindowMode` a string. Default `"pretty"` if `hAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the horizontal axis to render the values within the chart area. The following string values are supported:

- `'pretty'` Scale the horizontal values so that the maximum and minimum data values are rendered a bit inside the left and right of the chart area.
- `'maximized'` Scale the horizontal values so that the maximum and minimum data values touch the left and right of the chart area.

'explicit' Specify the left and right scale values of the chart area. Data values outside these values will be cropped. You must specify a hAxis.viewWindow object describing the maximum and minimum values to show.

hAxis.viewWindow Object. Default NULL. Specifies the maximum and minimum data values to show on the horizontal axis. Present only if vAxis.viewWindowMode='explicit

hAxis.viewWindow.max A number. Default 0. The maximum vertical data value to render.

hAxis.viewWindow.min A number. Default 0. The minimum vertical data value to render.

height a number. Default height of the containing element. Height of the chart, in pixels.

legend a JSON object. Default NULL. An object with members to configure various aspects of the legend. To specify properties of this object, you can use object literal notation, as shown here:

```
{position: 'top', textStyle: {color: 'blue', fontSize: 16}}
```

legend.position a string. Default 'right'. Position of the legend. Can be one of the following:

'right' To the right of the chart.

'top' Above the chart.

'bottom' Below the chart.

'none' No legend is displayed.

legend.textStyle a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the legend text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: 'red' or '#00cc00'. Also see fontName and fontSize.

lineWidth a number. Default 2. Line width in pixels. Use zero to hide all lines and show only the points.

pointSize a number. Default 0. Diameter of data points, in pixels. Use zero to hide all points.

series a JSON array of objects, or object with nested objects. Default {}. An array of objects, each describing the format of the corresponding series in the chart. To use default values for a series, specify an empty object . If a series or a value is not specified, the global value will be used. Each object supports the following properties:

color The color to use for this series. Specify a valid HTML color string.

targetAxisIndex Which axis to assign this series to, where 0 is the default axis, and 1 is the opposite axis. Default value is 0; set to 1 to define a chart where different series are rendered against different axes. You can define a different scale for different axes.

pointSize Overrides the global pointSize value for this series.

lineWidth Overrides the global lineWidth value for this series.

`curveType` Overrides the global curveType value for this series.

`visibleInLegend` A boolean value, where true means that the series should have a legend entry, and false means that it should not. Default is `TRUE`.

You can specify either an array of objects, each of which applies to the series in the order given, or you can specify an object where each child has a numeric key indicating which series it applies to. For example, the following two declarations are identical, and declare the first series as black and absent from the legend, and the fourth as red and absent from the legend:

```
series: [{color: 'black', visibleInLegend: false},{}, {}, {color:
'red', visibleInLegend: false}]

series: {0:{color: 'black', visibleInLegend: false}, 3:{color: 'red',
visibleInLegend: false}}
```

`theme` a string. Default `NULL`. A theme is a set of predefined option values that work together to achieve a specific chart behavior or visual effect. Currently only one theme is available:

`maximized` Maximizes the area of the chart, and draws the legend and all of the labels inside the chart area. Sets the following options:

```
chartArea: {width: '100%', height: '100%'},
legend: {position: 'in'},
titlePosition: 'in', axisTitlesPosition: 'in',
hAxis: {textPosition: 'in'}, vAxis: {textPosition: 'in'}
```

`title` a string. Default no title. Text to display above the chart.

`titlePosition` a string. Default `'out'`. Where to place the chart title, compared to the chart area. Supported values:

`'in'` Draw the title inside the chart area.

`'out'` Draw the title outside the chart area.

`'none'` Omit the title.

`titleTextStyle` a JSON object. Default

`{color:'black', fontName:<global-font-name>,fontSize:<global-font-size>}`.

An object that specifies the title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip` a JSON object. Default `NULL`. An object with members to configure various tooltip elements. To specify properties of this object, you can use object literal notation, as shown here:

`{textStyle: {color: '#FF0000'}, showColorCode: true}`

`tooltip.showColorCode` boolean. Default automatic. If true, show colored squares next to the series information in the tooltip. The default is true when `focusTarget` is set to `'category'`, otherwise the default is `FALSE`.

`tooltip.textStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}`

An object that specifies the tooltip text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip.trigger` The user interaction that causes the tooltip to be displayed:

`'hover'` The tooltip will be displayed when the user hovers over an element.

`'none'` The tooltip will not be displayed.

`vAxis` a JSON object. Default `'null'`. An object with members to configure various vertical axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`vAxis.baseline` a number. Default automatic. `vAxis` property that specifies the baseline for the vertical axis. If the baseline is smaller than the highest grid line or smaller than the lowest grid line, it will be rounded tothe closest gridline.

`vAxis.baselineColor` a string. Default `'black'`. `vAxis` property that specifies the color of the baseline for the vertical axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`.

`vAxis.direction` 1 or -1. Default 1. The direction in which the values along the vertical axis grow. Specify -1 to reverse the order of the values.

`vAxis.format` a string. Default auto. A format string for numeric axis labels. This is a subset of the ICU pattern set. For instance,

`{format:'#,###%'}`.

will display values 1,000%, 750%, and 50% for values 10, 7.5, and 0.5.

`vAxis.gridlines` a JSON object. Default `NULL`. An object with members to configure the gridlines on the vertical axis. To specify properties of this object, you can use object literal notation, as shown here:

`{color: '#333', count: 4}`

`vAxis.gridlines.color` a string. Default `'#CCC'`. The color of the vertical gridlines inside the chart area. Specify a valid HTML color string.

`vAxis.gridlines.count` a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

`vAxis.logScale` boolean. Default `FALSE`. `vAxis` property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to `TRUE` for yes.

`vAxis.textPosition` a string. Default `'out'`. Position of the vertical axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`vAxis.textStyle` a JSON object. Default

`{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}.`

An object that specifies the vertical axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.title` a string. Default no title. `vAxis` property that specifies a title for the vertical axis.

`vAxis.titleTextStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}.
```

An object that specifies the vertical axis title text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
 <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.maxValue` a number. Default automatic. vAxis property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.

`vAxis.minValue` a number. Default automatic. `vAxis` property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.

`vAxis.viewWindowMode` a string. Default `"pretty"` if `vAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the vertical axis to render the values within the chart area. The following string values are supported:

`'pretty'` Scale the vertical values so that the maximum and minimum data values are rendered a bit inside the top and bottom of the chart area.

`'maximized'` Scale the vertical values so that the maximum and minimum data values touch the top and bottom of the chart area.

`'explicit'` Specify the top and bottom scale values of the chart area. Data values outside these values will be cropped. You must specify a `vAxis.viewWindow` object describing the maximum and minimum values to show.

`vAxis.viewWindow` Object. Default `NULL`. Specifies the maximum and minimum data values to show on the vertical axis. Present only if `vAxis.viewWindowMode='explicit'`

`vAxis.viewWindow.max` A number. Default 0. The maximum vertical data value to render.

`vAxis.viewWindow.min` A number. Default 0. The minimum vertical data value to render.

`width` a number. Default width of the containing element. Width of the chart, in pixels.

`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, here 'ScatterChart'
`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.
  - `jsChart` Call of the `jsDisplayChart` function.
  - `jsFooter` End tag `</script>`.
  - `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

Google Visualisation API: You cannot load both scatterchart and corechart packages at the same time on the same page.

Author(s)

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

Diego de Castillo <`decastillo@gmail.com`>

References

Google Scatter Chart API: `http://code.google.com/apis/chart/interactive/docs/gallery/scatterchart.html`

Follow the link for Google's data policy.

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=0,
                 title="Women", vAxis="{title:'weight (lbs)'}",
                 hAxis="{title:'height (in)'}", width=300, height=300))

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)
```

`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.

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

Usage

```
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 for Google Stepped Area Chart.

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

Further possible components are, taken from https://google-developers.appspot.com/chart/interactive/docs/gallery/steppedareachart.html#Configuration_Options:

`areaOpacity` a number between 0.0 - 1.0. Default 0.3. The default opacity of the colored area under an area chart series, where 0.0 is fully transparent and 1.0 is fully opaque. To specify opacity for an individual series, set the areaOpacity value in the series property.

`axisTitlesPosition` a string. Default 'out'. Where to place the axis titles, compared to the chart area. Supported values:

`'in'` Draw the axis titles inside the the chart area.

`'out'` Draw the axis titles outside the chart area.

`'none'` Omit the axis titles.

`backgroundColor` a string or object. Default `'white'`. The background color for the main area of the chart. Can be either a simple HTML color string, for example: `'red'` or `'#00cc00'`, or an object with the following properties.

`backgroundColor.stroke` a string. Default `'#666'`. The color of the chart border, as an HTML color string.

`backgroundColor.strokeWidth` a number. Default 0. The border width, in pixels.

`backgroundColor.fill` a string. Default `'white'`. The chart fill color, as an HTML color string.

`chartArea` a string. Default `'null'`. An object with members to configure the placement and size of the chart area (where the chart itself is drawn, excluding axis and legends). Two formats are supported: a number, or a number followed by `%`. A simple number is a value in pixels; a number followed by `%` is a percentage. Example:

`{left:20,top:0,width:\"50%\",height:\"75%\"}`

`chartArea.height` a number or string. Default auto. Chart area height.

`chartArea.left` a number or string. Default auto. How far to draw the chart from the left border.

`chartArea.top` a number or string. Default auto. How far to draw the chart from the top border.

`chartArea.width` a number or string. Default auto. Chart area width.

`colors` an array of strings. Default 'colors'. The colors to use for the chart elements. An array of strings, where each element is an HTML color string, for example: `colors:[red','#004411']`.

`connectSteps` boolean. Default `TRUE`. If set to `TRUE`, will connect the steps to form a stepped line. Otherwise, only a top line appears. The default is to connect the steps.

`enableInteractivity` boolean. Default `TRUE`. Whether the chart throws user-based events or reacts to user interaction. If false, the chart will not throw 'select' or other interaction-based events (but will throw ready or error events), and will not display hovertext or otherwise change depending on user input.

`focusTarget` a string. Default 'datum'. The type of the entity that receives focus on mouse hover. Also affects which entity is selected by mouse click, and which data table element is associated with events. Can be one of the following:
`'datum'` Focus on a single data point. Correlates to a cell in the data table.
`'category'` Focus on a grouping of all data points along the major axis. Correlates to a row in the data table.
In focusTarget `'category'` the tooltip displays all the category values. This may be useful for comparing values of different series.

`fontSize` a number. Default automatic. The default font size, in pixels, of all text in the chart. You can override this using properties for specific chart elements.

`fontName` a string. Default `'Arial'`. The default font face for all text in the chart. You can override this using properties for specific chart elements.

`hAxis` a JSON object. Default `'null'`. An object with members to configure various horizontal axis elements. To specify properties of this object, you can use object literal notation, as shown here:

```
{title: 'Hello', titleTextStyle: {color: '#FF0000'}}
```

`hAxis.direction` 1 or -1. Default 1. The direction in which the values along the horizontal axis grow. Specify -1 to reverse the order of the values.

`hAxis.textPosition` a string. Default `'out'` Position of the horizontal axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`hAxis.textStyle` a JSON object. Default

```
{color: 'black',
fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the horizontal axis text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize: <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.title` a string. Default `'null'`. `hAxis` property that specifies the title of the horizontal axis.

`hAxis.titleTextStyle` a JSON object. Default

```
{color: 'black',
fontName: <global-font-name>, fontSize: <global-font-size>}.
```

An object that specifies the horizontal axis title text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize: <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`hAxis.slantedText` Boolean. Default automatic. If `TRUE`, draw the horizontal axis text at an angle, to help fit more text along the axis; if false, draw horizontal axis text upright. Default behavior is to slant text if it cannot all fit when drawn upright.

`hAxis.slantedTextAngle` a number, 1-90. Default 30. The angle of the horizontal axis text, if it's drawn slanted. Ignored if `hAxis.slantedText` is false, or is in auto mode, and the chart decided to draw the text horizontally.

`hAxis.maxAlternation` a number. Default 2. Maximum number of levels of horizontal axis text. If axis text labels become too crowded, the server might shift neighboring labels up or down in order to fit labels closer together. This value specifies the most number of levels to use; the server can use fewer levels, if labels can fit without overlapping.

`hAxis.showTextEvery` a number. Default automatic. How many horizontal axis labels to show, where 1 means show every label, 2 means show every other label, and so on. Default is to try to show as many labels as possible without overlapping.

`height` a number. Default height of the containing element. Height of the chart, in pixels.

`isStacked` boolean. Default `FALSE`. If set to `TRUE`, bar values are stacked (accumulated).

`legend` a JSON object. Default `NULL`. An object with members to configure various aspects of the legend. To specify properties of this object, you can use object literal notation, as shown here:

```
{position: 'top', textStyle: {color: 'blue', fontSize: 16}}
```

`legend.position` a string. Default `'right'`. Position of the legend. Can be one of the following:

`'right'` To the right of the chart.

`'top'` Above the chart.

`'bottom'` Below the chart.

`'none'` No legend is displayed.

`legend.textStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the legend text style. The object has this format:

```
{color: <string>, fontName: <string>, fontSize:
  <number>}
```

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`lineWidth` a number. Default 2. Line width in pixels. Use zero to hide all lines and show only the points.

`pointSize` a number. Default 0. Diameter of data points, in pixels. Use zero to hide all points.

`reverseCategories` Boolean. Default `FALSE`. If set to true, will draw series from right to left. The default is to draw left-to-right.

`series` an array of objects, or object with nested objects. Default {}. An array of objects, each describing the format of the corresponding series in the chart. To use default values for a series, specify an empty object . If a series or a value is not specified, the global value will be used. Each object supports the following properties:

`color` The color to use for this series. Specify a valid HTML color string.

`targetAxisIndex` Which axis to assign this series to, where 0 is the default axis, and 1 is the opposite axis. Default value is 0; set to 1 to define a chart where different series are rendered against different axes. You can define a different scale for different axes.

`pointSize` Overrides the global pointSize value for this series.

`lineWidth` Overrides the global lineWidth value for this series.

`curveType` Overrides the global curveType value for this series.

`visibleInLegend` A boolean value, where true means that the series should have a legend entry, and false means that it should not. Default is `TRUE`.

You can specify either an array of objects, each of which applies to the series in the order given, or you can specify an object where each child has a numeric key indicating which series it applies to. For example, the following two declarations are identical, and declare the first series as black and absent from the legend, and the fourth as red and absent from the legend:

```
series: [{color: 'black', visibleInLegend: false},{}, {}, {color:
'red', visibleInLegend: false}]

series: {0:{color: 'black', visibleInLegend: false}, 3:{color: 'red',
visibleInLegend: false}}
```

`theme` a string. Default `NULL`. A theme is a set of predefined option values that work together to achieve a specific chart behavior or visual effect. Currently only one theme is available:

`maximized` Maximizes the area of the chart, and draws the legend and all of the labels inside the chart area. Sets the following options:

```
chartArea: {width: '100%', height: '100%'},
legend: {position: 'in'},
titlePosition: 'in', axisTitlesPosition: 'in',
hAxis: {textPosition: 'in'}, vAxis: {textPosition: 'in'}
```

`title` a string. Default no title. Text to display above the chart.

`titlePosition` a string. Default `'out'`. Where to place the chart title, compared to the chart area. Supported values:

`'in'` Draw the title inside the chart area.

`'out'` Draw the title outside the chart area.

`'none'` Omit the title.

`titleTextStyle` a JSON object. Default

`{color:'black', fontName:<global-font-name>,fontSize:<global-font-size>}`.

An object that specifies the title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip` a JSON object. Default `NULL`. An object with members to configure various tooltip elements. To specify properties of this object, you can use object literal notation, as shown here:

`{textStyle: {color: '#FF0000'}, showColorCode: true}`

`tooltip.showColorCode` boolean. Default automatic. If true, show colored squares next to the series information in the tooltip. The default is true when `focusTarget` is set to `'category'`, otherwise the default is `FALSE`.

`tooltip.TextStyle` a JSON object. Default

```
{color: 'black',
  fontName: <global-font-name>, fontSize: <global-font-size>}
```

An object that specifies the tooltip text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`tooltip.trigger` The user interaction that causes the tooltip to be displayed:

`'hover'` The tooltip will be displayed when the user hovers over an element.

`'none'` The tooltip will not be displayed.

`vAxes` Array of object, or object with child objects null. Specifies properties for individual vertical axes, if the chart has multiple vertical axes. Each child object is a `vAxis` object, and can contain all the properties supported by `vAxis`. These property values override any global settings for the same property.
To specify a chart with multiple vertical axes, first define a new axis using `series.targetAxisIndex`, then configure the axis using `vAxes`. The following example assigns series 2 to the right axis and specifies a custom title and text style for it:

```
series:{2:{targetAxisIndex:1}},
  vAxes:{1:{title:'Losses',textStyle:{color: 'red'}}}
```

This property can be either an object or an array: the object is a collection of objects, each with a numeric label that specifies the axis that it defines–this is the format shown above; the array is an array of objects, one per axis. For example, the following array-style notation is identical to the `vAxis` object shown above:

```
vAxes:[
{}, // Nothing specified for axis 0
{title:'Losses',textStyle:{color: 'red'}} // Axis 1
]
```

`vAxis` a JSON object. Default `'null'`. An object with members to configure various vertical axis elements. To specify properties of this object, you can use object literal notation, as shown here:

`{title: 'Hello', titleTextStyle: {color: '#FF0000'}}`

`vAxis.baseline` a number. Default automatic. `vAxis` property that specifies the baseline for the vertical axis. If the baseline is smaller than the highest grid line or smaller than the lowest grid line, it will be rounded tothe closest gridline.

`vAxis.baselineColor` a string. Default `'black'`. `vAxis` property that specifies the color of the baseline for the vertical axis. Can be any HTML color string, for example: `'red'` or `'#00cc00'`.

`vAxis.direction` 1 or -1. Default 1. The direction in which the values along the vertical axis grow. Specify -1 to reverse the order of the values.

`vAxis.format` a string. Default auto. A format string for numeric axis labels. This is a subset of the ICU pattern set. For instance,

`{format:'#,###%'}`.

will display values 1,000%, 750%, and 50% for values 10, 7.5, and 0.5.

`vAxis.gridlines` a JSON object. Default `NULL`. An object with members to configure the gridlines on the vertical axis. To specify properties of this object, you can use object literal notation, as shown here:

`{color: '#333', count: 4}`

`vAxis.gridlines.color` a string. Default `'#CCC'`. The color of the vertical gridlines inside the chart area. Specify a valid HTML color string.

`vAxis.gridlines.count` a number. Default 5.The number of vertical gridlines inside the chart area. Minimum value is 2.

`vAxis.logScale` Boolean. Default `FALSE`. `vAxis` property that makes the vertical axis a logarithmic scale (requires all values to be positive). Set to `TRUE` for yes.

`vAxis.textPosition` a string. Default `'out'`. Position of the vertical axis text, relative to the chart area. Supported values: `'out', 'in', 'none'`.

`vAxis.textStyle` a JSON object. Default

`{color: 'black', fontName: <global-font-name>, fontSize: <global-font-size>}`.

An object that specifies the vertical axis text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

`vAxis.title` a string. Default no title. `vAxis` property that specifies a title for the vertical axis.

`vAxis.titleTextStyle` a JSON object. Default

`{color: 'black',`
`fontName: <global-font-name>, fontSize: <global-font-size>}`.

An object that specifies the vertical axis title text style. The object has this format:

`{color: <string>, fontName: <string>, fontSize: <number>}`

The color can be any HTML color string, for example: `'red'` or `'#00cc00'`. Also see `fontName` and `fontSize`.

- `vAxis.maxValue` a number. Default automatic. vAxis property that specifies the highest vertical axis grid line. The actual grid line will be the greater of two values: the maxValue option value, or the highest data value, rounded up to the next higher grid mark.
- `vAxis.minValue` a number. Default automatic. `vAxis` property that specifies the lowest vertical axis grid line. The actual grid line will be the lower of two values: the minValue option value, or the lowest data value, rounded down to the next lower grid mark.
- `vAxis.viewWindowMode` a string. Default `"pretty"` if `vAxis.viewWindow` is null, `"explicit"` otherwise. Specifies how to scale the vertical axis to render the values within the chart area. The following string values are supported:
  - `'pretty'` Scale the vertical values so that the maximum and minimum data values are rendered a bit inside the top and bottom of the chart area.
  - `'maximized'` Scale the vertical values so that the maximum and minimum data values touch the top and bottom of the chart area.
  - `'explicit'` Specify the top and bottom scale values of the chart area. Data values outside these values will be cropped. You must specify a `vAxis.viewWindow` object describing the maximum and minimum values to show.
- `vAxis.viewWindow` Object. Default `NULL`. Specifies the maximum and minimum data values to show on the vertical axis. Present only if `vAxis.viewWindowMode='explicit'`
- `vAxis.viewWindow.max` A number. Default 0. The maximum vertical data value to render.
- `vAxis.viewWindow.min` A number. Default 0. The minimum vertical data value to render.
- `width` a number. Default width of the containing element. Width of the chart, in pixels.

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

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, here 'SteppedAreaChart'

`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.
- `jsChart` Call of the `jsDisplayChart` function.
- `jsFooter` End tag `</script>`.
- `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

Google Visualisation API: You cannot load both steppedareachart and corechart packages at the same time on the same page.

Author(s)

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

Diego de Castillo <`decastillo@gmail.com`>

References

Google Stepped Area Chart API: `http://code.google.com/apis/chart/interactive/docs/gallery/steppedareachart.html`

Follow the link for Google's data policy.

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 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.

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.

Usage

```
gvisTable(data, options = list(), chartid)
```

Arguments

`data` a `data.frame` to be displayed as a table

`options` list of configuration options for Google Table.

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

Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/table.html#Configuration_Options`:

`allowHtml` boolean. Default `FALSE`. If set to `TRUE`, formatted values of cells that include HTML tags will be rendered as HTML. If set to `FALSE`, most custom formatters will not work properly.

`alternatingRowStyle` boolean. Default `TRUE` Determines if alternating color style will be assigned to odd and even rows.

`cssClassNames` An object in which each property name describes a table element, and the property value is a string, defining a class to assign to that table element. Use this property to assign custom CSS to specific elements of your table. To use this property, assign an object, where the property name specifies the table element, and the property value is a string, specifying a class name to assign to that element. You must then define a CSS style for that class on your page. The following property names are supported:
`headerRow` - Assigns a class name to the table header row (<tr> element).
`tableRow` - Assigns a class name to the non-header rows (<tr> elements).
`oddTableRow` - Assigns a class name to odd table rows (<tr> elements). Note: the `alternatingRowStyle` option must be set to true.
`selectedTableRow` - Assigns a class name to the selected table row (<tr> element).
`hoverTableRow` - Assigns a class name to the hovered table row (<tr> element).
`headerCell` - Assigns a class name to all cells in the header row (<td> element).
`tableCell` - Assigns a class name to all non-header table cells (<td> element).
`rowNumberCell` - Assigns a class name to the cells in the row number column (<td> element). Note: the `showRowNumber` option must be set to `TRUE`.
Example: var cssClassNames = {headerRow: 'bigAndBoldClass', hoverTableRow: 'highlightClass'};

`firstRowNumber` number. Default 1. The row number for the first row in the data. Used only if `showRowNumber` is `TRUE`.

`height` string. Sets the height of the visualization's container element. You can use standard HTML units (for example, '100px', '80em', '60'). If no units are specified the number is assumed to be pixels. If not specified, the browser will set the width automatically to fit the table; if set smaller than the size required by the table, will add a vertical scroll bar.

`page` string. Default `'disable'`. If and how to enable paging through the data. Choose one of the following string values:
`'enable'` - The table will include page-forward and page-back buttons. Clicking on these buttons will perform the paging operation and change the displayed page. You might want to also set the `pageSize` option.
`'event'` - The table will include page-forward and page-back buttons, but clicking them will trigger a 'page' event and will not change the displayed page. This option should be used when the code implements its own page turning logic. See the TableQueryWrapper example for an example of how to handle paging events manually.
`'disable'` - [Default] Paging is not supported.

`pageSize` number. Default 10. The number of rows in each page, when paging is enabled with the page option.

`rtlTable` boolean. Default `FALSE`. Adds basic support for right-to-left languages (such as Arabic or Hebrew) by reversing the column order of the table, so that column zero is the rightmost column, and the last column is the leftmost column. This does not affect the column index in the underlying data, only the order of display. Full bi-directional (BiDi) language display is not supported by the table visualisation even with this option. This option will be ignored if you enable paging (using the page option), or if the table has scroll bars because you have specified height and width options smaller than the required table size.

`scrollLeftStartPosition` number. Default 0. Sets the horizontal scrolling position, in pixels, if the table has horizontal scroll bars because you have set the width property. The table will open scrolled that many pixels past the leftmost column.

`showRowNumber` boolean. Default `FALSE`. If set to true, shows the row number as the first column of the table.

`sort` string. Default `'enable'`. If and how to sort columns when the user clicks a column heading. If sorting is enabled, consider setting the `sortAscending` and `sortColumn` properties as well. Choose one of the following string values:
`'enable'` - [Default] Users can click on column headers to sort by the clicked column. When users click on the column header, the rows will be automatically sorted, and a 'sort' event will be triggered.
`'event'` - When users click on the column header, a 'sort' event will be triggered, but the rows will not be automatically sorted. This option should be used when the page implements its own sort. See the TableQueryWrapper example for an example of how to handle sorting events manually.
`'disable'` - Clicking a column header has no effect.

`sortAscending` boolean. Default `TRUE`. The order in which the initial sort column is sorted. True for ascending, false for descending. Ignored if `sortColumn` is not specified.

`sortColumn` number. Default -1. An index of a column in the data table, by which the table is initially sorted. The column will be marked with a small arrow indicating the sort order.

`startPage` number. Default 0. The first table page to display. Used only if page is in mode enable/event.

`width` string. Sets the width of the visualisation's container element. You can use standard HTML units (for example, '100px', '80em', '60'). If no units are specified the number is assumed to be pixels. If not specified, the browser will set the width automatically to fit the table; if set smaller than the size required by the table, will add a horizontal scroll bar.

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

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, here 'Table'
`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.
- `jsChart` Call of the `jsDisplayChart` function.
- `jsFooter` End tag `</script>`.
- `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 Table API: https://google-developers.appspot.com/chart/interactive/docs/gallery/table.html

Follow the link for Google's data policy.

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=300))

plot(tbl2)
```

gvisTreeMap *Google Tree Map with R*

Description

The gvisTreeMap 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.

Usage

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

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 for Google Tree Map. `gvis.editor` a character label for an on-page button which opens an in-page dialog box that enables users to edit, change and customise the chart. By default no value is given and therefore no button is displayed. Further possible components are, taken from `https://google-developers.appspot.com/chart/interactive/docs/gallery/treemap.html#Configuration_Options`:

`headerColor` string, default '#988f86'. The color of the header section for each node. Specify an HTML color value.

`headerHeight` number, default 0. The height of the header section for each node, in pixels (can be zero).

`headerHighlightColor` string, default `'null'`. The color of the header of a node being hovered over. Specify an HTML color value, null, or 'auto'; if null or 'auto', this value will be `headerColor` lightened by 35%

`maxColor` string, default #00dd00. The color for a rectangle with a `sizevar` value of `maxColorValue`. Specify an HTML color value.

`maxDepth` number, default 1. The maximum number of node levels to show in the current view. Levels will be flattened into the current plane. If your tree has more levels than this, you will have to go up or down to see them. You can additionally see `maxPostDepth` levels below this as shaded rectangles within these nodes.

`maxHighlightColor` string, default null. The highlight color to use for the node with the largest value in column 3. Specify an HTML color value, null, or 'auto. If null or 'auto', this value will be the value of `maxColor` lightened by 35%

`maxPostDepth` number, default 1. How many levels of nodes beyond `maxDepth` to show in "hinted" fashion. Hinted nodes are shown as shaded rectangles within a node that is within the `maxDepth` limit.

`maxColorValue` number, default null. The maximum value allowed in column `sizevar`. All values greater than this will be trimmed to this value. If set to null or 'auto', it will be set to the max value in the column.

`midColor` string, default #000000. The color for a rectangle with a column `sizevar` value midway between `maxColorValue` and `minColorValue`. Specify an HTML color value.

`midHighlightColor` string, default null. The highlight color to use for the node with a column `sizevar` value near the median of `minColorValue` and `maxColorValue`. Specify an HTML color value or 'auto'. If null or 'auto', this value will be the value of `midColor` lightened by 35%.

`minColor` string, default #dd0000. The color for a rectangle with the column `sizevar` value of `minColorValue`. Specify an HTML color value.

`minHighlightColor` string, default null. The highlight color to use for the node with a column `sizevar` value nearest to `minColorValue`. Specify an HTML color value or 'auto'. If null or 'auto', this value will be the value of `minColor` lightened by 35'

`minColorValue` number, default null. The minimum value allowed in column `sizevar`. All values less than this will be trimmed to this value. If set to null or 'auto', it will be calculated as the minimum value in the column.

`noColor` string, default #000000. The color to use for a rectangle when a node has no value for column `sizevar`, and that node is a leaf (or contains only leaves). Specify an HTML color value.

`noHighlightColor` string, default null. The color to use for a rectangle of "no" color when highlighted. This will be the value of `noColor` lightened by 35%. Specify an HTML value.

	`showScale` boolean, default `FALSE`. Whether or not to show a color gradient scale from `minColor` to `maxColor` along the top of the chart. Specify true to show the scale.
	`showTooltips` boolean, default `TRUE`. Whether or not to show tooltips.
	`fontColor` string, default #ffffff. The text color. Specify an HTML color value.
	`fontFamily` string, default auto. The font family to use for all text.
	`fontSize` number, default 12. The font size for all text, in points.
`chartid`	character. If missing (default) a random chart id will be generated based on chart type and `tempfile`

Details

From `http://code.google.com/apis/chart/interactive/docs/gallery/treemap.html#Overview`:

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, here 'TreeMap'
`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.
	`jsChart` Call of the `jsDisplayChart` function.
	`jsFooter` End tag `</script>`.
	`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 Tree Map API: `https://google-developers.appspot.com/chart/interactive/docs/gallery/treemap.html`

Follow the link for Google's data policy.

See Also

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

Please note that the `treemap` package offeres 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",
                          "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)
```

`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

Google Visualisation: Candlestick Chart `http://code.google.com/apis/chart/interactive/docs/gallery/candlestickchart.html`

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

Sourced from Wikipedia (columns 1 to 5): http://en.wikipedia.org/wiki/List_of_countries_by_population, 9 October 2010.

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)
```

Stock *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://google-developers.appspot.com/chart/interactive/docs/gallery/annotatedtimeline.html

Examples

```
## Create data as used by Google in their annotated time line example

 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 of stock on 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