

# Using the Google Visualisation API with R

by Markus Gesmann and Diego de Castillo  
February 7, 2011

**Abstract** The **googleVis** package provides an interface between R and the Google Visualisation API. The Google Visualisation API offers interactive charts which can be embedded into web pages. The best known of these charts is probably the Motion Chart, popularised by Hans Rosling in his TED talks. With the **googleVis** package users can create easily 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. The current version (0.2.4) of the package provides interfaces to Motion Charts, Annotated Time Lines, Maps, Geo Maps, Tables and Tree Maps.

## Motivation

In 2006 Hans Rosling gave an inspiring talk at TED (Rosling, 2006) 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 (Gapminder Foundation, 2010) had developed animated bubble charts, aka motion charts, see Figure 1.

Rosling's presentation popularised the idea and use of interactive charts, and as a result the software behind Gapminder was bought by Google and integrated as motion charts into their Visualisation API (Google Inc., 2010h) one year later.

In 2010 Sebastián Pérez Saaibi (Saaibi, 2010) presented at the R/Rmetrics Workshop on Computational Finance and Financial Engineering the idea to link Google motion charts with R using the **R.rsp** package (Bengtsson, 2009).

Inspired by those talks and the desire to use interactive data visualisation tools to foster the dialogue between data analysts and others the authors of this article started the development of the **googleVis** package (Gesmann and de Castillo, 2011).

## Google Visualisation API

The Google Visualisation API (Google Inc., 2010h), (Google Inc) allows users to create interactive charts as part of Google documents, spreadsheets and web pages. In this text we will focus on the usage of the API as part of web sites.

The Google Public Data Explorer (Google Inc., 2010e) provides a good example, demonstrating the use of motion charts and how they can help to analyse data. Please note, that most charts are rendered within a browser using Adobe Flash (Adobe Incorporated).

The charting data can either be embedded into the html file or read dynamically. The key to the Google Visualisation API is that the data is structured in a DataTable (Google Inc., 2010a), and this is where the **googleVis** package helps, as it uses the functionality of the **RJSONIO** package (Temple Lang, 2010) to transform R data frames into JSON (JSON.org, 2006) objects as the basis for a DataTable.

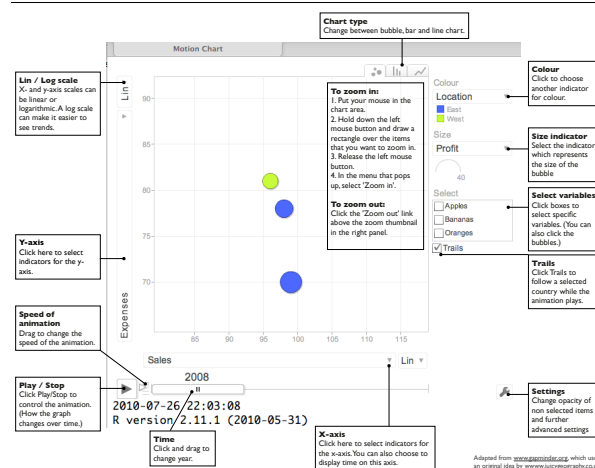


Figure 1: Overview of a Google Motion Chart. Screenshot of the output of plot(gvisMotionChart(Fruits, idvar='Fruit', timevar='Year'))

As an example we shall look at the html code of a motion chart from Google's visualisation gallery (Google Inc., 2010d), which generates output similar to Figure 1:

```
<html>
<head>
  <script type="text/javascript"
    src="http://www.google.com/jsapi">
  </script>
  <script type="text/javascript">
    google.load('visualization', '1',
      {'packages':['motionchart']});
    google.setOnLoadCallback(drawChart);
    function drawChart() {
      var data=new google.visualization.DataTable();
      data.addColumn('string', 'Fruit');
      data.addColumn('date', 'Date');
      data.addColumn('number', 'Sales');
```

```

data.addColumn('number', 'Expenses');
data.addColumn('string', 'Location');
data.addRows([
  ['Apples', new Date(1988, 0, 1), 1000, 300, 'East'],
  ['Oranges', new Date(1988, 0, 1), 1150, 200, 'West'],
  ['Bananas', new Date(1988, 0, 1), 300, 250, 'West'],
  ['Apples', new Date(1989, 6, 1), 1200, 400, 'East'],
  ['Oranges', new Date(1989, 6, 1), 750, 150, 'West'],
  ['Bananas', new Date(1989, 6, 1), 788, 617, 'West']
]);
var chart=new google.visualization.MotionChart(
  document.getElementById('chart_div'));
chart.draw(data, {width: 600, height:300});
}
</script>
</head>
<body>
<div id="chart_div"
  style="width:600px; height:300px;">
</div>
</body>
</html>

```

You will notice that the above html code has three generic parts:

- reference to a JavaScript function provided by Google. Shown here as 'motionchart',
- data to visualise as a DataTable,
- chart with chart id ('chart\_div') and options, shown here as width and height.

These principles hold true for most of the interactive charts of the Google Visualisation API, see the examples in Figure 2.

## The googleVis package

The **googleVis** package provides an interface between R and the Google Visualisation API. The functions of the package allow the user to visualise data stored in R data frames with the Google Visualisation API.

The output of a **googleVis** function is html code that contains the data and references to JavaScript functions hosted by Google. To view the output a browser with Flash and Internet connection is required, the actual chart is rendered in the browser; it may not work when loaded as a local file. For more details see the Google Visualisation API documentation (Google Inc., 2010d).

Fortunately, R comes with an internal HTTP server which allows the **googleVis** package to display pages locally. Other options are to use the **R.jsp** package or RApache (Horner, 2011) with **brew** (Horner, 2010). Both **R.jsp** and **brew** have the capability to extract and execute R code from html code, similar to the approach taken by Sweave (Leisch, 2002) for L<sup>A</sup>T<sub>E</sub>X. For more details see the package documentation.

Currently the **googleVis** package provides interfaces to Motion Chart, Annotated Time Line, Geo Map, Map, Table and Tree Map; see Figure 2 for examples.

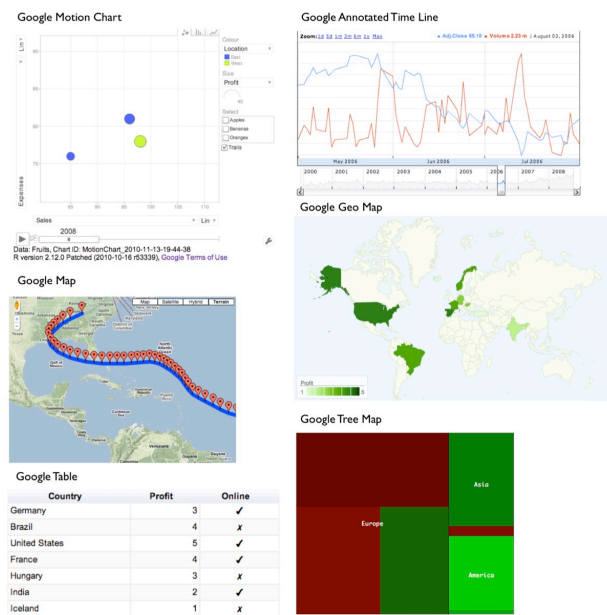


Figure 2: Screenshot of some of the outputs of `demo(googleVis)` with `gvisMotionChart`, `gvisAnnotatedTimeLine`, `gvisMap`, `gvisGeoMap`, `gvisTable` and `gvisTreeMap` from top left to bottom right.

The individual functions of the package are documented in detail in the help pages. Here we will cover only the principles of the package.

As an example we will show how to generate a motion chart as displayed in Figure 1. It works similarly for the other APIs. Further examples are covered in the demos of the **googleVis** package.

The design of the visualisation functions are fairly generic. The name of the visualisation function is 'gvis' + ChartType. So for the Motion Chart we have:

```

gvisMotionChart(data, idvar='id', timevar='date',
  options=list())

```

Here `data` is the input `data.frame` and `idvar` and `timevar` specify the column names of the id variable and time variable for the plot, while display options are set in an optional list. The options and data requirements follow those of the Google Visualisation API and are documented in the help pages, see `help('gvisMotionChart')`.

The output of a **googleVis** function is a list of lists (a nested list) containing information about the chart type, chart id and the html code in a sub-list split into header, chart, caption and footer.

The idea behind this concept is that users can get a complete web page while at the same time extracting specific parts, such as the chart. This is particular

helpful if the package functions are used in solutions where the user wants to feed the visualisation output into other sites, or would like to embed them into rsp-pages, or use RApache or Google Gadgets.

The output of a **googleVis** function will be of class 'gvis' and 'list'. Generic print (print.gvis) and plot (plot.gvis) functions exist to ease the handling of such objects.

To illustrate the concept we shall create a motion chart using the Fruits data set.

## Motion Chart Example

Following the documentation of the Google Motion Chart API we need a data set which has at least four columns: one identifying the variable we would like to plot, one time variable and at least two numerical variables, further numerical and character columns are allowed.

As an example we use the Fruits data set:

```
R> data(Fruits)
R> Fruits[, -7] # ignore column 7
```

	Fruit	Year	Location	Sales	Expenses	Profit
1	Apples	2008	West	98	78	20
2	Apples	2009	West	111	79	32
3	Apples	2010	West	89	76	13
4	Oranges	2008	East	96	81	15
5	Bananas	2008	East	85	76	9
6	Oranges	2009	East	93	80	13
7	Bananas	2009	East	94	78	16
8	Oranges	2010	East	98	91	7
9	Bananas	2010	East	81	71	10

Here we will use the columns 'Fruit' and 'Year' as id and time variable respectively.

```
R> M <- gvisMotionChart(Fruits, idvar="Fruit",
  timevar="Year")
```

The structural output of gvisMotionChart is a list of lists as described above

```
R> str(M)

List of 3
 $ type      : chr "MotionChart"
 $ chartid   : chr "MotionChartID12ae2fff"
 $ html      :List of 4
  ..$ header : chr "<!DOCTYPE html PUBLIC ..."
  ..$ chart  : Named chr [1:7] "<!-- Moti ..."
  .. ..- attr(*, "names")= chr [1:7] "jsH ..."
  ..$ caption: chr "<div><span>Data: Fruit..."
  ..$ footer : chr "\n<!-- htmlFooter -->\n..."
  - attr(*, "class")= chr [1:2] "gvis" "list"
```

The first two items of the list contain information about the chart type used and the individual chart id. The html output is a list with header, chart, caption and footer. This allows the user to extract only certain parts of the page, or to create a complete html page.

The header part of the html page has only basic html and formatting tags and provides a simple layout.

The actual Google visualisation code is stored with the data in the named character vector chart of the html list, see

```
R> print(M, 'chart') # output not shown
```

The sub-items of the chart object give the user more granular access to JavaScript functions and html tags of the visualisation. A basic chart caption and html footer complete the html list:

```
R> print(M, 'caption') # output not shown
R> print(M, 'footer') # output not shown
```

## Displaying gvis objects

To display the page locally, type:

```
R> plot(M)
```

The plot method for gvis-objects creates html files in a temporary folder using the type and chart id information of the object and it will display the output using the R HTTP help web server locally.

Click on the chart id and you will get access to the html code. The R command tempdir() will show you the path of the per-session temporary directory.

Further examples are part of the **googleVis** demo, including one example demonstrating how the output of several visualisations can be incorporated into a single page.

## Embedding googleVis in web sites dynamically

With the R packages **R.rsp** and **brew** we have two options to integrate R snippets into html code. While the **R.rsp** package comes with its own internal web server, **brew** requires the Apache HTTP server with the RApache module installed.

Both packages allow the user to embed R code into html-code. The R code is filtered by the **R.rsp** or RApache web server and executed at run time. As an example, we can insert the above motion chart into a page:

```
<html>
<body>
<% library(googleVis) %>
<% M <- gvisMotionChart(Fruits, idvar="Fruit",
  timevar="Year") %>
<%= M$html$chart %>
</body>
</html>
```

The syntax of **R.rsp** and **brew** are very similar. The R code included in <%...%> is executed when read by the HTTP server, but no R output will be displayed. To insert the R output into the html code we have

to add an equal sign, `<%=...%>`, which acts as a cat statement. In the example above the chart code is embedded into the html code.

Examples for both approaches are part of the **googleVis** package. For more information see the vignettes and help files of the packages.

## Summary

Combining *R* with the Google Visualisation API enables the user to create powerful analysis tools which are easily accessible both to data and non-data analysts.

The [Statistics Relating to Lloyd's](http://www.lloyds.com/stats) site (<http://www.lloyds.com/stats>) is an example where the functions of the **googleVis** package were used to create a data visualisation page. The site shows how interactive charts can be used to slice and dice the data, to view it from various angles and to find stories worth telling: outliers, trends or even the obvious.

Please contact us if you have any feedback, questions or would like to collaborate: [rvisualisation@gmail.com](mailto:rvisualisation@gmail.com)

## Bibliography

- Adobe Inc. Adobe Flash Player. <http://get.adobe.com/flashplayer/>.
- H. Bengtsson. R.rsp: R server pages. <http://CRAN.R-project.org/package=R.rsp>, 2009. R package version 0.4.0.
- Gapminder Foundation. Gapminder. <http://www.gapminder.org>, 2010.
- M. Gesmann and D. de Castillo. googleVis: Using the Google Visualisation API with R. <http://code.google.com/p/google-motions-chart-with-r/>, 2011. R package version 0.2.4.
- Google Inc. Google Visualization API Terms of Service. <http://code.google.com/apis/visualization/terms.html>.
- Google Inc. Google Visualisation Reference. <http://code.google.com/apis/visualization/documentation/reference.html>, 2010a.
- Google Inc. Google Geo Map API. <http://code.google.com/apis/visualization/documentation/gallery/geomap.html>, 2010b.
- Google Inc. Google Map API. <http://code.google.com/apis/visualization/documentation/gallery/map.html>, 2010c.
- Google Inc. Google Motion Chart API. <http://code.google.com/apis/visualization/documentation/gallery/motionchart.html>, 2010d.
- Google Inc. Google Public Data Explorer. <http://www.google.com/publicdata/home>, 2010e.
- Google Inc. Google Table API. <http://code.google.com/apis/visualization/documentation/gallery/table.html>, 2010f.
- Google Inc. Google Tree Map API. <http://code.google.com/apis/visualization/documentation/gallery/treemap.html>, 2010g.
- Google Inc. Google Visualization API. <http://code.google.com/apis/visualization/documentation/gallery.html>, 2010h.
- Jeffrey Horner. brew: Templating framework for report generation. <http://CRAN.R-project.org/package=brew>, 2010. R package version 1.0-4.
- Jeffrey Horner. RApache: Web application development with R and Apache. <http://www.rapache.net/>, 2011.
- JSON.org. JSON. <http://www.json.org/>, 2006. RFC 4627 application/json.
- F. Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In W. Härdle and B. Rönz, editors, *Compstat 2002 — Proceedings in Computational Statistics*, pages 575–580. Physica Verlag, Heidelberg, 2002. URL <http://www.stat.uni-muenchen.de/~leisch/Sweave>. ISBN 3-7908-1517-9.
- H. Rosling. TED Talk: Hans Rosling shows the best stats you've ever seen. [http://www.ted.com/talks/hans\\_rosling\\_shows\\_the\\_best\\_stats\\_you\\_ve\\_ever\\_seen.html](http://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen.html), 2006.
- S. P. Saaibi. R/RMETRICS Generator Tool for Google Motion Charts. <https://www.rmetrics.org/>, 2010. Meielisalp, Lake Thune Switzerland, June 27 - July 1, 2010.
- D. Temple Lang. RJSONIO: Serialize R objects to JSON, JavaScript Object Notation. <http://www.omegahat.org/RJSONIO/>, 2010. R package version 0.4-1.

Markus Gesmann  
[markus.gesmann@gmail.com](mailto:markus.gesmann@gmail.com)

Diego de Castillo  
[decastillo@gmail.com](mailto:decastillo@gmail.com)