Using the Google Visualisation API with R: googleVis-0.2.5 Package Vignette

Markus Gesmann[*], Diego de Castillo[†]
Contact: rvisualisation@gmail.com

February 28, 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.

The functions of the `googleVis` package allow the user to visualise data stored in R data frames with the Google Visualisation API without uploading their data to Google. The output of a `googleVis` function is html code that contains the data and references to JavaScript functions hosted by Google and can be displayed via a browser.

Currently the package provides interfaces to Motion Charts, Annotated Time Lines, Maps, Geo Maps, Tables and Tree Maps.

[*]markus.gesmann@gmail.com
[†]decastillo@gmail.com

Contents

1 Introduction

1.1 Motivation

More and more data is becoming available, and yet stories and insights are still often missed: we are lost in the data jungle and struggle to see the wood for the trees.

Hence new tools are required to bring data to life, to engage with users, to enable them to slice and dice the data, to view it from various angles and to find stories worth telling: outliers, trends or even the obvious.

In 2006 Hans Rosling gave an inspiring talk at TED [Ros06] 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 [Fou10b] had developed animated bubble charts, aka motion charts, see Figure 1.

Rosling's presentation popularised the idea and use of interactive charts, and as one result the software behind Gapminder was bought by Google and integrated as motion charts into their Visualisation API [Inc10i] one year later.

We also notice that data journalism has grown over the recent years. The data blogs of the Guardian (UK), and taz.de (Die Tageszeitung, Germany) have brought data analysis and data visualisation to a wider audience.

In 2010 Sebastián Pérez Saaibi [Saa10] 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 [Ben10].

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 vignette started the development of the `googleVis` package [GdC11].

Of course there are many other alternative visualisation toolkits out there, e.g. Many Eyes [RtICsg10], Open Flash Chart (Flash) [JG10], OpenLayers (JavaScript) [Fou10c], Processing (Java) [FR10], simile (AJAX) [DKM10] and FLARE (ActionScript) [Lab10].

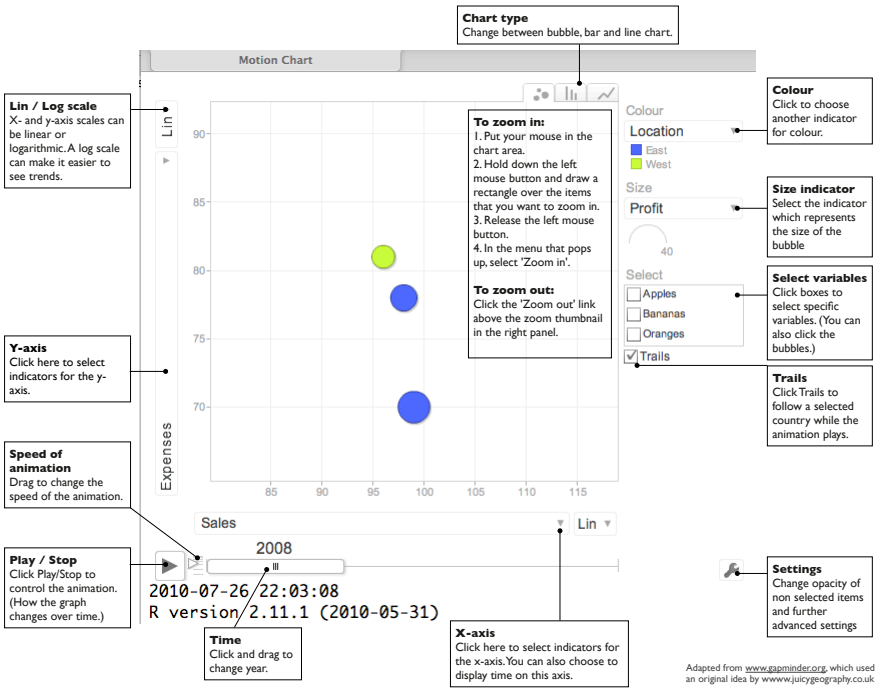


Figure 1: Overview of a Google Motion Chart. Screenshot of the output of `plot(gvisMotionChart(Fruits, idvar='Fruit', timevar='Year'))`

1.2 Google Visualisation API

The Google Visualisation API [Inc10i], [Inca] 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 pages.

The Google Public Data Explorer [Inc10e] provides a good example, demonstrating the use of motion charts and how they can help to analyse data. Please note, that most of those charts are rendered within a browser using Adobe Flash [Incb].

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 [Inc10h], and this is where the `googleVis` package helps, as it uses the functionality of the `RJSONIO` package [Lan10] to transform R data frames into JSON [JSO06] objects as the basis for a DataTable.

As an example we shall look at the html-code of a motion chart from Google's visualisation gallery [Inc10d], 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:

- references to JavaScript functions provided by Google.
- 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.

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 [Inc10d].

Fortunately, R comes with an internal HTTP server which allows the `googleVis` package to display pages locally.

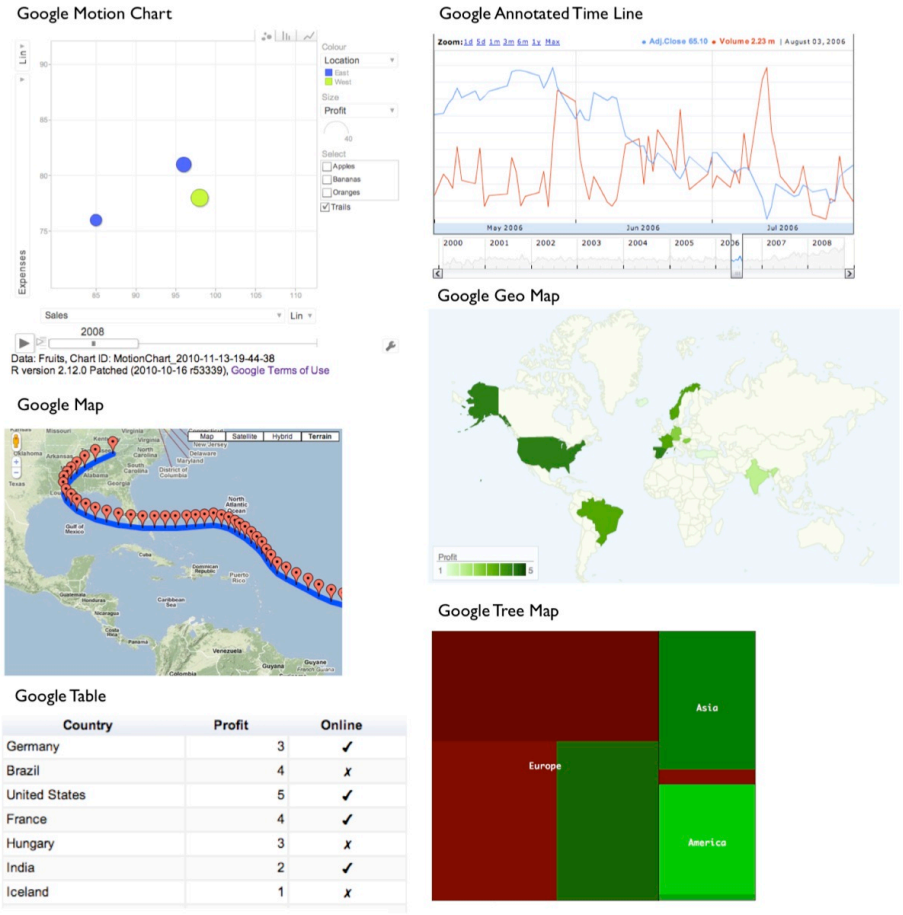
Currently the package provides interfaces to Motion Chart [Inc10d], Annotated Time Line [Inc10a], Geo Map [Inc10c], Map [Inc10b], Table [Inc10f] and Tree Map [Inc10g]; see Figure 2 for examples.

2.1 Installation

We can install `googleVis` in the usual way from CRAN, e.g.:

```
R> install.packages('googleVis')
```

The installation was successful if the command `library(googleVis)` gives you the following message:



Country	Profit	Online
Germany	3	✓
Brazil	4	✗
United States	5	✓
France	4	✓
Hungary	3	✗
India	2	✓
Iceland	1	✗

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.

```
R> library(googleVis)

To suppress the following message use the statement:
suppressPackageStartupMessages(library(googleVis))

Welcome to googleVis version 0.2.5

Type ?googleVis to access the overall documentation and
vignette('googleVis') for the package vignette.
You can execute the demo of the package via: demo(googleVis)

More information is available on the googleVis project web-site:
http://code.google.com/p/google-motion-charts-with-r/

Please read also the Google Visualisation API Terms of Use:
http://code.google.com/apis/visualization/terms.html

Feel free to send us an email <rvisualisation@gmail.com>
if you would like to be keept informed of new versions,
or if you have any feedback, ideas, suggestions or would
like to collaborate.
```

2.2 Using the `googleVis` package

The individual functions of the `googleVis` 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, see also Figure 2.

The design of the visualisation functions is 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(), chartid)
```

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

```
R> help('gvisMotionChart')
```

The argument `chartid` allows the user to set a chart id of the output chart manually. If the argument is missing a random id using `tempfile(pattern='')` will be generated. Unique chart ids are required to place more than one chart into a page.

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 with 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 (see page 16), or use *RApache* (see page 16) 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.

2.3 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
```

```
    Fruit Year Location Sales Expenses Profit       Date
1  Apples 2008     West    98       78     20 2008-12-31
2  Apples 2009     West   111       79     32 2009-12-31
3  Apples 2010     West    89       76     13 2010-12-31
4 Oranges 2008     East    96       81     15 2008-12-31
5 Bananas 2008     East    85       76      9 2008-12-31
6 Oranges 2009     East    93       80     13 2009-12-31
7 Bananas 2009     East    94       78     16 2009-12-31
8 Oranges 2010     East    98       91      7 2010-12-31
9 Bananas 2010     East    81       71     10 2010-12-31
```

Here we will use the columns 'Fruit' and 'Year' as id and time variable respectively. However we could heave used 'Date' instead of 'Year' as well.

```
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 "MotionChartID29d45fdc"
 $ html   :List of 4
  ..$ header : chr "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0
  ..$ chart  : Named chr [1:7] "<!-- MotionChart generated in R 2.
  .. ..- attr(*, "names")= chr [1:7] "jsHeader" "jsData" "jsDrawCh
  ..$ caption: chr "<div><span>Data: Fruits &#8226; Chart ID: <a h
  ..$ footer : chr "\n<!-- htmlFooter -->\n<span> \nR version 2.12
 - 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:

```
R> M$type
```

```
[1] "MotionChart"
```

```
R> M$chartid
```

```
[1] "MotionChartID29d45fdc"
```

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:

```
R> print(M, tag='header')
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>MotionChartID29d45fdc</title>
  <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  <style type="text/css">
    body {
          color: #444444;
          font-family: Arial,Helvetica,sans-serif;
          font-size: 75%;
    }
    a {
          color: #4D87C7;
          text-decoration: none;
    }
```

```
  </style>
</head>
<body>
```

Here we used the `print` statement with the tag `'header'` instead of `M$html$header` to achieve a formatted screen output. This is the same output as `cat(M$html$chart)`.

The actual Google visualisation code is stored with the data as a named character vector in the `chart` item of the html list. The chart is made up of several JavaScript and HTML statements. Please notice that the JavaScript functions are uniquely named with the information of the chart id. This concept allows the user get all the chart code directly or only specific parts; see the examples in the help page of `print.gvis` for more details.

```
R> names(M$html$chart)
```

```
[1] "jsHeader"       "jsData"         "jsDrawChart"    "jsDisplayChart"
[5] "jsChart"        "jsFooter"       "divChart"
```

The complete chart can be displayed via:

```
R> print(M, tag='chart')  ## or cat(M$html$chart)
```

```
<!-- MotionChart generated in R 2.12.2 by googleVis 0.2.5 package -->
<!-- Mon Feb 28 22:31:29 2011 -->


<!-- jsHeader -->
<script type="text/javascript" src="http://www.google.com/jsapi">
</script>
<script type="text/javascript">

// jsData
function gvisDataMotionChartID29d45fdc ()
{
  var data = new google.visualization.DataTable();
  var datajson =
[
 [
 "Apples",
  2008,
"West",
    98,
    78,
    20,
```

```
"2008-12-31"
],
[
 "Apples",
  2009,
"West",
   111,
    79,
    32,
"2009-12-31"
],
[
 "Apples",
  2010,
"West",
    89,
    76,
    13,
"2010-12-31"
],
[
 "Oranges",
  2008,
"East",
    96,
    81,
    15,
"2008-12-31"
],
[
 "Bananas",
  2008,
"East",
    85,
    76,
     9,
"2008-12-31"
],
[
 "Oranges",
  2009,
"East",
    93,
    80,
    13,
"2009-12-31"
```

```
],
[
 "Bananas",
  2009,
"East",
    94,
    78,
    16,
"2009-12-31"
],
[
 "Oranges",
  2010,
"East",
    98,
    91,
     7,
"2010-12-31"
],
[
 "Bananas",
  2010,
"East",
    81,
    71,
    10,
"2010-12-31"
]
];
data.addColumn('string','Fruit');
data.addColumn('number','Year');
data.addColumn('string','Location');
data.addColumn('number','Sales');
data.addColumn('number','Expenses');
data.addColumn('number','Profit');
data.addColumn('string','Date');
data.addRows(datajson);
return(data);
}

// jsDrawChart
function drawChartMotionChartID29d45fdc() {
  var data = gvisDataMotionChartID29d45fdc()
  var chart = new google.visualization.MotionChart(
   document.getElementById('MotionChartID29d45fdc')
  );
```

```
  var options ={};
options["width"] =    600;
options["height"] =    500;
  chart.draw(data,options);
}

// jsDisplayChart
function displayChartMotionChartID29d45fdc()
{
  google.load("visualization", "1", { packages:["motionchart"] });
  google.setOnLoadCallback(drawChartMotionChartID29d45fdc);
}

// jsChart
displayChartMotionChartID29d45fdc()

<!-- jsFooter -->
//-->
</script>

<!-- divChart -->
<div id="MotionChartID29d45fdc"
  style="width: 600px; height: 500px;">
</div>
```

Similarly you can also access specific components of the chart, e.g.

```
R> cat(M$html$chart['jsChart']) # or print(M, 'jsChart')

// jsChart
displayChartMotionChartID29d45fdc()
```

A basic chart caption and html footer are the final items of the html list (output is truncated):

```
R> print(M, tag='caption')

<div><span>Data: Fruits &#8226; Chart ID: <a href="Chart_MotionCha

R> print(M, tag='footer')

<!-- htmlFooter -->
<span>
R version 2.12.2 (2011-02-25) &#8226; <a href="http://code.google.
```

```
&#8226; <a href="http://code.google.com/apis/visualization/terms.h
</span></div>
</body>
</html>
```

2.4 Displaying `gvis` objects locally

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.

Please note that the chart caption provides a link to the chart code via the chart id for easy copy and paste.

The R command `tempdir()` will show you the path of the per-session temporary directory, in which the files were written.

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.

3 Embedding `googleVis` in web sites

3.1 Integrating `gvis` objects in existing sites

Suppose you have an existing web page and would like to integrate the output of a `googleVis` function, such as `gvisMotionChart`. In this case you only need the chart output from `gvisMotionChart`. So you can either copy and paste the output from the R console

```
R> print(M, 'chart')  ## or cat(M$html$chart)
```

into your existing html page, or write the content directly into a file

```
R> print(M, 'chart', file='myfilename')
```

and process it from there.

3.2 Embedding googleVis in web sites dynamically

In this section we provide examples how the googleVis functions can be embedded into web sites dynamically. With the R packages R.rsp [Ben10] and brew [Hor10] 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 [Fou10a] with the *RApache* [Hor11] module installed. Please note that currently the *RApache* module only runs on UNIX/Linux and Mac OS X.

3.2.1 Using googleVis with R.rsp

The R.rsp package allows the user to integrate R code into html code. The R code is filtered by the R.rsp web server and executed at run time.

As an example, we can embed the above motion chart into a rsp-page:

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

The R code included in <%...%> is executed when read by the R.rsp HTTP server, but no R output will be displayed. To embed the R output into the html code we have to add an equal sign, <%=...%>, which acts as a cat statement.

You find an example as part of the googleVis package. This example can be displayed via the following R command:

```
R> library(R.rsp)
R> browseRsp()
R> # Follow the link for googleVis in the opening browser window
```

The actual rsp-file is located within the googleVis package directory and again R allows you to find the file with the following command:

```
R> file.path(system.file("rsp", package = "googleVis"), "index.rsp")
```

For more information read the documentation of the R.rsp package.

3.2.2 Using googleVis with RApache and brew

RApache supports web application development using R and the Apache HTTP server. The *RApache* module embeds the R interpreter into the Apache web server.

However, as we would like to mix R and html code we also need a parser and this is where the R package `brew` comes into place.

Files sitting in a dedicated `brew` folder of the HTTP repository are parsed by `brew` when opened in the browser. The R code is executed with *RApache* and the output is embedded into the site. Hence the approach is similar to `R.rsp` with the difference that the two tasks are split. This has the advantage that R does not have to run in a separate window.

Detailed installation instructions for *RApache* are available on the project site: http://rapache.net/manual.html, for specific comments on Mac OS X see: http://-worldofrcraft.blogspot.com/2010/08/installing-rapache-on-mac-os-x-snow.html

Following the installation of *RApache* you will have to configure Apache. Most likely you have to add something along the following lines to your `apache2.conf` or `httpd.conf` file (often found in `/etc/httpd` or `/private/etc/apache2/httpd.conf` on Mac OS X):

```
LoadModule R_module /usr/lib/apache2/modules/mod_R.so
## On Mac OS X more likely to be:
## LoadModule R_module libexec/apache2/mod_R.so
ROutputErrors
RSourceOnStartup "/var/www/rapache/R/startup.R"
## On Mac OS X the www folder is often equivalent to:
## /Library/WebServer/Documents/
```

The first line loads the R module when the Apache web server is started, the second line deals with error handling, while the `startup.R` file is suitable for initial set ups, e.g. libraries and global variables:

```
## Ensure the packages are installed so that mod_R
## has access to them, e.g. not in your home folder
library{googleVis}
library{lattice}
library{Cairo}
MyGlobalVar <- 42
```

To test that *RApache* is working open http://localhost/RApacheInfo and you should find details about your system, an example can be found on the *RApache* site: http://biostat.mc.vanderbilt.edu/rapache/files/RApacheInfo.html

The next step is to install the `brew` R package in the usual way:

```
R> install.packages('brew')
```

Following this we have to tell Apache that files in a specific folder should be parsed by `brew`. Again we edit the `apache2.conf` or `httpd.conf` and add the connection of the RHandler with the function `brew`:

```
<Directory /var/www/rapache/brew>
## On Mac OS more likely to be something like:
## <Directory /Library/WebServer/Documents/rapache/brew>
      SetHandler r-script
      RHandler brew::brew
</Directory>
```

That's all. Restart the HTTP daemon and you can start placing files in the `brew` directory and access them via `http://localhost/rapache/brew/filename`, e.g. a file containing:

```
<html>
<body>
<h1>Fruits</h1>
<% library(googleVis)%>
<% M <- gvisMotionChart(Fruits, idvar="Fruit", timevar="Year") %>
<%= M$html$chart %>
</body>
</html>
```

You will notice that the `brew` syntax is very similar to `rsp`. For more information read the documentation of the *RApache* module and `brew` package. You find two simple examples of brew files in the `googleVis` package. Again the following R command shows you the folder path:

```
R> system.file("brew", package = "googleVis")
```

4 Contact

4.1 Collaboration

Obviously, the package is work in progress and there are many other functions of the Google Visualisation API which are still untouched.

Please feel free to send us an email if you would like to be kept informed of new versions, or if you have any feedback, ideas, suggestions or would like to collaborate, our address is rvisualisation@gmail.com.

4.2 Citation

Please cite R and/or `googleVis` if you use it in your work or publications. Use

```
R> citation()
```

or

```
R> citation("googleVis")
```

for information on how to cite the software.

4.3 Training and consultancy

Please contact us if you would like to discuss tailored training or consultancy: rvisualisation@gmail.com

References

[Ben10] Henrik Bengtsson. R.rsp: R server pages. http://CRAN.R-project.org/package=R.rsp, 2010. R package version 0.4.1.

[DKM10] MacKenzie Smith (MIT Libraries) David Karger (MIT CSAIL). Simile: Semantic Interoperability of Metadata and Information in unLike Environments. http://simile.mit.edu/, 2010.

[Fou10a] Apache Foundation. Apache HTTP Server 2.2. http://httpd.apache.org, 2010.

[Fou10b] Gapminder Foundation. Gapminder. http://www.gapminder.org, 2010.

[Fou10c] Open Source Geospatial Foundation. Openlayers: Free maps for the web. http://www.openlayers.org/, 2010.

[FR10] Ben Fry and Casey Reas. Processing an open source programming language and environment to create images, animations, and interactions. http://processing.org/, 2010.

[GdC11] Markus Gesmann and Diego 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.

[Hor10] Jeffrey Horner. brew: Templating framework for report generation. http://CRAN.R-project.org/package=brew, 2010. R package version 1.0-4.

[Hor11] Jeffrey Horner. RApache: Web application development with R and Apache. http://www.rapache.net/, 2011.

[Inca] Google Inc. Google Visualization API Terms of Service. http://code.google.com/apis/visualization/terms.html.

[Incb] Adobe Systems Incorporated. Adobe Flash Player. http://get.adobe.com/flashplayer/.

[Inc10a] Google Inc. Google Annotated Time Line API. http://code.google.com/apis/visualization/documentation/gallery/annotatedtimeline.html, 2010.

[Inc10b] Google Inc. Google Geo API. http://code.google.com/apis/visualization/documentation/gallery/map.html, 2010.

[Inc10c] Google Inc. Google Geo Map API. http://code.google.com/apis/visualization/documentation/gallery/geomap.html, 2010.

[Inc10d] Google Inc. Google Motion Chart API. http://code.google.com/apis/visualization/documentation/gallery/motionchart.html, 2010.

[Inc10e] Google Inc. Google Public Data Explorer. http://www.google.com/publicdata/home, 2010.

[Inc10f] Google Inc. Google Table API. http://code.google.com/apis/visualization/documentation/gallery/table.html, 2010.

[Inc10g] Google Inc. Google Tree Map API. http://code.google.com/apis/visualization/documentation/gallery/treemap.html, 2010.

[Inc10h] Google Inc. Google Visualisation Reference. http://code.google.com/apis/visualization/documentation/reference.html, 2010.

[Inc10i] Google Inc. Google Visualization API. http://code.google.com/apis/visualization/documentation/gallery.html, 2010.

[JG10] George Neusse John Glazebrook, Guenther Harrasser. Open flash chart. http://teethgrinder.co.uk/open-flash-chart/, 2010.

[JSO06] JSON.org. JSON. http://www.json.org/, 2006. RFC 4627 application/json.

[Lab10] UC Berkeley Visualization Lab. flare: Data visualisation for the web. http://flare.prefuse.org, 2010.

[Lan10] Duncan Temple Lang. RJSONIO: Serialize R objects to JSON, JavaScript Object Notation. http://www.omegahat.org/RJSONIO/, 2010. R package version 0.4-1.

[Ros06] Hans 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, 2006.

[RtICsg10] IBM Research and the IBM Cognos software group. Many eyes. http://services.alphaworks.ibm.com/manyeyes/page/Create_a_Visualization.html, 2010.

[Saa10] Sebastián Pérez Saaibi. *R/RMETRICS Generator Tool for Google Motion Charts*. https://www.rmetrics.org/, 2010. Meielisalp, Lake Thune Switzerland, June 27 - July 1, 2010.