MET CS688

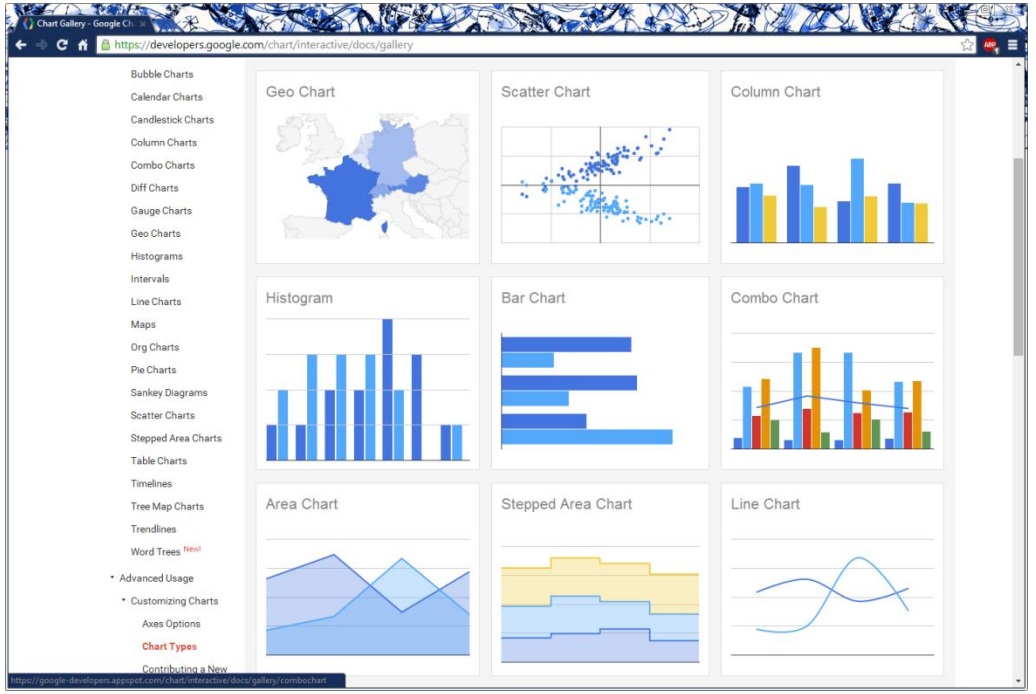
WEB ANALYTICS AND MINING

ZLATKO VASILKOSKI

GOOGLE VISUALIZATION

Data Visualization with Google Charts

- Google charts allow visualization of data on websites.
- The charts are based on HTML5/SVG and hence can be used in web pages without the need of plug-ins.
- The charts API makes it relatively straightforward to create interactive charts. The charts are embedded in the HTML pages and use JavaScript and JSON.
- A browser with an Internet connection is required to display the Google charts.
- The following link shows a snapshot of some of the charts provided by the charts API: https://developers.google.com/chart/interactive/docs/gallery



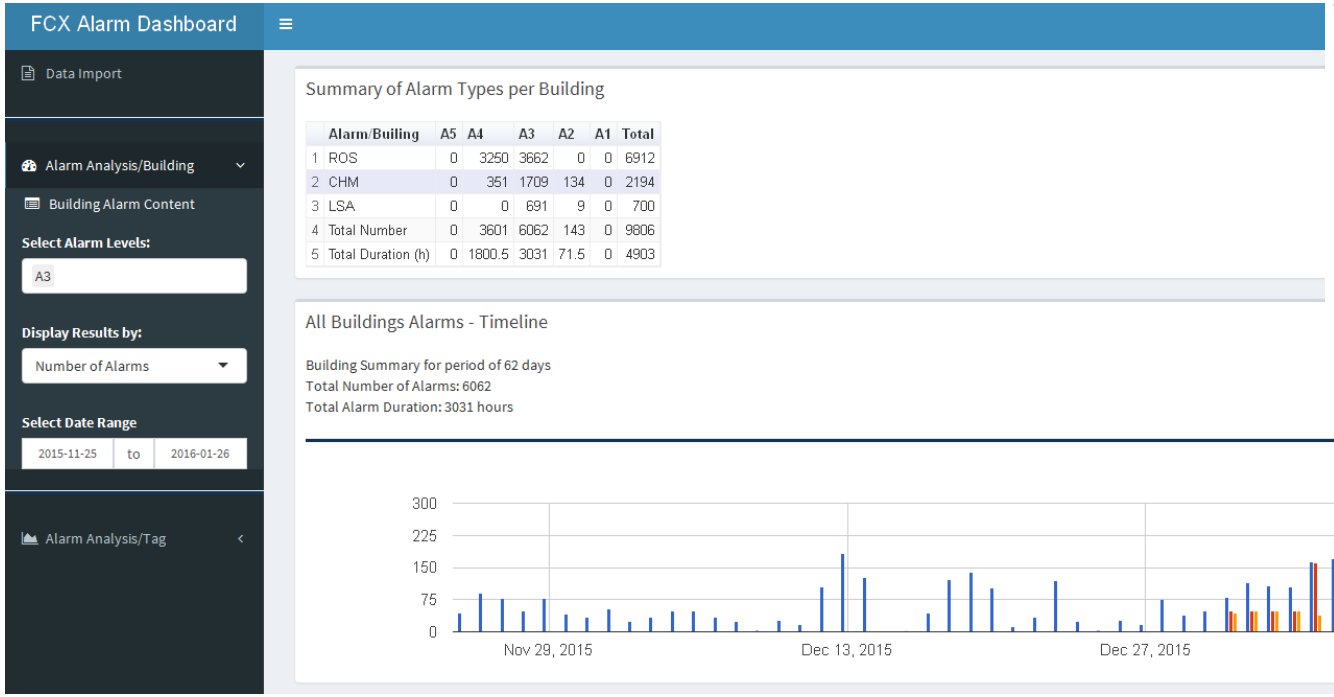
googleVis R Package

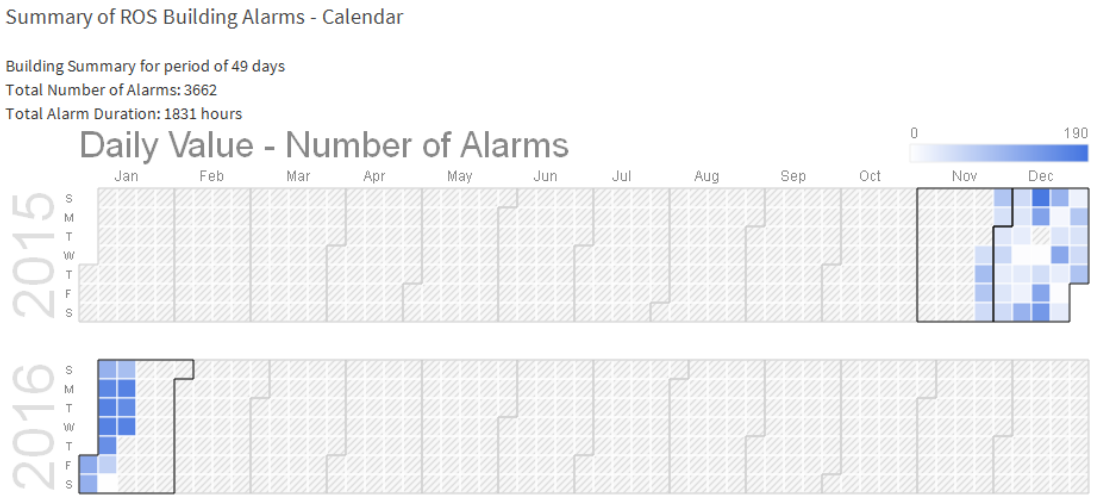
- The googleVis *R* package provides the interface between *R* and the Google charts.
- The googleVis package version 0.5.8 is used for this module, but you can download the latest.
- This package provides interfaces to HTML5/SVG-based and Flash-based visualizations.
  1. The HTML5/SVG-based charts include
     - line, bar, column, area,
     - combo, scatter, bubble,
     - candlestick, pie, organizational, tables,
     - gauges, tree maps, maps, geo charts and intensity maps.
  2. The Flash-based charts include
     - motion charts,
     - annotated time lines,
     - geo maps.
- The package version, along with its dependencies, needs to be installed for this module.

  ```
  > install.packages("googleVis")
  ```

Case Study

- Combining googleVis *R* package with shiny or shiny dashboard provides an easy and quick way of creating effective and interactive custom applications.





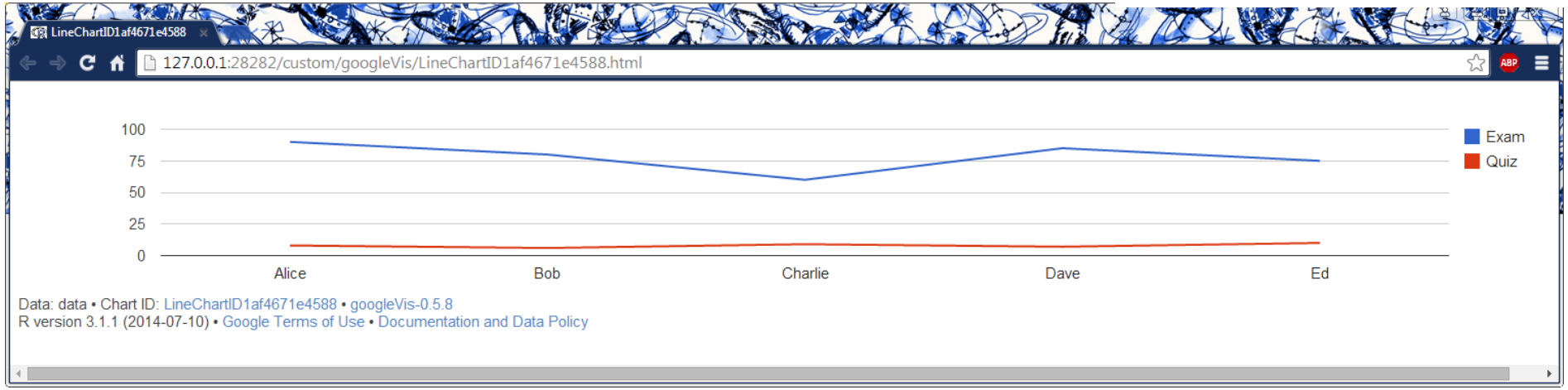
Line Chart

- The **gvisLineChart** function uses the specified data frame and creates the corresponding line chart.
- The data frame used in this example shows the scores of 5 students in an exam (out of 100) and in a quiz (out of 10).

```
> student.data
   Student Exam Quiz
1    Alice   90    8
2      Bob   80    6
3  Charlie   60    9
4     Dave   85    7
5       Ed   75   10
```

- The function is invoked with the default values for the rest of the parameters as shown below.

```
> chart1 <- gvisLineChart(student.data)
> plot(chart1)
```

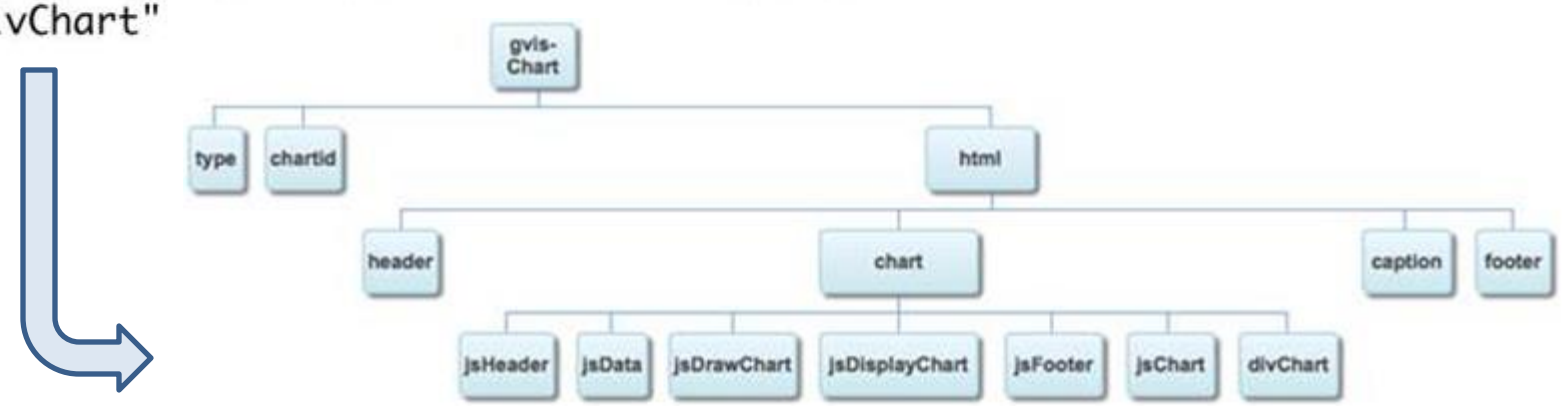


Line Chart Data Structure

- The data structure returned by any of the chart functions is a ***list*** with three components:
  - *type,*
  - *chartid,* and
  - *html*. The *html* component itself is again a list of four components:
    - *header,*
    - *chart,*
    - *caption,* and
    - *footer.*
- You can see some of the chart *HTML* components as

```
chart1$type
chart1$chartid
chart1$html$header
chart1$html$chart
chart1$html$caption
chart1$html$footer
```

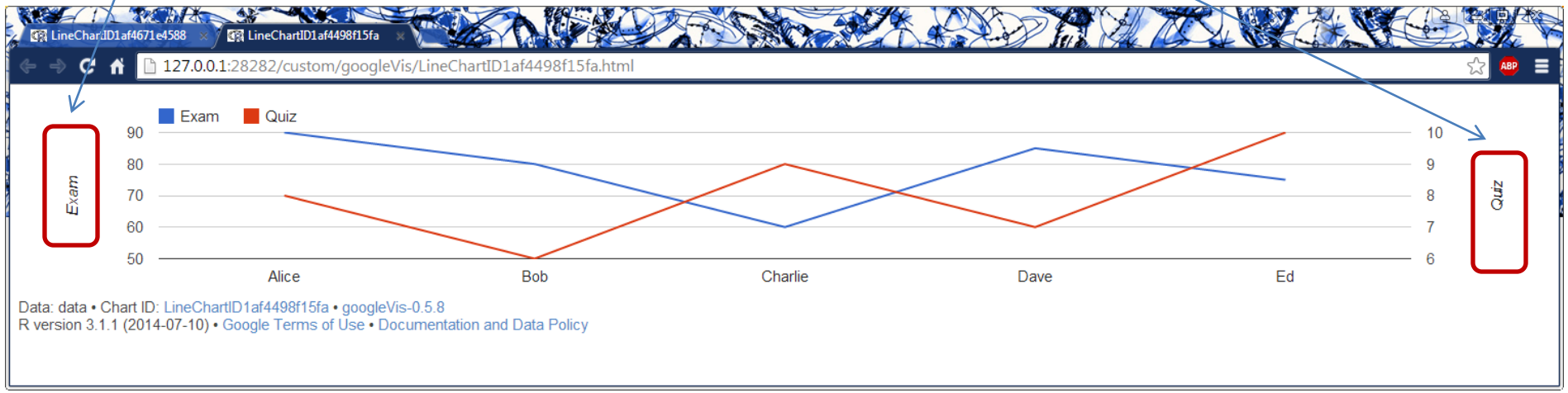
```
> names(chart1$html$chart)
[1] "jsHeader"       "jsData"        "jsDrawChart"
[4] "jsDisplayChart" "jsFooter"      "jsChart"
[7] "divChart"
```



Line Chart - Dual Axes

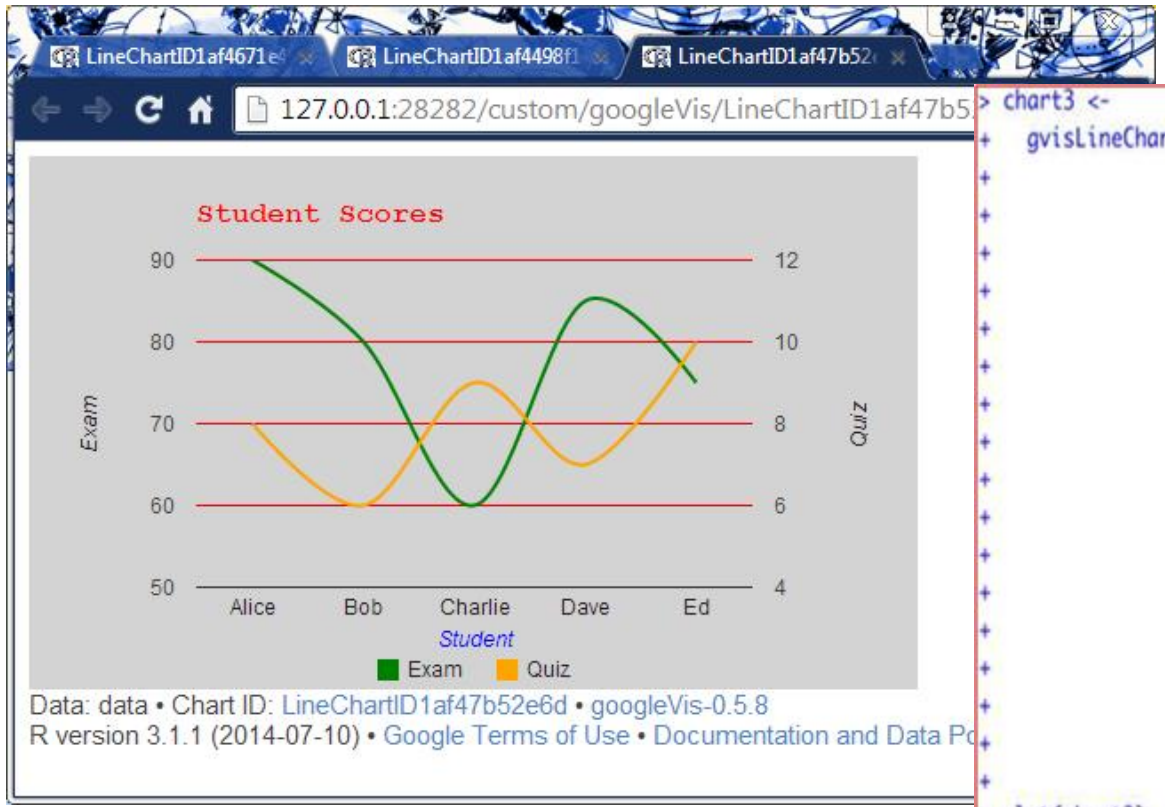
- Dual axes can be specified for the line chart using the *series* option as shown below.
- Straight lines connect the data points in the graph below.
- For more options see
  > ?gvisLineChart

```
> chart2 <-
+   gvisLineChart(student.data,
+                 "Student",
+                 c("Exam","Quiz"),
+                 options=list(
+                   series="[{targetAxisIndex: 0},
+                            {targetAxisIndex:1}]",
+                   vAxes="[{title:'Exam'}, {title:'Quiz'}]"
+                 ))
>
> plot(chart2)
```



Smoothed Line Chart

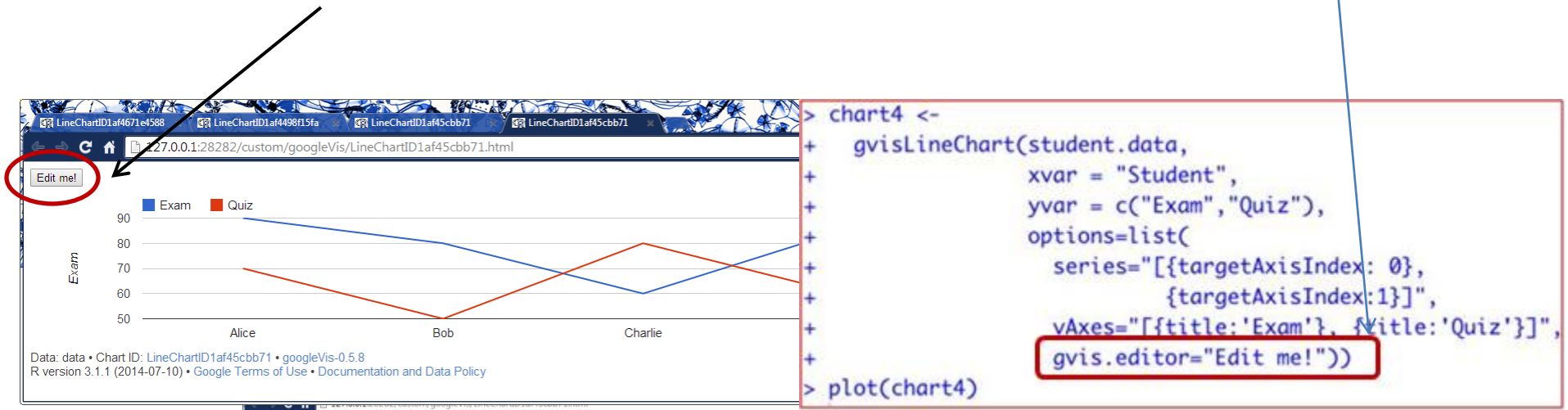
- The connecting points can be smoothed using the *curveType* option with the value *function*.
- Options may also include specific colors for the lines.

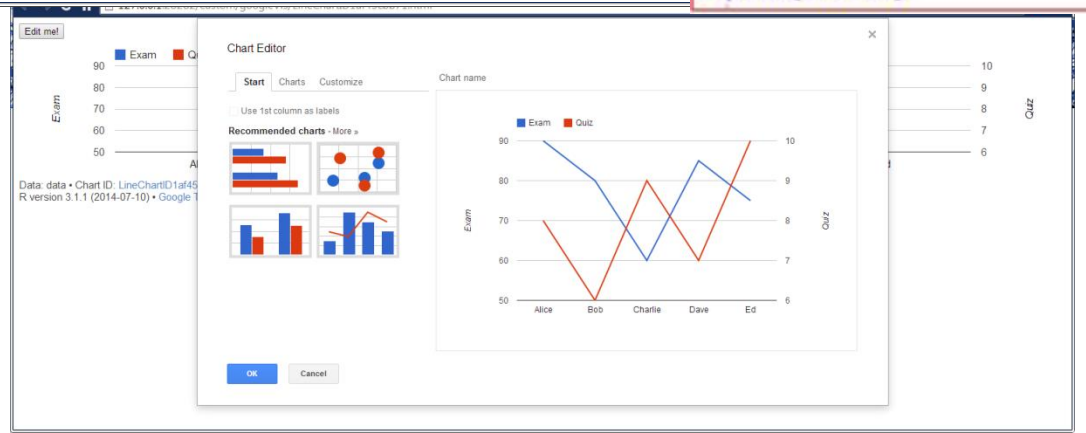



```
> chart3 <-
+   gvisLineChart(student.data,
+                 xvar="Student",
+                 yvar=c("Exam","Quiz"),
+                 options=list(
+                   title="Student Scores",
+                   titleTextStyle="{color:'red',
+                           fontName:'Courier',
+                           fontSize:16}",
+                   backgroundColor="#D3D3D3",
+                   vAxis="{gridlines:{color:'red', count:5}}",
+                   hAxis="{title:'Student', titleTextStyle:{color:'blue'}}",
+                   series="[{color:'green', targetAxisIndex: 0},
+                            {color: 'orange',targetAxisIndex:1}]",
+                   vAxes="[{title:'Exam'}, {title:'Quiz'}]",
+                   legend="bottom",
+                   curveType="function",
+                   width=500, height=300
+                         ))
> plot(chart3)
```

Editing Line Chart

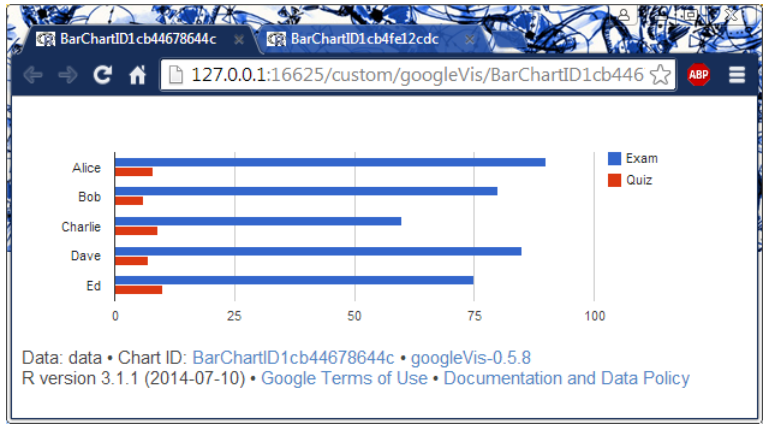
- All the charts may optionally specify the editing capability using the *gvis.editor* option.
- When the *edit* button is clicked, the user is provided the options to change the chart type and customization features.



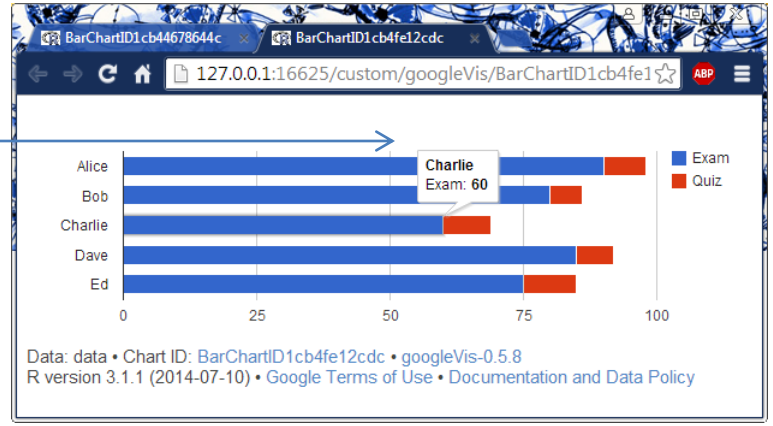


Bar Chart

- The **gvisBarChart** function uses the specified data frame and creates the corresponding bar chart with the horizontal bars.
- The data frame used in this example shows the scores of five students in an exam (out of 100) and in a quiz (out of 10).

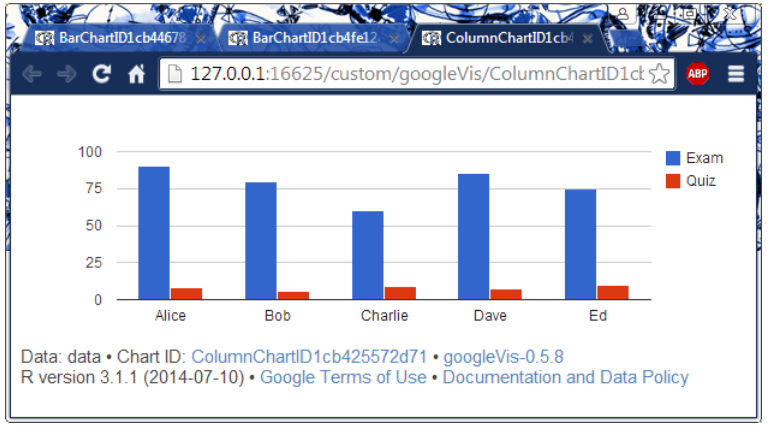


- A stacked bar chart can be created with the *isStacked* option set to *true*. The default value for stacking is *false.*
- You can get additional information if you go over particular data with your mouse

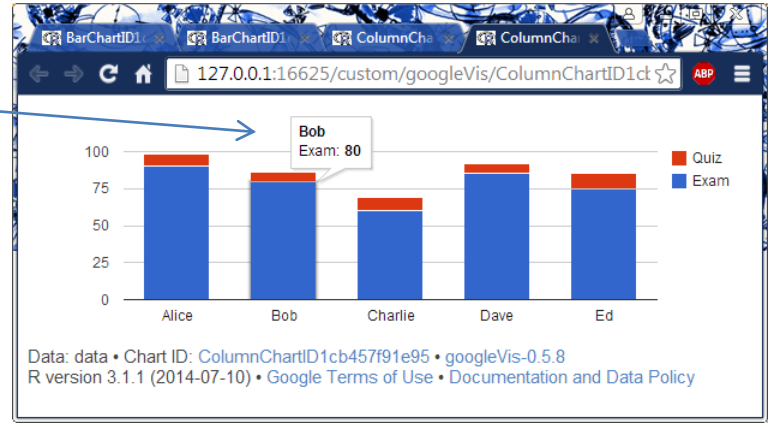


Column Chart

- The **gvisColumnChart** function uses the specified data frame and creates the corresponding column chart.
- Reminder: The data frame used in this example shows the scores of five students in an exam (out of 100) and in a quiz (out of 10).

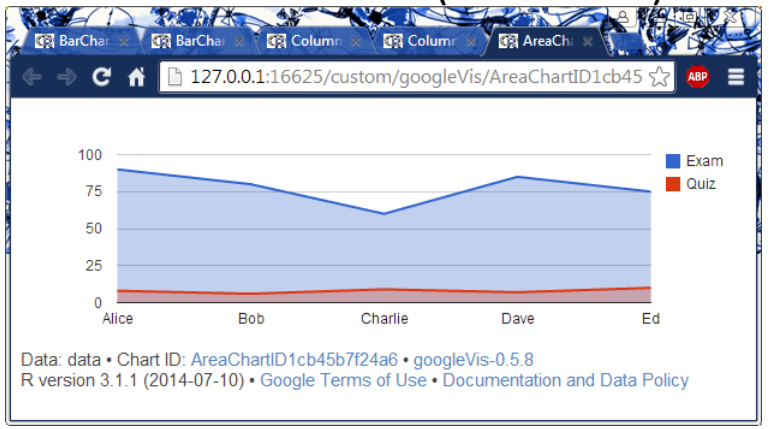


- A stacked column chart can be created with the *isStacked* option set to *true*. The default value for stacking is *false*
- You can get additional information if you go over particular data with your mouse

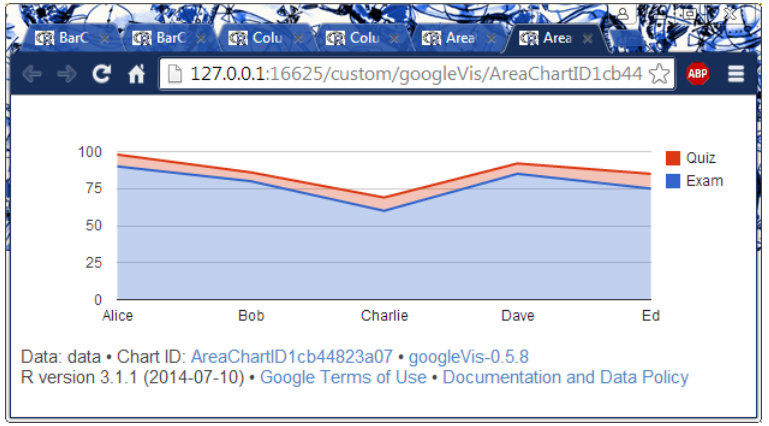


Area Chart

- The **gvisAreaChart** function uses the specified data frame and creates the corresponding area chart.
- Re: The data frame used in this example shows the scores of five students in an exam (out of 100) and in a quiz (out of 10).



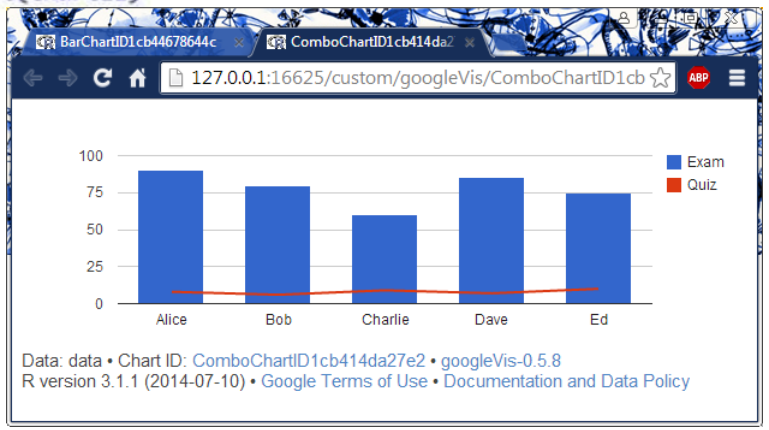
- A stacked area chart can be created with the *isStacked* option set to *true*. The default value for stacking is *false.*



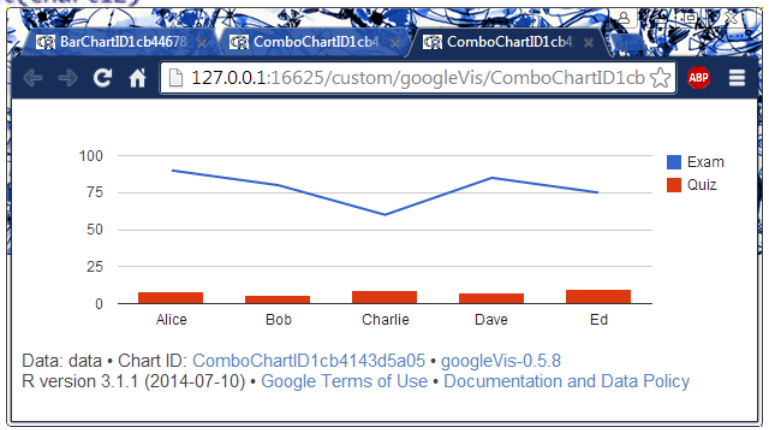
Combo Chart

- The **gvisComboChart** function uses the specified data frame and creates the corresponding combinations of charts as specified. The data frame used in this example is the same as before (scores of five students in an exam (out of 100) and in a quiz (out of 10)). The default type for each series in the data is the line chart. In the following example, the *seriesType* option is specified as *bars* and the second series (index 1) is to be shown as a line chart. The available options for the *type* are *line, area, bars, candlesticks,* and *steppedArea.*

```
> chart11 <-
+   gvisComboChart(student.data,
+                  xvar="Student",
+                  yvar=c("Exam", "Quiz"),
+                  options=list(seriesType="bars",
+                               series='{1: {type:"line"}}'))
> plot(chart11)
```

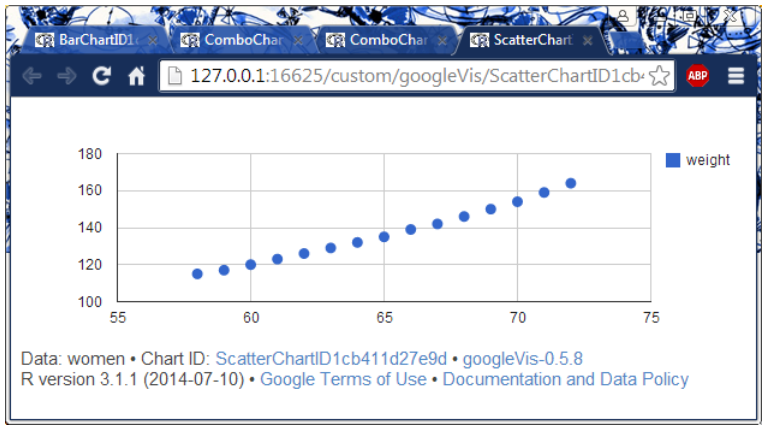



```
> chart12 <-
+   gvisComboChart(student.data,
+                  xvar="Student",
+                  yvar=c("Exam", "Quiz"),
+                  options=list(seriesType="line",
+                               series='{1: {type:"bars"}}'))
> plot(chart12)
```



Scatter Chart

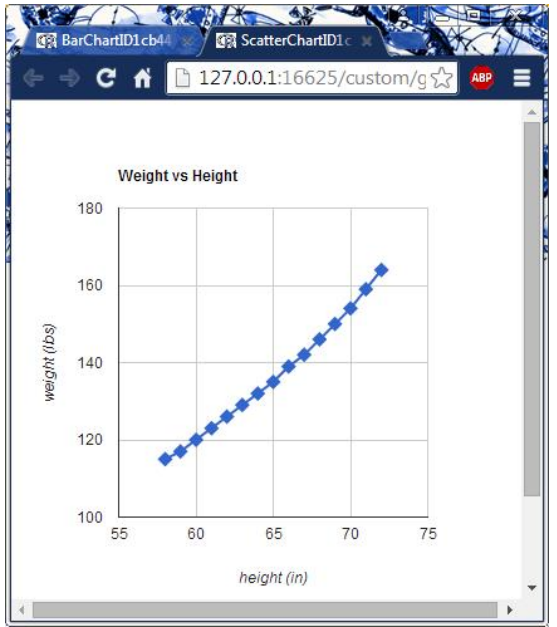
- The dataset ***women*** (provided in the *R* datasets package) shows the average heights and weights. The data frame has 15 observations. The first six rows in the data frame are shown below.




```
> head(women)
  height weight
1     58    115
2     59    117
3     60    120
4     61    123
5     62    126
6     63    129
```

```
> chart13 <- gvisScatterChart(women)
>
> plot(chart13)
```

- The points can be connected by lines by specifying the *lineWidth* option (default value is 0).
- The default value for *pointSize* is 7. The default *pointShape* is *circle* (other options are *triangle*, *square*, *diamond*, or *star*). The above scatter chart with different options is plotted as shown below.



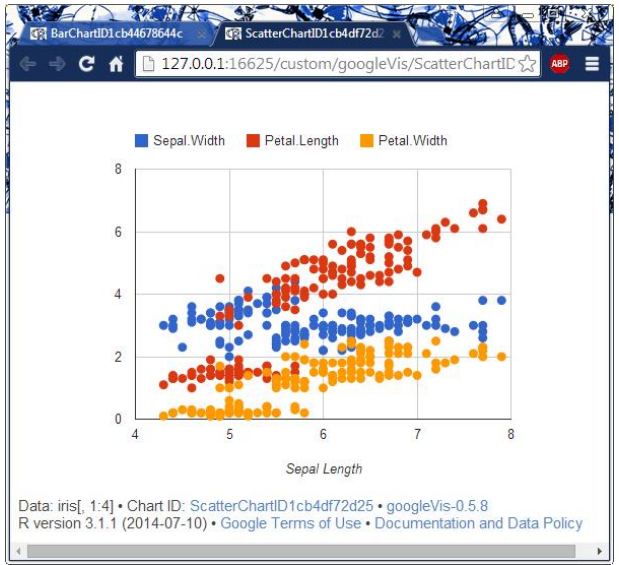
Scatter Chart

- The ***iris*** data set (provided in the *R* datasets package) gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of three species of *Iris*. The species are *Iris setosa*, *versicolor*, and *virginica*.

```
> chart15 <- gvisScatterChart(iris[,1:4],
+                             options=list(
+                               legend="top",
+                               hAxis="{title:'Sepal Length'}",
+                               width=600, height=400))
> plot(chart15)
```

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

- The legend is specified on the top.
- The title of the horizontal axis

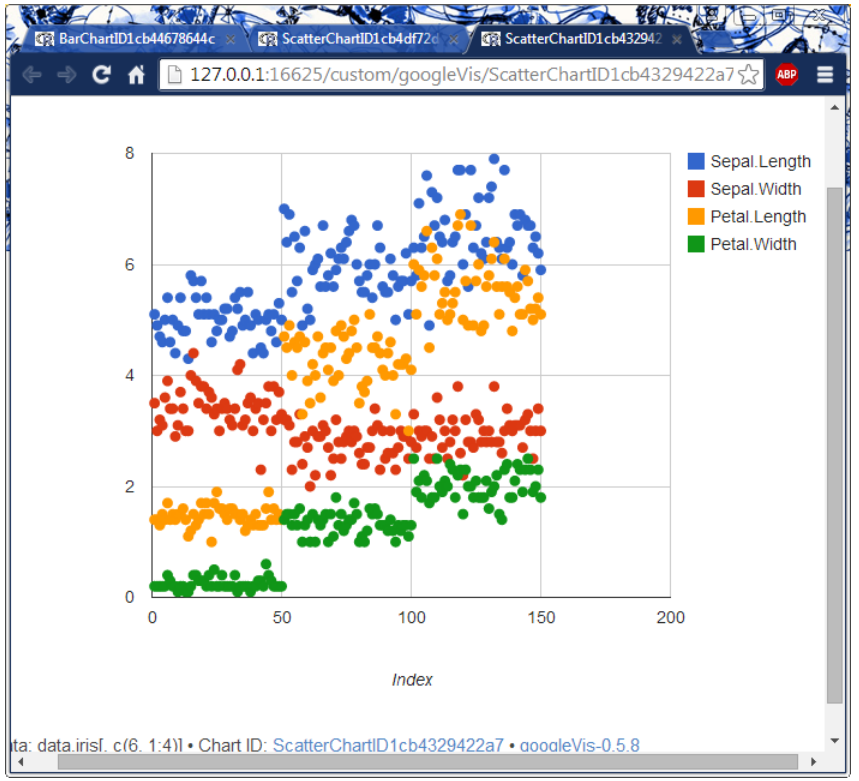


Scatter Chart

- For comparing all the four properties for each species, the data frame is modified by adding an *Index* column as shown below.

```
> data.iris <- iris
>
> data.iris$Index <- 1:nrow(iris)
>
> head(data.iris, n = 2)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species Index
1          5.1         3.5          1.4         0.2  setosa     1
2          4.9         3.0          1.4         0.2  setosa     2

> head(data.iris[, c(6, 1:4)], n = 2)
  Index Sepal.Length Sepal.Width Petal.Length Petal.Width
1     1          5.1         3.5          1.4         0.2
2     2          4.9         3.0          1.4         0.2
> chart15_1 <- gvisScatterChart(data.iris[, c(6, 1:4)],
+                               options=list(
+                                 hAxis="{title:'Index'}",
+                                 width=700, height=600))
> plot(chart15_1)
```

- The scatter plot comparing all the four properties is shown below. The first 50 values are for *Iris-setosa*, the next 50 values are for *Iris-versicolor*, and the last 50 values are for *Iris-virginica*.

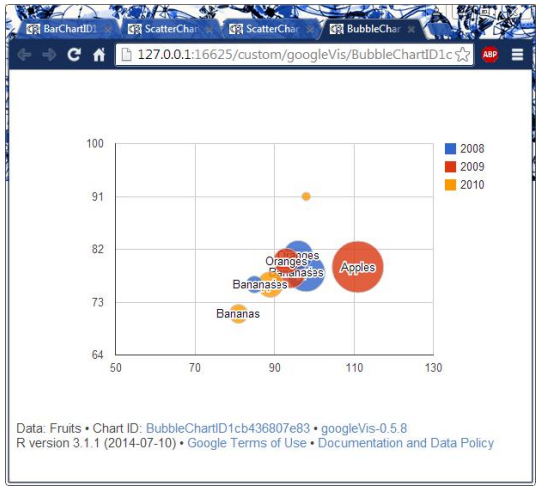


Bubble Chart

- The dataset ***Fruits*** (provided in the *R* googleVis package) shows the sales, expenses and profit for three types of fruit. The data frame has nine observations. The first six rows in the data frame are shown below.
- The **gvisBubbleChart** function uses the specified data frame and creates the corresponding bubble chart. The *idvar* argument is the column name of the data with the bubble. The *xvar* and *yvar* are the column names of the numerical data plotted on the x-axis and y-axis. The *colorvar* argument is the column name of the data that identifies bubbles that belong to the same series. Bubbles in the same series get the same color. The *sizevar* argument is the column name of the data that maps to the size of the bubble.

```
> head(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     Ea
6 Oranges 2009     Ea
```

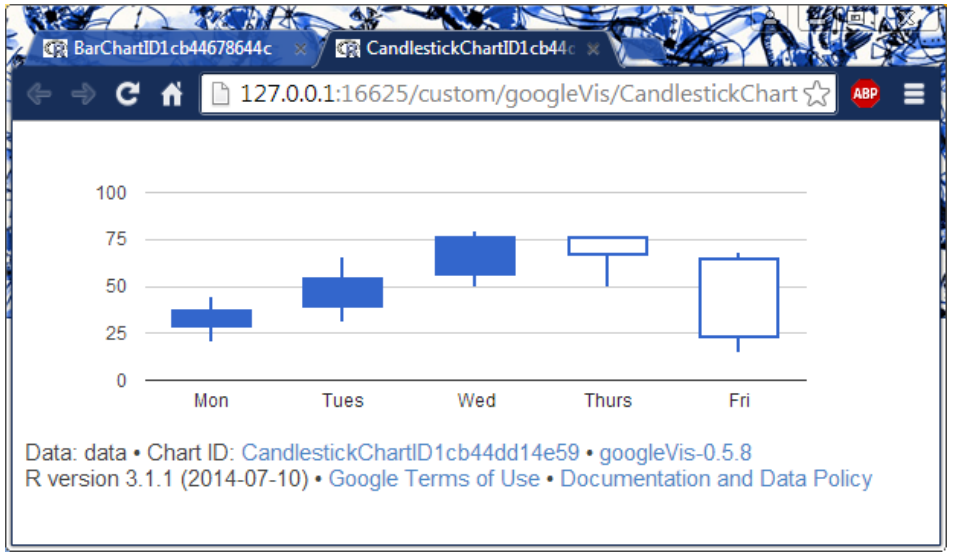
```
> chart16 <-
+   gvisBubbleChart(Fruits,
+                   idvar="Fruit",
+                   xvar="Sales", yvar="Expenses",
+                   colorvar="Year", sizevar="Profit",
+                   options=list(
+                     hAxis='{minValue:70, maxValue:120}',
+                     width=600, height=400))
> plot(chart16)
```



Candlestick Chart

- The dataset ***OpenClose*** (provided in the *R* googleVis package) shows the *open, close, low,* and *high* values of a stock for five days in a particular week. The data frame has five observations as shown below.
- The **gvisCandlestickChart** function uses the specified data frame and creates the corresponding candlestick chart. The *xvar* argument is the name of the character column that will show the labels for the x-axis. The ***low*** argument is the name of the numeric column that specifies the minimum value for the marker. This is the **base** of the candle's center line. The ***high*** argument is the name of the numeric column that specifies the **maximum** value for the marker. This is the **top** of the candle's center line. The ***open*** argument is the name of the numeric column that specifies the initial value for the marker. The ***close*** argument is the name of the numeric column that specifies the final value for the marker. If the *open* value is less than the *close* value, the candle will be **filled**. Otherwise, the candle will be **hollow**. For the above data, the first three candles are filled and the last two are hollow.

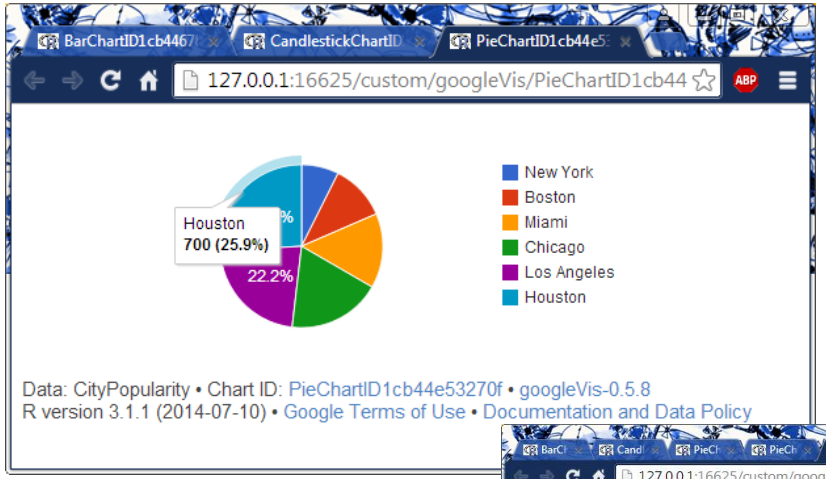
```
> OpenClose
  Weekday Low Open Close High
1     Mon  20   28    38   45
2    Tues  31   38    55   66
3     Wed  50   55    77   80
4   Thurs  50   77    66   77
5     Fri  15   66    22   68
> chart17 <-
+   gvisCandlestickChart(OpenClose, xvar="Weekday",
+                        low="Low", high="High",
+                        open="Open", close="Close",
+                         options=list(
+                           vAxis='{minValue:0, maxValue:100}',
+                           legend='none'))
>
> plot(chart17)
```



Pie Chart

- The dataset ***CityPopularity*** (provided in the *R* googleVis package) shows the *city and popularity* values for six cities. The data frame has six observations as shown below.

```
> CityPopularity
         City Popularity
1    New York        200
2      Boston        300
3       Miami        400
4     Chicago        500
5 Los Angeles        600
6     Houston        700
```



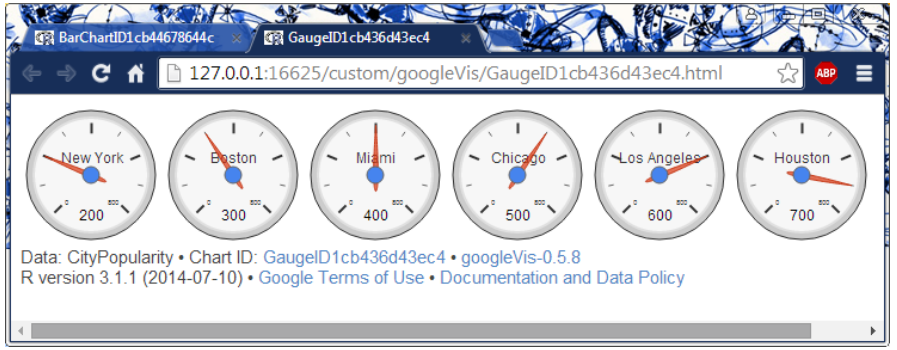
- The donut-shaped pie chart with offsets for the individual slices may also be included in the list of the options for the pie chart.
- **Note the indexing**: the offsets for the slices 1 and 3 correspond to the cities *Boston* and *Chicago* (0 being *New York*).

- A 3D version of the pie chart can also be specified.

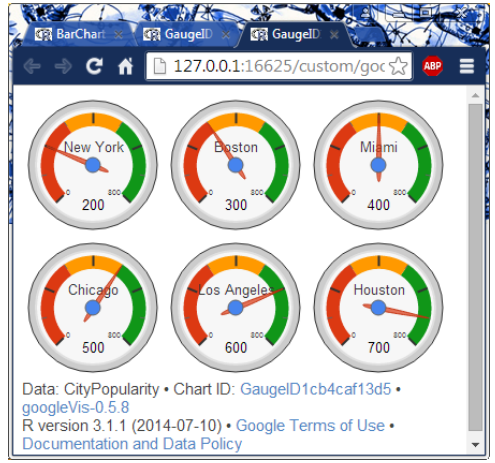
Gauge Chart

- The **gvisGauge** function uses the specified data frame and creates the corresponding gauge with dial. The optional arguments include *labelvar*, the name of the column that contains the labels for the gauge meters, and *numvar*, the column name for the dial values. The default values for *min* and *max* are 0 and 100, respectively.

```
> chart20 <-
+   gvisGauge(CityPopularity,
+             options=list(min=0, max=800))
> plot(chart20)
```



- The options also allow to configure the *red*, *yellow*, and *green* regions of the gauges. The *redFrom* and *redTo* are the lowest and highest values of a range marked by red color. Similalry, the *yellowFrom, yellowTo, greenFrom,* and *greenTo* parameters. The default colors are used for the options *redColor*, *yellowColor*, and *greenColor*.



Histogram Chart

- The **gvisHistogram** function uses the specified data frame and creates the corresponding histogram. The *iris* data set is used in the following example. Each column in the dataset will be displayed as a histogram.

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

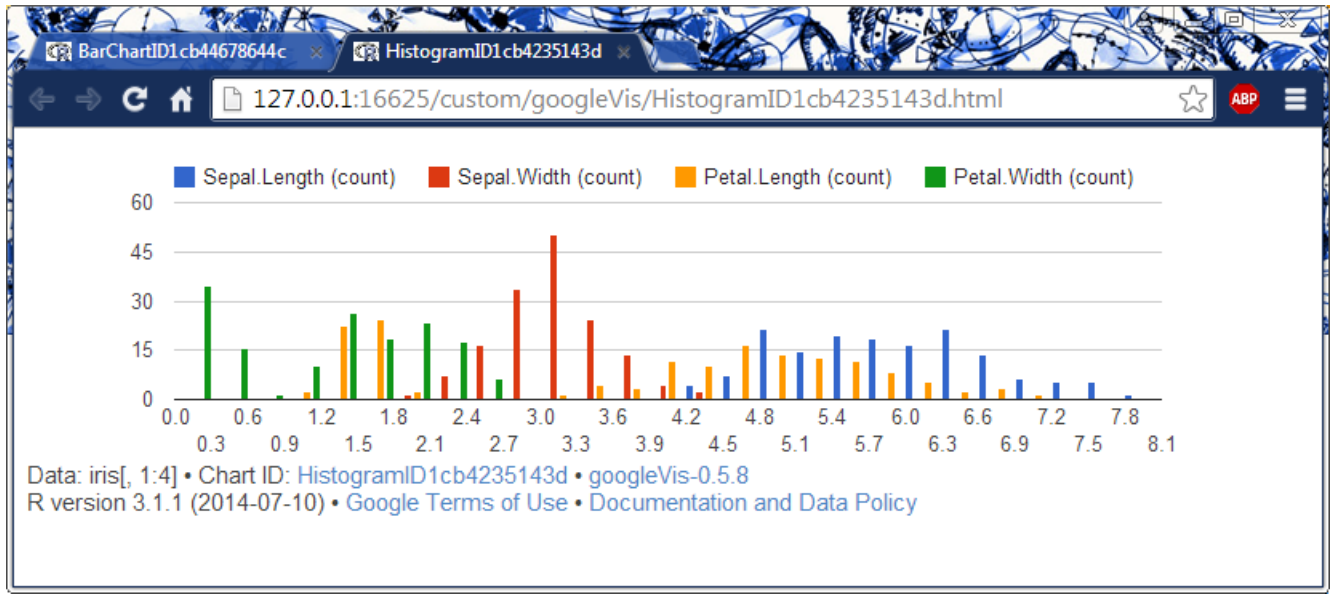


Table Chart

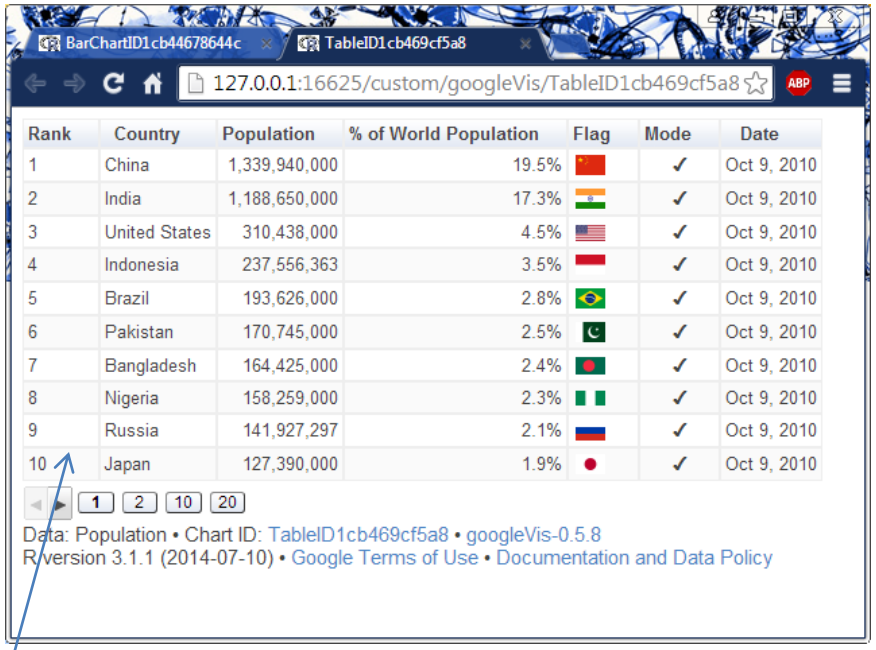
- The dataset ***Stock*** (provided in the *R* googleVis package) shows the stock of pencils and pens. The data frame has 12 observations; the first six are shown below.

```
> head(Stock)
        Date  Device  Value          Title          Annotation
1 2008-01-01 Pencils   3000           <NA>                <NA>
2 2008-01-02 Pencils  14045           <NA>                <NA>
3 2008-01-03 Pencils   5502           <NA>                <NA>
4 2008-01-04 Pencils  75284           <NA>                <NA>
5 2008-01-05 Pencils  41476 Bought pencils Bought 200k pencils
6 2008-01-06 Pencils 333222           <NA>                <NA>

> chart23 <-
+   gvisTable(Stock,
+             formats=list(Value="#,###"))
>
> plot(chart23)
```

Table Chart

- The dataset ***Population*** (provided in *R* googleVis package) shows the population information for various countries. The data frame has 195 observations and seven columns. The first four and the last two columns for the first six observations are shown below.

```
> names(Population)
[1] "Rank"                 "Country"              "Population"
[4] "% of World Population" "Flag"                 "Mode"
[7] "Date"
> head(Population[, c(1:4, 6:7)])
  Rank       Country Population % of World Population Mode       Date
1    1         China 1339940000                0.1950 TRUE 2010-10-09
2    2         India 1188650000                0.1730 TRUE 2010-10-09
3    3 United States  310438000                0.0452 TRUE 2010-10-09
4    4     Indonesia  237556363                0.0346 TRUE 2010-10-09
5    5        Brazil  193626000                0.0282 TRUE 2010-10-09
6    6      Pakistan  170745000                0.0248 TRUE 2010-10-09
> chart24 <-
+   gvisTable(Population,
+             formats=list(Population="#,###",
+                          '% of World Population'='#.#%'),
+             options=list(page='enable'))
> plot(chart24)
```



Rank	Country	Population	% of World Population	Flag	Mode	Date
1	China	1,339,940,000	19.5%		✓	Oct 9, 2010
2	India	1,188,650,000	17.3%		✓	Oct 9, 2010
3	United States	310,438,000	4.5%		✓	Oct 9, 2010
4	Indonesia	237,556,363	3.5%		✓	Oct 9, 2010
5	Brazil	193,626,000	2.8%		✓	Oct 9, 2010
6	Pakistan	170,745,000	2.5%		✓	Oct 9, 2010
7	Bangladesh	164,425,000	2.4%		✓	Oct 9, 2010
8	Nigeria	158,259,000	2.3%		✓	Oct 9, 2010
9	Russia	141,927,297	2.1%		✓	Oct 9, 2010
10	Japan	127,390,000	1.9%		✓	Oct 9, 2010

- The above data frame is rendered as shown below. The *formats* argument can be used to specify the formatting for the desired columns. The *paging* option can also be enabled for the table. The default *pageSize* option is 10. The default value for the *sort* option for the columns is *enable*.

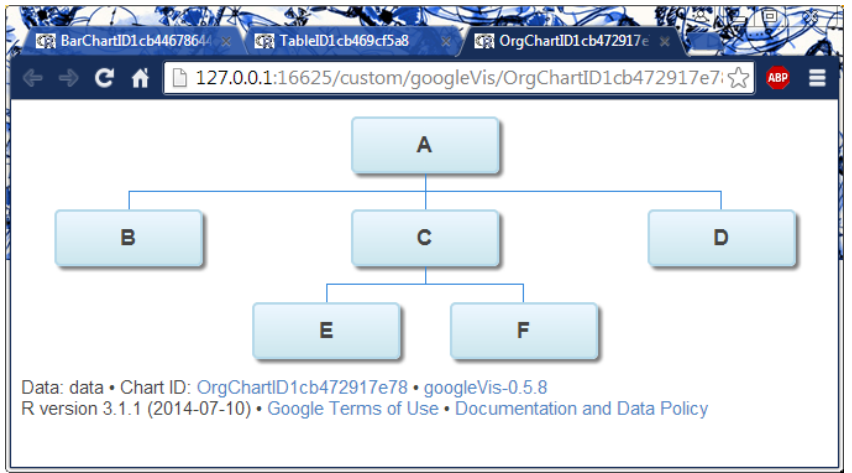
Organizational Chart

- The organizational charts (Org Charts) show the hierarchical relationship between a set of nodes. The following example uses a data frame show the relationship between the node and who the parent of the node is. The root node's parent is NA.

```
> data = data.frame(
+   Node =  c("A", "B", "C", "D", "E", "F"),
+   Parent = c(NA, "A", "A", "A", "C", "C"),
+   val = 1:6
+ )
>
> data
  Node Parent val
1    A   <NA>   1
2    B      A   2
3    C      A   3
4    D      A   4
5    E      C   5
6    F      C   6
```

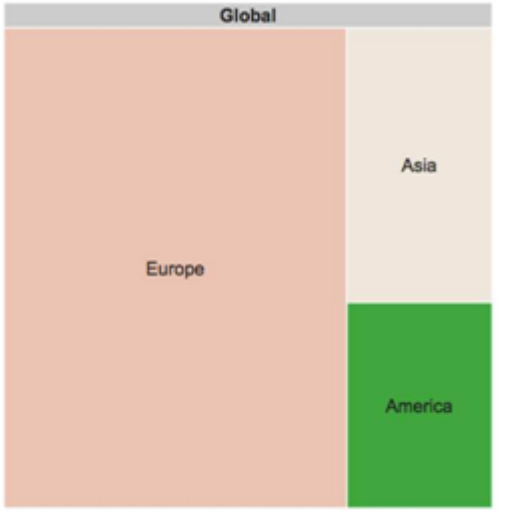
```
> chart25 <-
+   gvisOrgChart(data,
+                idvar="Node",
+                parentvar="Parent",
+                tipvar="val",
+                options=list(width=600, height=250,
+                             size='large',
+                             allowCollapse=TRUE))
>
> plot(chart25)
```

- The **gvisOrgChart** function uses the specified data frame and creates the corresponding hierarchical chart. The data frame needs to have at least three columns.

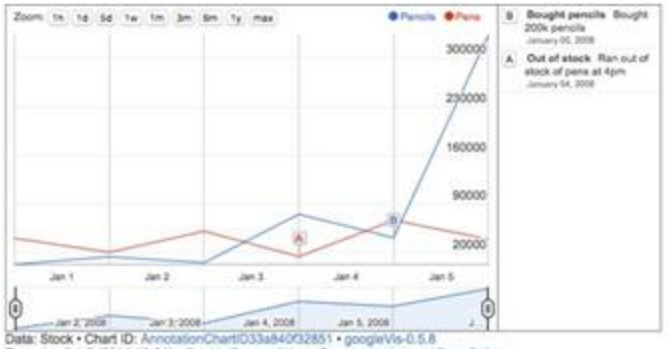


More Charts

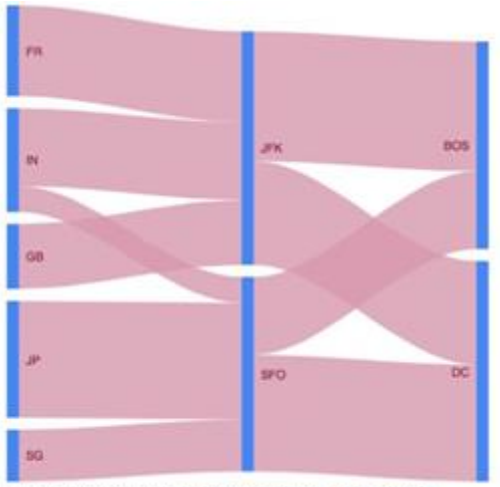
- Tree Map
- Annotation Chart
- Sankey Diagram
- Geo Chart
- Map Chart
- Calendar Chart
- Timeline Chart



Data: Regions • Chart ID: TreeMapID33a85b0185bc • googleVis-0.5.8
R version 3.1.2 (2014-10-31) • Google Terms of Use • Documentation and Data Policy



Data: Stock • Chart ID: AnnotationChartID33a840f32851 • googleVis-0.5.8
R version 3.1.2 (2014-10-31) • Google Terms of Use • Documentation and Data Policy



Data: data.SK • Chart ID: SankeyID33a81bdbff27 • googleVis-0.5.8
R version 3.1.2 (2014-10-31) • Google Terms of Use • Documentation and Data Policy

Data: Exports • Chart ID: GeoChartID33a884c67bb • googleVis-0.5.8
R version 3.1.2 (2014-10-31) • Google Terms of Use • Documentation and Data Policy

17. Annotation Chart

- The annotation charts are interactive time series line charts and support annotations on the charts.
- The dataset *Stock* (provided in R googleVis package) shows the stock of pencils and pens. The data frame has 12 observations; the first 6 are shown below.

```
> head(Stock)
        Date  Device  Value          Title          Annotation
1 2008-01-01 Pencils   3000           <NA>                <NA>
2 2008-01-02 Pencils  14045           <NA>                <NA>
3 2008-01-03 Pencils   5502           <NA>                <NA>
4 2008-01-04 Pencils  75284           <NA>                <NA>
5 2008-01-05 Pencils  41476 Bought pencils Bought 200k pencils
6 2008-01-06 Pencils 333222           <NA>                <NA>
```

- The gvisAnnotationChart function uses the specified data frame and creates the corresponding annotation chart. The data frame needs to have at least **two columns**, one with data information (datevar) and one with numeric data. The datevar argument is the column name of data showing the date dimension. The numvar argument is the column name of data showing the values to be displayed against the date dimension. The idvar argument is the column name of data identifying different groups of the data. The titlevar argument is the column name of the data that shows the title of the annotations. The annotationvar argument is the name of the column with the annotation text.

17. Annotation Chart

- The corresponding annotation chart is shown in the web browser. The data frame has two groups of data, one for pencils and the other for pens. The chart displays the tooltips when hovering over the data points.


```
> chart28 <-
+   gvisAnnotationChart(Stock,
+                       datevar="Date",
+                       numvar="Value",
+                       idvar="Device",
+                       titlevar="Title",
+                       annotationvar="Annotation")
>
> plot(chart28)
```

Google Annotation Chart Exercise

Create an Annotation chart using the posted code

- Analyze how the “Stock” data is organized to create the chart
- Analyze how the Title and Annotation columns are organized and related to the chart
- Copy the Stock data into “MyStock” and
  - Replace Pens with Erasers
  - Move annotation “A” (Out of stock…) to Jan 02 2008
  - Display your chart

Note: Do not do the replacement by hard coding the indices!

Find the indices that need to be replaced. For example:

```
ix <- MyStock$Device == "Pens" # Find indices where Pens appear
```

```
> Stock
         Date  Device  Value          Title                          Annotation
1  2008-01-01 Pencils   3000           <NA>                                <NA>
2  2008-01-02 Pencils  14045           <NA>                                <NA>
3  2008-01-03 Pencils   5502           <NA>                                <NA>
4  2008-01-04 Pencils  75284           <NA>                                <NA>
5  2008-01-05 Pencils  41476 Bought pencils            Bought 200k pencils
6  2008-01-06 Pencils 333222           <NA>                                <NA>
7  2008-01-01    Pens  40645           <NA>                                <NA>
8  2008-01-02    Pens  20374           <NA>                                <NA>
9  2008-01-03    Pens  50766           <NA>                                <NA>
10 2008-01-04    Pens  14334   Out of stock Ran out of stock of pens at 4pm
11 2008-01-05    Pens  66467           <NA>                                <NA>
12 2008-01-06    Pens  39463           <NA>                                <NA>
>
```

18. Sankey Diagram

- A sankey diagram shows the flow of information from one set of values to another. The components being connected are called nodes and the connections are called links. Consider the following sample data set that shows the relative numbers of passengers arriving from various countries to the airports JFK and SFO. From JFK and SFO, the passengers go to BOS or DC.

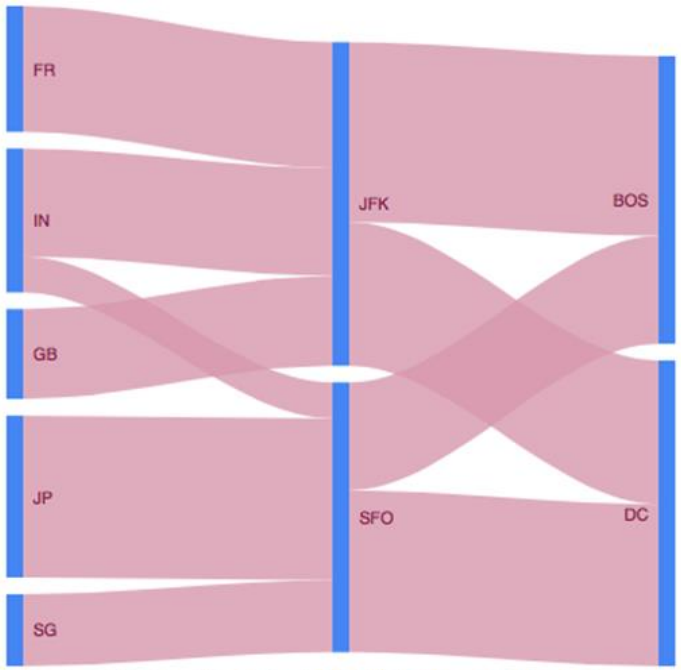
```
> data.SK <- data.frame(
+   From=c('GB','FR','IN', 'IN','JP','SG','JFK','JFK','SFO','SFO'),
+   To=c(rep('JFK', 3), rep('SFO', 3), rep(c('BOS', 'DC'),2)),
+   Weight=c(5,7,6,2,9,4, 10,8, 6,9))
```

```
> data.SK
   From  To Weight
1    GB JFK      5
2    FR JFK      7
3    IN JFK      6
4    IN SFO      2
5    JP SFO      9
6    SG SFO      4
7   JFK BOS     10
8   JFK  DC      8
9   SFO BOS      6
10  SFO  DC      9
```

- The *gvisSankey* function uses the specified data frame and creates the corresponding sankey diagram. The *from* argument is the column name of data for the source nodes. The *to* argument is the column name of data for the destination nodes. The *weight* argument is the column with the numerical weight of the connections.
- Named after Irish Captain Matthew Henry Phineas Riall Sankey, who used this type of diagram in 1898 for showing the energy efficiency of a steam engine.

18. Sankey Diagram

- The corresponding sankey diagram is shown in the web browser. The chart displays the connected regions when hovering over the nodes and the links.



Data: data.SK • Chart ID: SankeyID33a81bdbff27 • googleVis-0.5.8
R version 3.1.2 (2014-10-31) • Google Terms of Use • Documentation and Data Policy

```
> chart29 <-
+   gvisSankey(data.SK,
+              from="From", to="To", weight="Weight",
+              options=list(
+                sankey="{link: {color: { fill: '#d799ae' } },
+                                node: { color: { fill: '#00ff00' },
+                                label: { color: '#871b47' } }}"))
> plot(chart29)
```

19. Geo Chart

- The geo chart is a map of a country, a continent, or a region. The areas on the map are identified with one of the modes: *region,* colors whole regions such as countries, states, or provinces; *markers*, uses circles to designate regions, scaled to the specified value; *text,* labels the regions with identifiers.

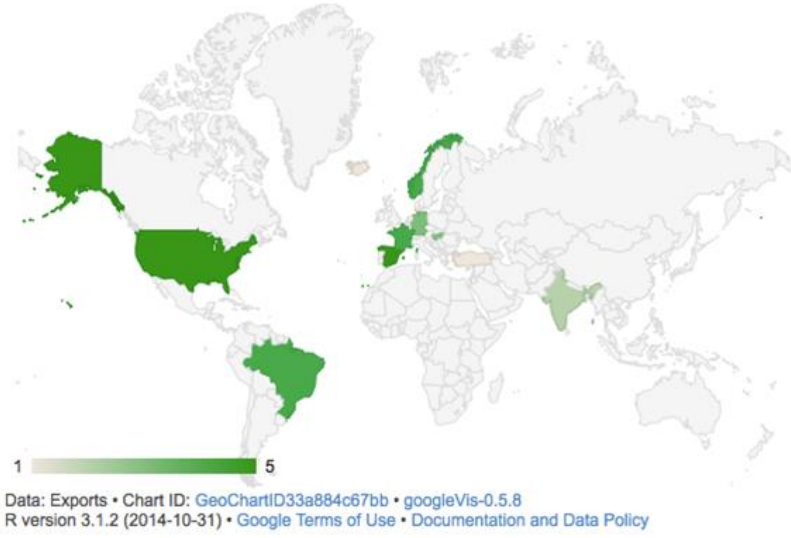
- The dataset *Exports* (provided in R googleVis package has 10 observations as shown below.

```
> Exports
         Country Profit Online
1        Germany      3   TRUE
2         Brazil      4  FALSE
3  United States      5   TRUE
4         France      4   TRUE
5        Hungary      3  FALSE
6          India      2   TRUE
7        Iceland      1  FALSE
8         Norway      4   TRUE
9          Spain      5   TRUE
10        Turkey      1  FALSE
```

19. Geo Chart

- The *gvisGeoChart* function uses the specified data frame and creates the corresponding geo chart. The *locationvar* argument is the column name of data with the geo locations. The locations can be provided in the *latitude:longitude* format, or as the country names and codes. The *colorvar* argument is the column name of data with values used for assigning color to the marker. The color values are scaled relative to each other If the *displayMode* option is *markers*, the *sizevar* argument is used for the size of the markers. The *hovervar* argument can be used to specify the column data that should displayed when the user hovers over a region.
- The corresponding geo chart is shown in the web browser. The chart displays the tooltips when hovering over the regions.

```
> chart30 <-
+   gvisGeoChart(Exports,
+                locationvar="Country",
+                colorvar="Profit")
>
> plot(chart30)
```



19. Geo Chart

- The dataset *state.x77* (provided in R datasets package) is a matrix showing the data for the 50 states (US). The column names in the dataset are shown below.
- The column names do not include the state information. The *rownames* of the matrix give the required information.
- Since the data frame is to be supplied as the argument for the chart, a data frame is created with the *State* column and the first two columns of the *state.x77* matrix as shown below.

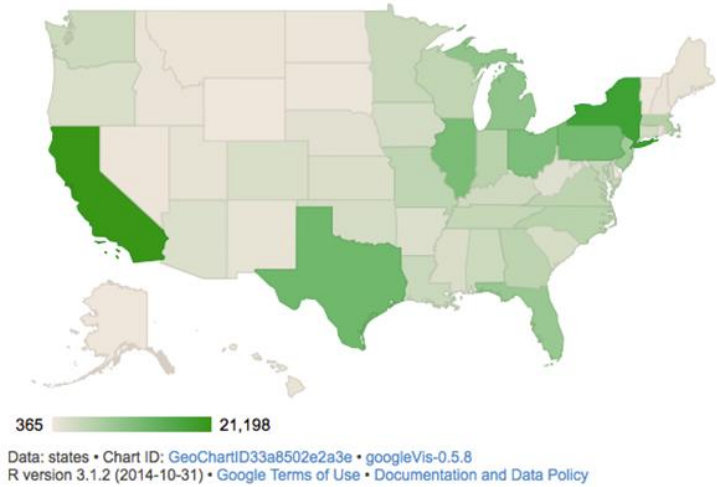
```
> library(datasets)
>
> colnames(state.x77)
[1] "Population" "Income"     "Illiteracy" "Life Exp"
[5] "Murder"     "HS Grad"    "Frost"      "Area"

> rownames(state.x77)
 [1] "Alabama"        "Alaska"         "Arizona"
 [4] "Arkansas"       "California"     "Colorado"
 [7] "Connecticut"    "Delaware"       "Florida"
[10] "Georgia"        "Hawaii"         "Idaho"
[13] "Illinois"       "Indiana"        "Iowa"
[16] "Kansas"         "Kentucky"       "Louisiana"
[19] "Maine"          "Maryland"       "Massachusetts"
[22] "Michigan"       "Minnesota"      "Mississippi"
[25] "Missouri"       "Montana"        "Nebraska"
[28] "Nevada"         "New Hampshire"  "New Jersey"
[31] "New Mexico"     "New York"       "North Carolina"
[34] "North Dakota"   "Ohio"           "Oklahoma"
[37] "Oregon"         "Pennsylvania"   "Rhode Island"
[40] "South Carolina" "South Dakota"   "Tennessee"
[43] "Texas"          "Utah"           "Vermont"
[46] "Virginia"       "Washington"     "West Virginia"
[49] "Wisconsin"      "Wyoming"
```

```
> states <- data.frame("State" = rownames(state.x77),
+                      state.x77[, c(1,2)])
>
> head(states)
                State Population Income
Alabama       Alabama       3615   3624
Alaska         Alaska        365   6315
Arizona       Arizona       2212   4530
Arkansas     Arkansas       2110   3378
California California      21198   5114
Colorado     Colorado       2541   4884
```

19. Geo Chart

- The geo chart is plotted using the code shown below. The *region* option is “US”, the *displayMode* option is *regions* and the *resolution* is *provinces*. The state regions are colored by the *population*.
- The corresponding geo chart is shown in the web browser. The chart displays the tooltips when hovering over the regions.

```
> states <- data.frame("State" = rownames(state.x77),
+                      state.x77[, c(1,2)])
>
> head(states)
                State Population Income
Alabama       Alabama       3615   3624
Alaska         Alaska        365   6315
Arizona       Arizona       2212   4530
Arkansas     Arkansas       2110   3378
California California      21198   5114
Colorado     Colorado       2541   4884
```

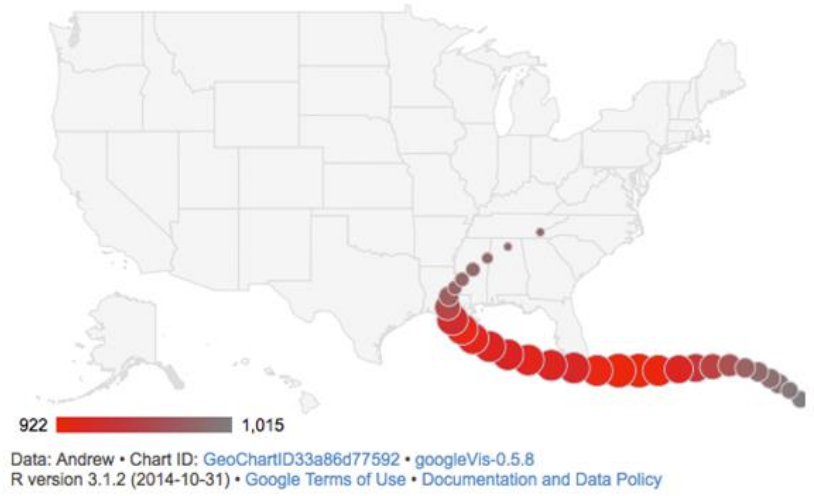


19. Geo Chart

- The dataset *Andrew* (provided in R googleVis package) shows the storm path of hurricane Andrew from August 16th to August 28th, 1992. The data frame has 47 observations with the columns shown below.
- The data for the first three observations for the columns used in the chart are shown below.
- The geo chart is plotted using the code shown below. The *region* option is "US", the *displayMode* option is *Markers* and the *resolution* is *provinces*. The states are colored relative to the values in the *Speed_kt* column in the range from *red* to *grey*. The size of the markers is determined by the values in the *Pressure_mb* column. The chart displays the tooltips when hovering over the markers.

```
> names(Andrew)
[1] "Date/Time UTC" "Lat"           "Long"
[4] "Pressure_mb"   "Speed_kt"      "Category"
[7] "LatLong"       "Tip"
```

```
> Andrew[1:3, c("LatLong","Speed_kt", "Pressure_mb")]
     LatLong Speed_kt Pressure_mb
1 10.8:-35.5       25        1010
2 11.2:-37.4       30        1009
3 11.7:-39.6       30        1008
```



20. Map Chart

- The *gvisMap* function uses the specified data frame and creates the corresponding map using the Google Maps API. The *locationvar* argument is the column name of data with the geo locations. The locations can be provided in the *latitude:longitude* format, or as the country names and codes. The data values are shown as markers on the map.The *tipvar* argument is the column name of data with values used for the tooltips.
-
- The dataset *Andrew* (provided in R googleVis package) shows the storm path of hurricane Andrew from August 16th to August 28th, 1992. The data for the first three observations for the columns used in the chart are shown below.

```
> Andrew[1:3, c("LatLong","Tip")]
     LatLong                                                    Tip
1 10.8:-35.5 Tropical Depression<BR>Pressure=1010<BR>Speed=25
2 11.2:-37.4 Tropical Depression<BR>Pressure=1009<BR>Speed=30
3 11.7:-39.6 Tropical Depression<BR>Pressure=1008<BR>Speed=30
```

```
> chart33 <-
+   gvisMap(Andrew,
+           locationvar = "LatLong" ,
+           tipvar = "Tip",
+           options=list(
+             showTip=TRUE,
+             showLine=TRUE,
+             enableScrollWheel=TRUE,
+             mapType='terrain',
+             useMapTypeControl=TRUE))
>
> plot(chart33)
```

20. Map Chart

- The corresponding map chart is shown in the web browser. The chart displays the tooltips when hovering over the data points.

20. Map Chart

- The population of the top 10 populous countries is available from the *Population* dataset as shown below.
- The above data is plotted on the map as shown below.

```
> Population[1:10, c("Country", "Population")]
         Country Population
1          China 1339940000
2          India 1188650000
3  United States  310438000
4      Indonesia  237556363
5         Brazil  193626000
6       Pakistan  170745000
7     Bangladesh  164425000
8        Nigeria  158259000
9         Russia  141927297
10         Japan  127390000
```

```
> chart33_1 <-
+   gvisMap(Population[1:10,],
+           locationvar = "Country" ,
+           tipvar = "Population",
+           options=list(
+             showTip=TRUE,
+             mapType='terrain',
+             useMapTypeControl=TRUE))
>
> plot(chart33_1)
```

20. Map Chart

- The corresponding map chart is shown in the web browser. The chart displays the tooltips when hovering over the data points.

21. Calendar Chart

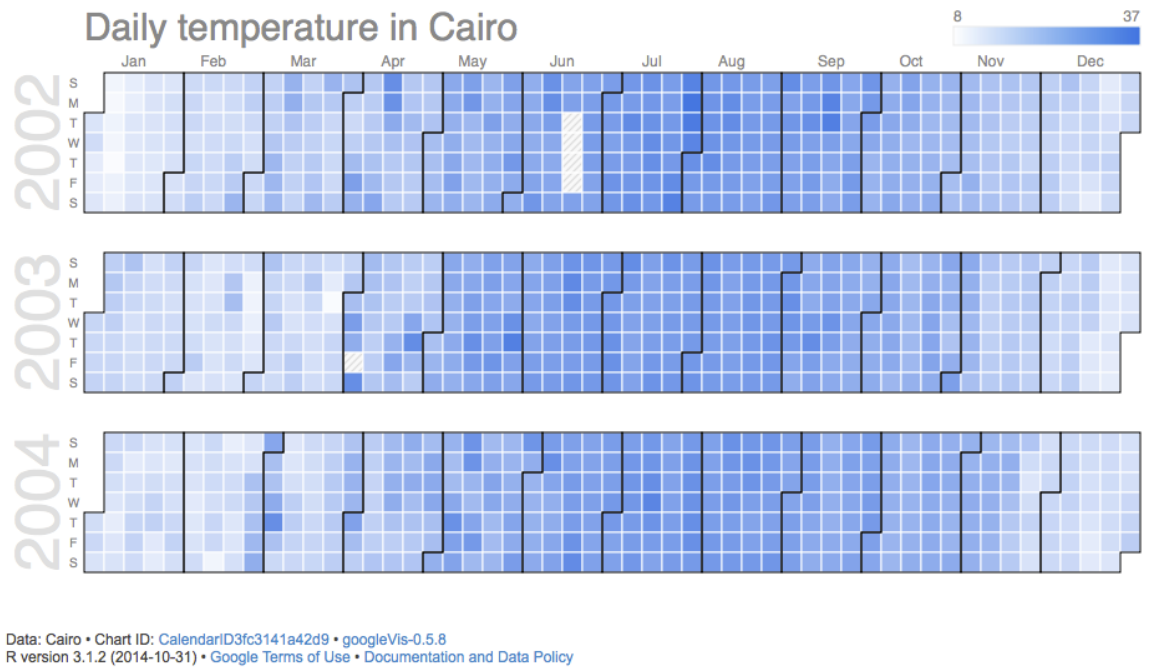
- The dataset *Cairo* (provided in R googleVis package) shows the average temperature (in Celsius) in Cairo from 2002 to 2004. The data frame has 1091 observations with two columns. The first 6 observations are shown below. The minimum and maximum values over the three year period are 8.67 and 36.56.
- The *gvisCalendar* function uses the specified data frame and creates the corresponding calendar chart. The data frame should have at least two columns. The *datevar* argument is the column name of the date dimension. The *numvar* argument is the column name of the numeric data that is displayed against the date dimension.

```
> head(Cairo)
           Date     Temp
2558 2002-01-01 13.61111
2559 2002-01-02 15.16667
2560 2002-01-03 12.00000
2561 2002-01-04 12.00000
2562 2002-01-05 13.05556
2563 2002-01-06 10.05556
> range(Cairo$Temp)
[1]  8.666667 36.555556
```

```
> chart34 <-
+   gvisCalendar(Cairo,
+                datevar="Date",
+                numvar="Temp",
+                options=list(
+                  title="Daily temperature in Cairo",
+                  width=950))
>
> plot(chart34)
```

21. Calendar Chart

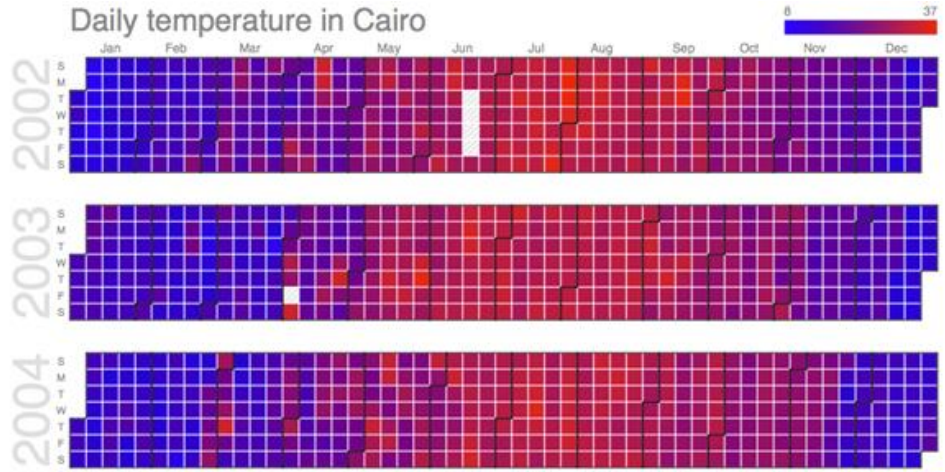
- The corresponding calendar chart is shown in the web browser. The chart displays the tooltips when hovering over the calendar cells.



21. Calendar Chart

- The range of the color axis can be configured as shown blue, the gradient spreading from blue to red.
- The corresponding calendar chart with the specified color range is shown in the web browser. The chart displays the tooltips when hovering over the calendar cells.

```
> chart34_1 <-
+   gvisCalendar(Cairo,
+                datevar="Date",
+                numvar="Temp",
+                options=list(
+                  title="Daily temperature in Cairo",
+                  width=950,
+                  colorAxis="{colors:['blue', 'red']}"))
>
> plot(chart34_1)
```



Data: Cairo • Chart ID: CalendarID3fc31323cfb8 • googleVis-0.5.8
R version 3.1.2 (2014-10-31) • Google Terms of Use • Documentation and Data Policy

22. Timeline Chart

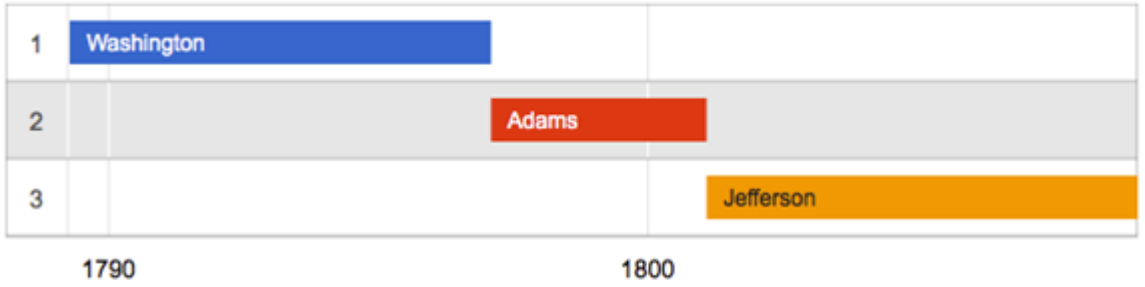
- The timeline chart shows how a set of resources is spread over the time period. Gantt chart is an example of timeline chart. The following data frame shows the starting and ending dates for the three presidents of United States.
- The *gvisTimeline* function uses the specified data frame and creates the corresponding line chart. For the data to be visualized, the *rowlabel* argument is the column name in the data frame for the row labels. The *start* argument is the column name specifying the start dates. The *end* argument is the column name specifying the end dates.

```
> data1 <- data.frame(
+   Term=c("1","2","3"),
+   President=c("Washington", "Adams", "Jefferson"),
+   Begin=as.Date(c("1789-03-29", "1797-02-03", "1801-02-03")),
+   End=  as.Date(c("1797-02-03", "1801-02-03", "1809-02-03")))
>
> data1
  Term  President      Begin        End
1    1 Washington 1789-03-29 1797-02-03
2    2      Adams 1797-02-03 1801-02-03
3    3  Jefferson 1801-02-03 1809-02-03
```

```
> chart35 <-
+   gvisTimeline(data=data1,
+                rowlabel="President",
+                start="Begin", end="End")
>
> plot(chart35)
>
> cat(chart35$html$chart, file = "chart35.html")
```

22. Timeline Chart

- The corresponding timeline chart is shown in the web browser. The chart displays the tooltips when hovering over the bars.

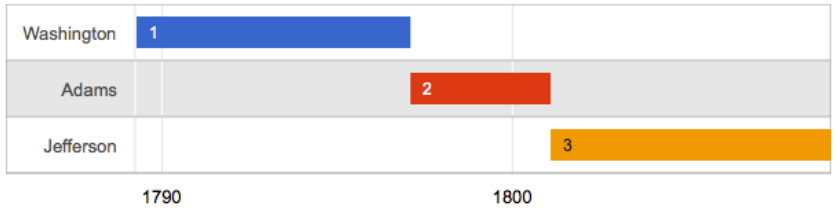


Data: data1 • Chart ID: TimelineID33a8602a2a36 • googleVis-0.5.8
R version 3.1.2 (2014-10-31) • Google Terms of Use • Documentation and Data Policy

22. Timeline Chart

- The data frame has the *Term* column that is also displayed in the above chart. The data frame without the *Term* column can be used as shown below using that information for the bar labels.
- The corresponding timeline chart is shown in the web browser. The chart displays the tooltips when hovering over the bars.

```
> chart36 <-
+   gvisTimeline(data=data1,
+                rowlabel="President",
+                barlabel = "Term",
+                start="Begin", end="End")
>
> plot(chart36)
```



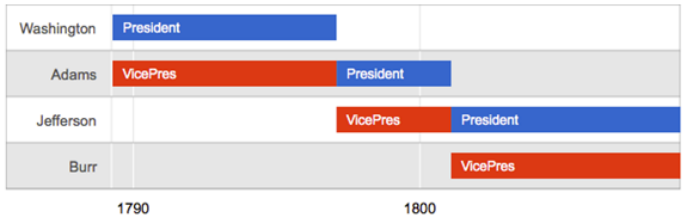
Data: data1 • Chart ID: TimelineID3fc3677c00c4 • googleVis-0.5.8
R version 3.1.2 (2014-10-31) • Google Terms of Use • Documentation and Data Policy

22. Timeline Chart

- The following data frame shows the information about the first three presidents and the vice-presidents under them.
- The timeline chart is created as shown below. It is easy to visualize in the chart the period when *Adams* was vice-president and then becomes the president.

```
> data2 <- data.frame(
+     Position=c(rep("President", 3), rep("VicePres", 3)),
+     Name=c("Washington", "Adams", "Jefferson",
+            "Adams", "Jefferson", "Burr"),
+     Begin=as.Date(rep(c("1789-03-29", "1797-02-03",
+                         "1801-02-03"),2)),
+     End=as.Date(rep(c("1797-02-03", "1801-02-03",
+                       "1809-02-03"),2)))
>
> data2
   Position       Name      Begin        End
1 President Washington 1789-03-29 1797-02-03
2 President      Adams 1797-02-03 1801-02-03
3 President   Jefferson 1801-02-03 1809-02-03
4  VicePres      Adams 1789-03-29 1797-02-03
5  VicePres   Jefferson 1797-02-03 1801-02-03
6  VicePres       Burr 1801-02-03 1809-02-03
```

```
> chart37 <-
+   gvisTimeline(data=data2,
+                rowlabel="Name",
+                barlabel="Position",
+                start="Begin", end="End",
+                options=list(
+                  height=350))
>
> plot(chart37)
```

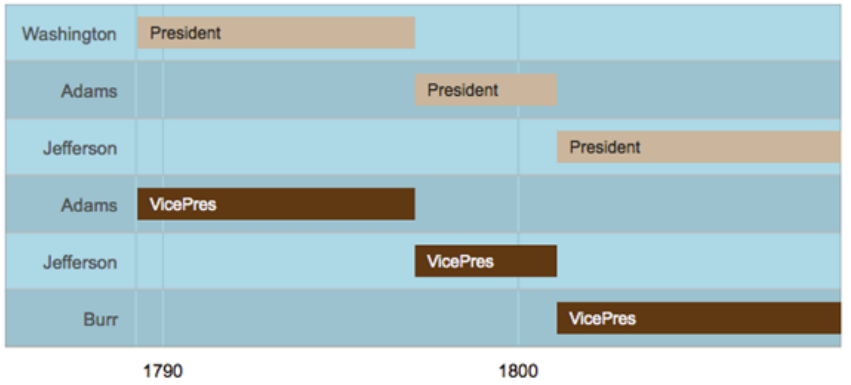


Data: data2 • Chart ID: TimelineID33a84b0c30e3 • googleVis-0.5.8
R version 3.1.2 (2014-10-31) • Google Terms of Use • Documentation and Data Policy

22. Timeline Chart

- By default, the rows are grouped by the *rowlabel* argument of the timeline function. In the following chart, the *groupByRowLabel* option is set to *false*.

```
> chart38 <-
+   gvisTimeline(data=data2,
+                rowlabel="Name",
+                barlabel="Position",
+                start="Begin",
+                end="End",
+                options=list(
+                  timeline="{groupByRowLabel:false}",
+                  height=350,
+                  backgroundColor='#add8e6',
+                  colors="['#cbb69d', '#603913']"))
>
> plot(chart38)
```

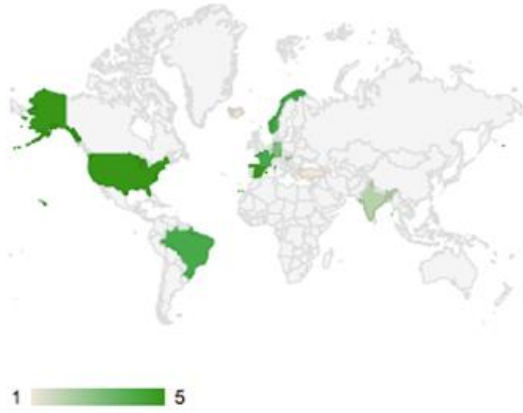


Data: data2 • Chart ID: TimelineID33a8d5dad93 • googleVis-0.5.8
R version 3.1.2 (2014-10-31) • Google Terms of Use • Documentation and Data Policy

23. Merging

- The *gvisMerge* function merges two chart objects into one chart object. The objects are arranged into a HTML table. By default, the *horizontal* option is FALSE.
-
- The geo chart and the table chart used in the previous examples are merged into object as shown below.
- The corresponding merged chart is shown in the web browser.

```
> exports.chart1 <-
+   gvisGeoChart(Exports,
+                locationvar="Country",
+                colorvar="Profit",
+                options=list(width=300, height=300))
>
> exports.chart2 <-
+   gvisTable(Exports,
+             options=list(width=200, height=300))
>
> chart39 <- gvisMerge(exports.chart1,
+                      exports.chart2,
+                      horizontal=TRUE)
> plot(chart39)
```



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

Data: various • Chart ID: MergedID33a833555c4 • googleVis-0.5.8
R version 3.1.2 (2014-10-31) • Google Terms of Use • Data Policy: See individual charts

23. Merging

- The following example merges 6 different charts of the *OpenClose* dataset.

```
> options.chart <- list(legend='none', width=320, height=130)
>
> chart40.1 <-
+   gvisTable(OpenClose,
+             options=options.chart)
>
> chart40.2 <-
+   gvisCandlestickChart(OpenClose, xvar="Weekday",
+                        low="Low", high="High",
+                        open="Open", close="Close",
+                        options=options.chart)
```

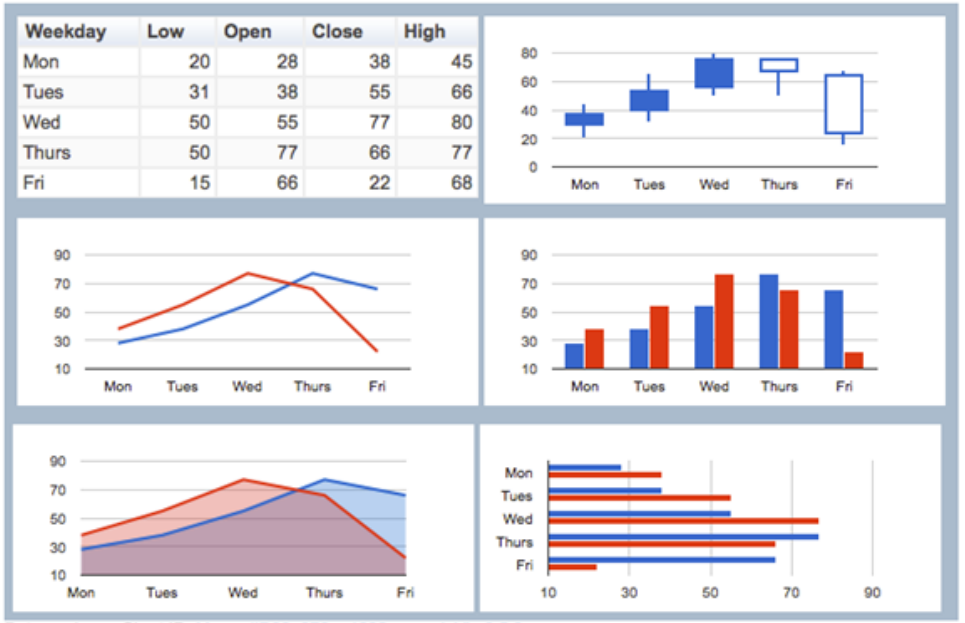
```
> chart40.3 <-
+   gvisLineChart(OpenClose, "Weekday",
+                 c("Open", "Close"),
+                 options=options.chart)
>
> chart40.4 <-
+   gvisColumnChart(OpenClose, "Weekday",
+                   c("Open", "Close"),
+                   options=options.chart)
```

```
> chart40.5 <-
+   gvisAreaChart(OpenClose, "Weekday",
+                 c("Open", "Close"),
+                 options=options.chart)
>
> chart40.6 <-
+   gvisBarChart(OpenClose, "Weekday",
+                c("Open", "Close"),
+                options=options.chart)
```

23. Merging

- Since the *gvisMerge* function only merges two objects at a time, the merging is nested. The above 6 individual chart objects are now merged as shown below.
- The corresponding merged chart is shown in the web browser.

```
> merged.1 <-
+   gvisMerge(
+     gvisMerge(chart40.1, chart40.2, horizontal=TRUE),
+     gvisMerge(chart40.3, chart40.4, horizontal=TRUE),
+     horizontal=FALSE, tableOptions="bgcolor=\"#AABBCC\"")
>
> chart40 <-
+   gvisMerge(
+     merged.1,
+     gvisMerge(chart40.5, chart40.6, horizontal=TRUE),
+     horizontal=FALSE, tableOptions="bgcolor=\"#AABBCC\"")
>
> plot(chart40)
```



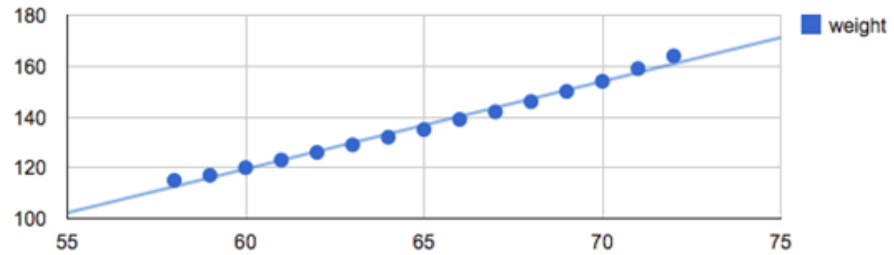
24. Trend Lines

- A trend line is a line superimposed on a chart to show the overall path of the data. Trend lines can automatically be added to scatter charts, bar charts, column charts, and line charts.

- The following example adds the trend line to the scatter chart. The *trendlines* option adds a linear trend line by default to the specified sequence.

- The corresponding scatter chart with the trend line is shown in the web browser. The chart displays the trend line equation along with the data points when hovering over the line.

```
> chart41 <-
+   gvisScatterChart(women,
+                    options=list(
+                      trendlines="0"))
> plot(chart41)
```

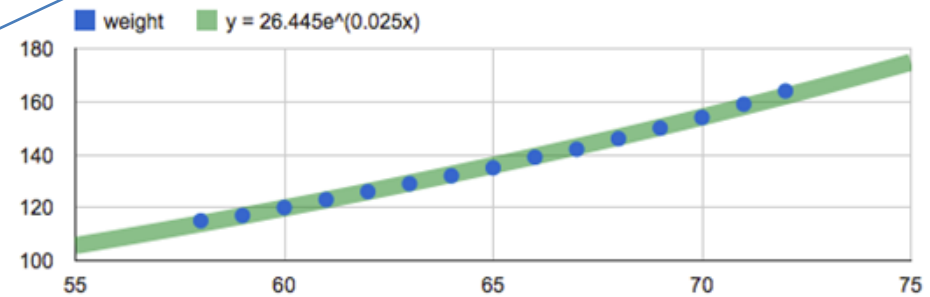


Data: women • Chart ID: ScatterChartID263b706528a9 • googleVis-0.5.5
R version 3.1.2 (2014-10-31) • Google Terms of Use • Documentation and Data Policy

24. Trend Lines

- The *trendlines* option can be customized. In the following example, the scatterplot data is fitted with an exponential trend line. The legend on the top shows the equation of the line.
- The corresponding scatter chart with the exponential trend line is shown in the web browser. The chart displays the trend line equation along with the data points when hovering over the line.

```
> chart42 <-
+   gvisScatterChart(women,
+     options=list(trendlines=
+       "{0: { type: 'exponential',
+              visibleInLegend: 'true',
+              color: 'green',
+              lineWidth: 10,
+             opacity: 0.5}}",
+       legend="top"))
>
>
> plot(chart42)
```

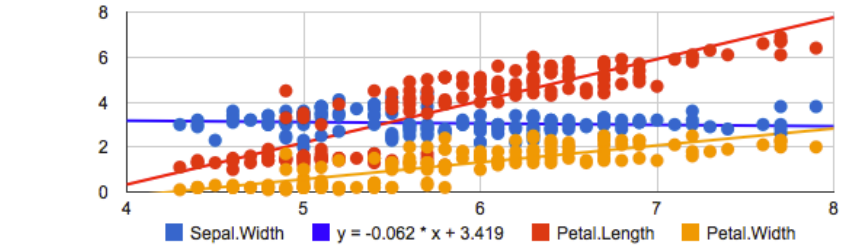


Data: women • Chart ID: ScatterChartID33a854112f6d • googleVis-0.5.8
R version 3.1.2 (2014-10-31) • Google Terms of Use • Documentation and Data Policy

24. Trend Lines

- When multiple series are shown on the scatter plot, a trend line can be customized for each sequence as shown below.
- The corresponding scatter chart with the three linear trend lines is shown in the web browser. The chart displays the trend line equation along with the data points when hovering over the line.

```
> chart43 <-
+   gvisScatterChart(iris[,1:4],
+     options=list(trendlines=
+       "{0: { visibleInLegend: 'true',
+              color: 'blue'},
+         1: { visibleInLegend: 'false',
+              color: 'red'},
+         2: { visibleInLegend: 'false',
+              color: 'orange'}
+       }",
+       legend="bottom"))
>
> plot(chart43)
```

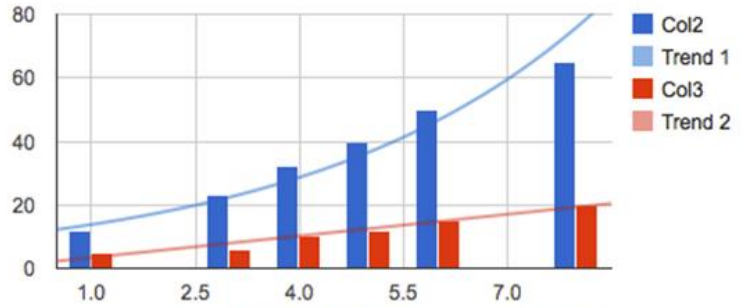


24. Trend Lines

- The following sample data frame has 3 columns of data.
- A column chart of the data frame shows the two series of data (Col2 and Col3) plotted against the first column data. In the following plot, the first sequence is fitted with an exponential trend line, while the second sequence is fitted with a linear trend line.

```
> data3 <- data.frame(
+   Col1=c(1,3,4,5,6,8),
+   Col2=c(12,23,32,40,50,65),
+   Col3=c(5,6,10,12,15,20))
>
> data3
  Col1 Col2 Col3
1    1   12    5
2    3   23    6
3    4   32   10
4    5   40   12
5    6   50   15
6    8   65   20
```

```
> chart44 <-
+   gvisColumnChart(data3,
+         options=list(trendlines="{
+           0: {
+                type: 'exponential',
+                labelInLegend: 'Trend 1',
+                visibleInLegend: true,},
+           1:{
+                labelInLegend: 'Trend 2',
+                visibleInLegend: true}
+           }",
+           chartArea="{left:50,top:20,
+                  width:'50%',height:'75%'}"
+             ))

t(chart44)
```



Data: data • Chart ID: ColumnChartID33a891930fc • googleVis-0.5.8
R version 3.1.2 (2014-10-31) • Google Terms of Use • Documentation and Data Policy

References

- 1. Using Google Charts
- https://developers.google.com/chart/interactive/docs/
-
- 2. Package ‘googleVis’: R Interface for Google Charts
- http://cran.r-project.org/web/packages/googleVis/googleVis.pdf
-