ggvis Exercises (Part-1): Solutions

Below are the solutions to these exercises on visualizations with ggvis.

```
####################
#                  #
#    Exercise 1    #
#                  #
####################

library(ggvis)
library(MASS)
library(magrittr)
attach(Cars93)
plot1 <- ggvis(Cars93, x = ~Horsepower, y = ~MPG.city)

####################
#                  #
#    Exercise 2    #
#                  #
####################

library(ggvis)
library(MASS)
library(magrittr)
attach(Cars93)
plot1 <- ggvis(Cars93, x = ~Horsepower, y = ~MPG.city)
layer_points(plot1)

####################
#                  #
#    Exercise 3    #
#                  #
####################

library(ggvis)
library(MASS)
```

```
library(magrittr)
attach(Cars93)
Cars93 %>%
  ggvis(x = ~Horsepower, y = ~MPG.city) %>%
  layer_points()

####################
#                  #
#    Exercise 4    #
#                  #
####################

library(ggvis)
library(MASS)
library(magrittr)
attach(Cars93)
Cars93 %>% ggvis(~Horsepower, ~MPG.city, stroke = ~Cylinders)
%>% layer_points()

####################
#                  #
#    Exercise 5    #
#                  #
####################

library(ggvis)
library(MASS)
library(magrittr)
attach(Cars93)
Cars93 %>% ggvis(~Horsepower, ~MPG.city, fill = ~Cylinders)
%>% layer_points()

####################
#                  #
#    Exercise 6    #
#                  #
####################

library(ggvis)
library(MASS)
library(magrittr)
```

```
attach(Cars93)
Cars93 %>% ggvis(~Horsepower, ~MPG.city, size = ~EngineSize)
%>% layer_points()

####################
#                  #
#    Exercise 7    #
#                  #
####################

library(ggvis)
library(MASS)
library(magrittr)
attach(Cars93)
Cars93 %>% ggvis(~Horsepower, ~MPG.city, shape =
~factor(Cylinders)) %>% layer_points()

####################
#                  #
#    Exercise 8    #
#                  #
####################

library(ggvis)
library(MASS)
library(magrittr)
attach(Cars93)
Cars93 %>% ggvis(~Horsepower, ~MPG.city, fill := "red", stroke
:= "black") %>% layer_points()

####################
#                  #
#    Exercise 9    #
#                  #
####################

library(ggvis)
library(MASS)
library(magrittr)
attach(Cars93)
Cars93 %>% ggvis(~Horsepower, ~MPG.city, size := 300, opacity
```

```
:= 0.5) %>% layer_points()

####################
#                  #
#    Exercise 10   #
#                  #
####################

library(ggvis)
library(MASS)
library(magrittr)
attach(Cars93)
Cars93 %>% ggvis(~Horsepower, ~MPG.city, shape := "cross") %>%
layer_points()
```

---

ggvis Exercises (Part-1)

INTRODUCTION

The ggvis package is used to make interactive data visualizations. The fact that it combines shiny's reactive programming model and dplyr's grammar of data transformation make it a useful tool for data scientists.

This package may allows us to implement features like interactivity, but on the other hand every interactive ggvis plot must be connected to a running R session.

Before proceeding, please follow our short tutorial.

Look at the examples given and try to understand the logic

behind them. Then try to solve the exercises below using R and without looking at the answers. Then check the solutions.
to check your answers.

Exercise 1

Create a list which will include the variables “Horsepower” and “MPG.city” of the “Cars93” data set. **HINT**: Use ggvis().

Exercise 2

Use the list you just created to make a scatterplot. **HINT**: Use layer_points().

Exercise 3

Use %>% to create the scatterplot of Exercise 2.

Learn more about using ggvis in the online course R: Complete Data Visualization Solutions. In this course you will learn how to:

- Work extensively with the ggvis package and its functionality
- Learn what visualizations exist for your specific use case
- And much more

Exercise 4

Use the list you created in Exercise 1 to create a scatterplot and use “Cylinders” as stroke.

Exercise 5

Use the list you created in Exercise 1 to create a scatterplot and use “Cylinders” as fill.

Exercise 6

Use the list you created in Exercise 1 to create a scatterplot and use “EngineSize” as size.

Exercise 7

Use the list you created in Exercise 1 to create a scatterplot and use “Cylinders” as shape.

Exercise 8

Use the list you created in Exercise 1 to create a scatterplot with red color and black stroke.

Exercise 9

Use the list you created in Exercise 1 to create a scatterplot with size set to 300 and opacity to 0.5 .

Exercise 10

Use the list you created in Exercise 1 to create a scatterplot with cross as shape.

How to create interactive data visualizations with ggvis

INTRODUCTION

The ggvis package is used to make interactive data visualizations. The fact that it combines shiny's reactive programming model and dplyr's grammar of data transformation make it a useful tool for data scientists.

This package may allows us to implement features like interactivity, but on the other hand every interactive ggvis plot must be connected to a running R session.

PACKAGE INSTALLATION & DATA FRAME

The first thing you have to do is install and load the ggvis package with:

```
install.packages("ggvis")
library(ggvis)
```

Moreover we need a data set to work with. Tha dataset we chose in our case is "Cars93" which contains data from 93 Cars on Sale in the USA in 1993 and we can find it in the MASS package which of course must be installed and called too. To install and call those packages and attach the "Cars93" dataset use:

```
install.packages("MASS")
library(MASS)
data("Cars93")
attach(Cars93)
```

You can use head(Cars93) in order to see the variables of your dataset.

Furthermore you need to install and call the shiny and the magrittr package with:

```
install.packages("shiny")
library(shiny)
install.packages("magrittr")
library(magrittr)
```

NOTE: Because of the fact that alla ggvis graphics are web graphics,if you're not using RStudio (which provides a built-in browser), you'll notice that this plot opens in your web

browser.

The ggvis() function

The first thing we have to do is call ggvis(). The first argument is the data set that we want to plot, and the second describes which variables we will use. Look at the example below.

```
plot1 <- ggvis(Cars93, x = ~Length, y = ~Wheelbase)
```

This doesn’t plot anything because you haven’t set how to display your data. The example below creates a scatterplot:

```
layer_points(plot1)
```

There is an alternative and maybe more practical way to produce the same result with the %>% function of the magrittr package like the example below:

```
Cars93 %>%
ggvis(x = ~Length, y = ~Wheelbase) %>%
layer_points()
```

We use ~ before the variable name to indicate that we don’t want to literally use the value of the variable, but instead we want we want to use the variable inside in the dataset. We will use it from now on. This is how we can drop x and y.

You can add more variables to the plot by mapping them to other visual properties like fill, stroke, size and shape. Look at the examples below.

```
Cars93 %>% ggvis(~Length, ~Wheelbase, stroke = ~EngineSize) %>% layer_points() #stroke
Cars93 %>% ggvis(~Length, ~Wheelbase, fill = ~EngineSize) %>% layer_points() #fill
Cars93 %>% ggvis(~Length, ~Wheelbase, size = ~EngineSize) %>% layer_points() #size
Cars93 %>% ggvis(~Length, ~Wheelbase, shape = ~factor(Passengers)) %>% layer_points()#shape
```

If you want to make the points a fixed colour, size or shape,

you need to use := instead of =. Look at the examples below.

```
Cars93 %>% ggvis(~Length, ~Wheelbase, fill := "yellow", stroke := "black") %>% layer_points() #fill
Cars93 %>% ggvis(~Length, ~Wheelbase, size := 350, opacity := 0.5) %>% layer_points() #size
Cars93 %>% ggvis(~Length, ~Wheelbase, shape := "cross") %>% layer_points() #shape
```

Learn more about using ggvis in the online course R: Complete Data Visualization Solutions. In this course you will learn how to:

- Work extensively with the ggvis package and its functionality
- Learn what visualizations exist for your specific use case
- And much more

Interaction

You can map visual properties to variables or set them to specific values, but it is far more interesting to connect them to interactive controls. Look at the two sliders' example below

```
Cars93 %>%
ggvis(~Length, ~Wheelbase,
size := input_slider(1, 100),
opacity := input_slider(0, 1)
) %>%
layer_points()
```

You can also add interactivity to other plot parameters like the width and centers of histogram bins like the example below:

```
Cars93%>%
ggvis(~Length) %>%
layer_histograms(width = input_slider(0, 2, step = 0.10, label
```

```
= "width"),
center = input_slider(0, 2, step = 0.05, label = "center"))
```

Except of input_slider(), ggvis also provides input_checkbox(), input_checkboxgroup(), input_numeric(), input_radiobuttons(), input_select() and input_text().

You can also use keyboard controls with left_right() and up_down(). The following example shows how to control the size of the points by pressing the left and right keyboard controls.

```
arrow % ggvis(~Length, ~Wheelbase, size := arrow, opacity := 0.5) %>% layer_points()
```

With tooltips you can add more complex interactivity:

```
Cars93 %>% ggvis(~Length, ~Wheelbase) %>%
layer_points() %>%
add_tooltip(function(df) df$Length)
```

Data visualization with googleVis solutions part 10

Below are the solutions to these exercises on visualizations with googleVis.

```
####################
#                  #
#    Exercise 1    #
#                  #
####################

library(googleVis)
TLC <- gvisTimeline(data=datTLc)
```

```
####################
#                  #
#    Exercise 2    #
#                  #
####################

library(googleVis)
TLC <- gvisTimeline(data=datTLc)
plot(TLC)

####################
#                  #
#    Exercise 3    #
#                  #
####################

library(googleVis)
TLC <- gvisTimeline(data=datTLc,
                    rowlabel="Name",
                    barlabel="Position",
                    start="start",
                    end="end")
plot(TLC)

####################
#                  #
#    Exercise 4    #
#                  #
####################

library(googleVis)
TLC <- gvisTimeline(data=datTLc,
                    rowlabel="Name",
                    barlabel="Position",
                    start="start",
                    end="end",
options=list(timeline="{groupByRowLabel:false}"))

plot(TLC)

####################
```

```
#                   #
#    Exercise 5     #
#                   #
#####################

library(googleVis)
TLC <- gvisTimeline(data=datTLc,
                    rowlabel="Name",
                    barlabel="Position",
                    start="start",
                    end="end",
options=list(timeline="{groupByRowLabel:false}",
                                  backgroundColor='white',
                                  height=400,
                                  colors="['red', 'green']"))
plot(TLC)

####################
#                   #
#    Exercise 6     #
#                   #
####################

library(googleVis)
Geo <- gvisGeoChart(Exports, "Country", "Profit",
                    options=list(width=300, height=300))
Table <- gvisTable(Exports,
                options=list(width=220, height=300))

GeoTable <- gvisMerge(Geo,Table, horizontal=TRUE)
plot(GeoTable)

####################
#                   #
#    Exercise 7     #
#                   #
####################

library(googleVis)
MotionC=gvisMotionChart(Fruits,
                        idvar = "Fruit",
```

```
                      timevar = "Year"
                      )
####################
#                  #
#    Exercise 8    #
#                  #
####################

library(googleVis)
MotionC=gvisMotionChart(Fruits,
                      idvar = "Fruit",
                      timevar = "Year"
                      )
plot(MotionC)

####################
#                  #
#    Exercise 9    #
#                  #
####################

library(googleVis)
MotionC=gvisMotionChart(Fruits,
                      idvar = "Fruit",
                      timevar = "Year",
                      xvar = "Expenses",
                      yvar = "Sales",
                      sizevar ="Profit",
                      colorvar = "Location")

####################
#                  #
#    Exercise 10   #
#                  #
####################

library(googleVis)
MotionC=gvisMotionChart(Fruits,
                      idvar = "Fruit",
                      timevar = "Year",
                      xvar = "Expenses",
```

```
                        yvar = "Sales",
                        sizevar ="Profit",
                        colorvar = "Location")
plot(MotionC)
```

Data visualization with googleVis exercises part 10

Timeline, Merging & Flash charts

This is part 10 of our series and we are going to explore the features of some interesting types of charts that googleVis provides like Timeline, Flash and learn how to merge two googleVis charts to one.

Read the examples below to understand the logic of what we are going to do and then test yous skills with the exercise set we prepared for you. Lets begin!

Answers to the exercises are available here.

Package & Data frame

As you already know, the first thing you have to do is install and load the googleVis package with:
install.packages("googleVis")
library(googleVis)

Secondly we will create an experimental data frame which will

be used for our charts' plotting. You can create it with:

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

You can explore the "datTLC" data frame with head().

NOTE: The charts are created locally by your browser. In case they are not displayed at once press F5 to reload the page. All charts require an Internet connection.

Timeline Chart

It is quite simple to create a timeline chart with googleVis. We will use the "datTLC" data frame we just created.
Look at the example below to create a simple timeline chart:

```
TLC <- gvisTimeline(data=datTLc)
plot(TLC)
```

Exercise 1

Create a list named "TLC" and pass to it the "datTLC" data frame as a timeline chart. **HINT**: Use gvisTimeline().

Exercise 2

Plot the the timeline chart. **HINT**: Use plot().

You can select the variables you want as rows and columns with:

```
TLC <- gvisTimeline(data=dataframe,
rowlabel="var1",
barlabel="var2",
start="var3",
end="var4")
```

```
plot(TLC)
```

Exercise 3

Put “Name” as rowlabel, “Position” as barlabel, “start” as start “end” as end and plot the chart.

Options

You can group your chart by row or not with:

```
options=list(timeline="{groupByRowLabel:true}")
```

Learn more about using GoogleVis in the online course Mastering in Visualization with R programming. In this course you will learn how to:

- Work extensively with the GoogleVis package and its functionality
- Learn what visualizations exist for your specific use case
- And much more

Exercise 4

Group your timeline chart NOT by rowlabel and plot it.

You can set the colours and size of your chart with:

```
options=list(timeline="{groupByRowLabel:false}",
backgroundColor='yellow',
height=300,
colors="['blue', 'brown']"))
plot(TLC)
```

Exercise 5

Set the background color of your chart to white, the “Position” colours to red and green respectively, the height to 400 and plot it.

Merging charts

We will now see how to merge two charts to one. For this purpose we are going to use a Geo Chart and a Table which we saw in parts 6 & 7 respectively.

```
Geo <- gvisGeoChart(Exports, "Country", "Profit",
options=list(width=400, height=400))
Table <- gvisTable(Exports,
options=list(width=320, height=400))
```

After you create these two charts you can merge them with:

```
GeoTable <- gvisMerge(Geo,Table, horizontal=TRUE)
plot(GeoTable)
```

Exercise 6

Create a Geo chart and Table like the example above and merge them. **HINT**: Use gvisMerge().

Flash charts

All the following charts require a Flash player.

Motion chart

The most exciting type of chart that googleVis provides, in my opinion, is the motion chart. It is quite simple to create a motion chart with googleVis. We will use the “Fruits” data set for this example. You can see the variables of your data set with head().
Look at the example below to create a simple motion chart:

```
MotionC=gvisMotionChart(Fruits,
idvar = "Fruit",
timevar = "Year"
)
plot(MotionC)
```

Exercise 7

Create a list named “MotionC” and pass to it the “Fruits” data set as a motion chart. **HINT**: Use gvisMotionChart().

Exercise 8

Plot the the motion chart. **HINT**: Use plot().

As you saw the variables were set automatically, but you can set them as you want with:

```
MotionC=gvisMotionChart(Fruits,
idvar = "Fruit",
timevar = "Year",
xvar = "Expenses",
yvar = "Sales",
sizevar ="Profit",
colorvar = "Location")
plot(MotionC)
```

Exercise 9

Create a list named “MotionC” and pass to it the “Fruits” data set as a motion chart. You can use the example above or you can use the variables differently to see the differences. **HINT**: Use gvisMotionChart().

Exercise 10

Plot the the motion chart. **HINT**: Use plot().

R Markdown exercises part 2: solutions

Below are the solutions to these RMarkdown exercises.

````
####################
#                  #
#    Exercise 1    #
#                  #
####################

---
title: "R Markdown Example"
author: "Makis"
date: 18/Jul/2017
output: html_document
---

```{r,echo=FALSE}
summary(cars)
```

```{r,fig.width=5, fig.height=5, echo=FALSE,dev='svg'}
plot(cars)
```

```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
dataframe
```

```{r,echo=FALSE,warning=FALSE}
library(knitr)
kable(dataframe, digits = 1)
```

####################
#                  #
#    Exercise 2    #
#                  #
####################

---
title: "**R Markdown Example**"
````

````
author: "Makis"
date: 18/Jul/2017
output: html_document
---

```{r,echo=FALSE}
summary(cars)
```

```{r,fig.width=5, fig.height=5, echo=FALSE,dev='svg'}
plot(cars)
```

```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
dataframe
```

```{r,echo=FALSE,warning=FALSE}
library(knitr)
kable(dataframe, digits = 1)
```

####################
#                  #
#    Exercise 3    #
#                  #
####################

---
title: "**R Markdown Example**"
author: "*Makis*"
date: 18/Jul/2017
output: html_document
---


```{r,echo=FALSE}
summary(cars)
````
````

```

```{r,fig.width=5, fig.height=5, echo=FALSE,dev='svg'}
plot(cars)
```

```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
dataframe
```

```{r,echo=FALSE,warning=FALSE}
library(knitr)
kable(dataframe, digits = 1)
```

```
####################
#                  #
#    Exercise 4    #
#                  #
####################

---
title: "**R Markdown Example**"
author: "*Makis*"
date: 18/Jul/2017
output: html_document
---

# Summary
```{r,echo=FALSE}
summary(cars)
```

```{r,fig.width=5, fig.height=5, echo=FALSE,dev='svg'}
plot(cars)
```

```{r, echo=FALSE}
```

```
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
dataframe
```

```{r,echo=FALSE,warning=FALSE}
library(knitr)
kable(dataframe, digits = 1)
```
```
####################
#                  #
#    Exercise 5    #
#                  #
####################
```

```
---
title: "**R Markdown Example**"
author: "*Makis*"
date: 18/Jul/2017
output: html_document
---

# Summary
```{r,echo=FALSE}
summary(cars)
```
Plot
```{r,fig.width=5, fig.height=5, echo=FALSE,dev='svg'}
plot(cars)
```


```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
```
Dataframe
```{r, echo=FALSE}
dataframe
```
```

````
### Table 1
```{r,echo=FALSE,warning=FALSE}
library(knitr)
kable(dataframe, digits = 1)
```

####################
#                  #
#    Exercise 6    #
#                  #
####################

---
title: "**R Markdown Example**"
author: "*Makis*"
date: 18/Jul/2017
output: html_document
---

# Summary
```{r,echo=FALSE}
summary(cars)
```
### Plot
```{r,fig.width=5, fig.height=5, echo=FALSE,dev='svg'}
plot(cars)
```

```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
```
### Dataframe
```{r, echo=FALSE}
dataframe
```

### Table 1
```{r,echo=FALSE,warning=FALSE}
library(knitr)
````
````

````
kable(dataframe, digits = 1)
```

| A    | B   |
|:-----|:---:|
| Bob  | 1.78|
| Tom  | 1.86|
| Bill | 1.85|
| Joe  | 1.70|


####################
#                  #
#    Exercise 7    #
#                  #
####################

---
title: "**R Markdown Example**"
author: "*Makis*"
date: 18/Jul/2017
output: html_document
---

# Summary
```{r,echo=FALSE}
summary(cars)
```
### Plot
```{r,fig.width=5, fig.height=5, echo=FALSE,dev='svg'}
plot(cars)
```

```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
```
### Dataframe
```{r, echo=FALSE}
dataframe
````

```

### Table 1
```{r,echo=FALSE,warning=FALSE}
library(knitr)
kable(dataframe, digits = 1)
```

A	B
Bob	1.78
Tom	1.86
Bill	1.85
Joe	1.70

####################
#   #
Exercise 8
#   #
####################

---
title: "**R Markdown Example**"
author: "*Makis*"
date: 18/Jul/2017
output: html_document

Summary
```{r,echo=FALSE}
summary(cars)
```
Plot
```{r,fig.width=5, fig.height=5, echo=FALSE,dev='svg'}
plot(cars)
```

```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
```
```

```
### Dataframe
```{r, echo=FALSE}
dataframe
```

Table 1
```{r,echo=FALSE,warning=FALSE}
library(knitr)
kable(dataframe, digits = 1)
```

A	B
Bob	1.78
Tom	1.86
Bill	1.85
Joe	1.70

* Bob
* Tom
* Bill
* Joe

####################
#   #
Exercise 9
#   #
####################

---
title: "**R Markdown Example**"
author: "*Makis*"
date: 18/Jul/2017
output: html_document

Summary
```{r,echo=FALSE}
summary(cars)
```
```

```
### Plot
```{r,fig.width=5, fig.height=5, echo=FALSE,dev='svg'}
plot(cars)
```


```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
```
Dataframe
```{r, echo=FALSE}
dataframe
```


Table 1
```{r,echo=FALSE,warning=FALSE}
library(knitr)
kable(dataframe, digits = 1)
```


A	B
Bob	1.78
Tom	1.86
Bill	1.85
Joe	1.70

1. Bob
2. Tom
3. Bill
4. Joe

####################
#   #
Exercise 10
#   #
####################
```

````
---
title: "**R Markdown Example**"
author: "*Makis*"
date: 18/Jul/2017
output: html_document
---

# Summary
```{r,echo=FALSE}
summary(cars)
```
### Plot
```{r,fig.width=5, fig.height=5, echo=FALSE,dev='svg'}
plot(cars)
```


```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
```
### Dataframe
```{r, echo=FALSE}
dataframe
```


### Table 1
```{r,echo=FALSE,warning=FALSE}
library(knitr)
kable(dataframe, digits = 1)
```


| A    | B   |
|:-----| ---:|
| Bob  | 1.78|
| Tom  | 1.86|
| Bill | 1.85|
| Joe  | 1.70|

1. Bob
2. Tom
````

3. Bill
4. Joe

[Link](https://www.r-exercises.com)

R Markdown exercises part 2

INTRODUCTION

R Markdown is one of the most popular data science tools and is used to save and execute code, create exceptional reports whice are easily shareable.

The documents that R Markdown provides are fully reproducible and support a wide variety of static and dynamic output formats.

Using markdown syntax, which provides an easy way of creating documents that can be converted to many other file types, while embeding R code in the report, so it is not necessary to keep the report and R script separately. Furthermore The report is written as normal text, so knowledge of HTML is not required. Of course no additional files are needed because everything is incorporated in the HTML file.

Before proceeding, please follow our short tutorial.

Look at the examples given and try to understand the logic behind them. Then try to solve the exercises below using R and

without looking at the answers. Then check the solutions.
to check your answers.

Exercise 1

Make a table out of the object dataframe you created and set its numbers to have one significant figure. **HINT**: Use kable().

Exercise 2

Use bold text for your report's title. **HINT**: Use ** **.

Exercise 3

Use Italic text for the author's name. **HINT**: Use * *.

Learn more about reporting your results in the online course: R for Data Science Solutions. In this course you will learn how to:

- Build a complete workflow in R for your data science problem
- Get indepth on how to report your results in a interactive way
- And much more

Exercise 4

Add "Summary" as Header of size 1 above your summary context.

Exercise 5

Add "Plot", "Dataframe" and "Table 1" as Headers of size 3 above the rest of the three objects of your report respectively.

Exercise 6

Create manually a small table for your dataframe.

Exercise 7

Apply right alignment to the column “B”.

Exercise 8

Create an unordered list of the contents of column “A” of your dataframe.

Exercise 9

Transform the list you just created to ordered.

Exercise 10

Add a link named “Link” that leads to “www.r-exercises.com”.

Data visualization with googleVis exercises part 9

Histogram & Calendar chart

This is part 9 of our series and we are going to explore the features of two interesting types of charts that googleVis provides like histogram and calendar charts.

Read the examples below to understand the logic of what we are going to do and then test yous skills with the exercise set we prepared for you. Lets begin!

Answers to the exercises are available here.

Package & Data frame

As you already know, the first thing you have to do is install and load the googleVis package with:
install.packages("googleVis")
library(googleVis)

To run this example we will first create an experimental data frame with:
Hist=data.frame(A=rpois(100, 10),
B=rpois(100, 20),
C=rpois(100, 30))

NOTE: The charts are created locally by your browser. In case they are not displayed at once press F5 to reload the page. All charts require an Internet connection.

Histogram

It is quite simple to create a Histogram with googleVis. We will use the "Hist" data frame we just created. You can see the variables of your data frame with head().
Look at the example below to create a simple histogram:
HistC <- gvisHistogram(Hist)
plot(HistC)

Exercise 1

Create a list named "HistC" and pass to it the "Hist" data frame as a histogram. **HINT**: Use gvisHistogram().

Exercise 2

Plot the the histogram. **HINT**: Use plot().

Options

To add a legend to your chart you can use:
options=list(

```
legend="{ position: 'top' }")
```

Exercise 3

Add a legend to the bottom of your histogram and plot it. **HINT**: Use list().

To decide the colours of your bars you can use:

```
options=list(
colors="['black', 'green', 'yellow']")
```

Learn more about using GoogleVis in the online course Mastering in Visualization with R programming. In this course you will learn how to:

- Work extensively with the GoogleVis package and its functionality
- Learn what visualizations exist for your specific use case
- And much more

Exercise 4

Change the colours of the histogram's bars to red, green and blue and plot it. **HINT**: Use colors.

To set the dimensions of your histogram you can use:

```
options=list(
width=400, height=400)
```

Exercise 5

Set width of your histogram to 500, its height to 400 and plot it.

Calendar chart

It is quite simple to create a Calendar Chart with googleVis. We will use the "Cairo" data set. You can see the variables of

“Cairo” with head().
Look at the example below to create a simple calendar chart:

```
CalC <- gvisCalendar(Cairo)
plot(CalC)
```

Exercise 6

Create a list named “CalC” and pass to it the “Cairo” data set as a calendar chart. **HINT**: Use gvisCalendar().

Exercise 7

Plot the the calendar chart. **HINT**: Use plot().

Options

You can add title to your chart and set the dimensions with:

```
options=list(
title="Title",
height=400)
```

Exercise 8

Add a title to your calendar chart, set height to 500 and plot it. **HINT**: Use list().

You can change the features of your labels with:

```
options=list(calendar="{yearLabel: { fontName: 'Times-Roman',
fontSize: 26, color: 'black', bold: false})
```

Exercise 9

Add labels to your chart ,set the font of your labels to “Times-Roman”, their size to 30, their color to black, make them bold and plot the chart.

To find more options about the cells you can use:

```
cellSize: 15,
cellColor: { stroke: 'red', strokeOpacity: 0.5 },
focusedCellColor: {stroke:'red'}
```

Exercise 10

Set the size of the cells to 10, the focused color to green and plot the chart.

Data visualization with googleVis solutions part 9

Below are the solutions to these exercises on visualizations with googleVis.

```
####################
#                  #
#    Exercise 1    #
#                  #
####################

library(googleVis)
HistC <- gvisHistogram(Hist)

####################
#                  #
#    Exercise 2    #
#                  #
####################

library(googleVis)
HistC <- gvisHistogram(Hist)
plot(HistC)

####################
#                  #
#    Exercise 3    #
#                  #
####################
```

```
library(googleVis)
HistC <- gvisHistogram(Hist,
                       options=list(
                         legend="{ position: 'bottom'}"))
plot(HistC)

####################
#                  #
#    Exercise 4    #
#                  #
####################

library(googleVis)
HistC <- gvisHistogram(Hist,
                       options=list(
                         legend="{ position: 'bottom'}",
                         colors="['red', 'green', 'blue']"))
plot(HistC)

####################
#                  #
#    Exercise 5    #
#                  #
####################

library(googleVis)
HistC <- gvisHistogram(Hist,
                       options=list(
                         legend="{ position: 'bottom'}",
                         colors="['red', 'green', 'blue']",
                         width=500, height=400))
plot(HistC)

####################
#                  #
#    Exercise 6    #
#                  #
####################

library(googleVis)
CalC <- gvisCalendar(Cairo)
```

```
####################
#                  #
#    Exercise 7    #
#                  #
####################

library(googleVis)
CalC <- gvisCalendar(Cairo)
plot(CalC)

####################
#                  #
#    Exercise 8    #
#                  #
####################

library(googleVis)
CalC <- gvisCalendar(Cairo,
                     options=list(
                       title="Daily temperature in Cairo",
                       height=500))
plot(CalC)

####################
#                  #
#    Exercise 9    #
#                  #
####################

library(googleVis)
CalC <- gvisCalendar(Cairo,
                     options=list(
                       title="Daily temperature in Cairo",
                       height=500,
                         calendar="{yearLabel: { fontName:
'Times-Roman',
                                  fontSize: 30, color: 'black',
bold: true}}"))
plot(CalC)

####################
```

```
#                   #
#    Exercise 10    #
#                   #
#####################

library(googleVis)
CalC <- gvisCalendar(Cairo,
                     options=list(
                       title="Daily temperature in Cairo",
                       height=500,
                         calendar="{yearLabel: { fontName:
'Times-Roman',
                                  fontSize: 30, color: 'black',
bold: true},
                       cellSize: 10,
                                  cellColor: { stroke: 'green',
strokeOpacity: 0.2 },
                       focusedCellColor: {stroke:'green'}}"))
plot(CalC)
```

R Markdown exercises part 1: solutions

Below are the solutions to these R Markdown exercises.

```
####################
#                   #
#    Exercise 1     #
#                   #
####################

# 1) File
# 2) New File
# 3) R Markdown
# 4) HTML
```

```
# 5) Delete the default instructions

####################
#                  #
#    Exercise 2    #
#                  #
####################

---
title: "R Markdown Example"
author: "Makis"
date: 18/Jul/2017
output: html_document
---

####################
#                  #
#    Exercise 3    #
#                  #
####################

---
title: "R Markdown Example"
author: "Makis"
date: 18/Jul/2017
output: html_document
---

```{r}
summary(cars)
```
####################
#   #
Exercise 4
#   #
####################

---
title: "R Markdown Example"
author: "Makis"
date: 18/Jul/2017
```

````
output: html_document
---

```{r}
summary(cars)
```

```{r}
plot(cars)
```

####################
#                  #
#    Exercise 5    #
#                  #
####################

---
title: "R Markdown Example"
author: "Makis"
date: 18/Jul/2017
output: html_document
---

```{r}
summary(cars)
```

```{r}
plot(cars)
```

```{r}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
dataframe
```

####################
#                  #
````

````
#    Exercise 6    #
#                  #
####################
---
title: "R Markdown Example"
author: "Makis"
date: 18/Jul/2017
output: html_document
---

```{r,echo=FALSE}
summary(cars)
```

```{r, echo=FALSE}
plot(cars)
```

```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
dataframe
```
####################
#                  #
#    Exercise 7    #
#                  #
####################

---
title: "R Markdown Example"
author: "Makis"
date: 18/Jul/2017
output: html_document
---

```{r,echo=FALSE}
summary(cars)
```
````

```{r, echo=FALSE}
plot(cars)
```

```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
dataframe
```

```{r,echo=FALSE}
library(knitr)
```

```
####################
#                  #
#    Exercise 8    #
#                  #
####################
```

```
---
title: "R Markdown Example"
author: "Makis"
date: 18/Jul/2017
output: html_document
---
```

```{r,echo=FALSE}
summary(cars)
```

```{r, echo=FALSE}
plot(cars)
```

```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
dataframe
```

```

```{r,echo=FALSE,warning=FALSE}
library(knitr)
```

####################
#   #
Exercise 9
#   #
####################

---
title: "R Markdown Example"
author: "Makis"
date: 18/Jul/2017
output: html_document

```{r,echo=FALSE}
summary(cars)
```

```{r,fig.width=5, fig.height=5, echo=FALSE}
plot(cars)
```

```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
dataframe
```

```{r,echo=FALSE,warning=FALSE}
library(knitr)
```

####################
#   #
Exercise 10
```

```
#                    #
####################

---
title: "R Markdown Example"
author: "Makis"
date: 18/Jul/2017
output: html_document
---

```{r,echo=FALSE}
summary(cars)
```

```{r,fig.width=5, fig.height=5, echo=FALSE,dev='svg'}
plot(cars)
```

```{r, echo=FALSE}
A <- c("Bob", "Tom", "Bill", "Joe")
B <- c(1.78, 1.86, 1.85, 1.70)
dataframe <- data.frame(A, B)
dataframe
```

```{r,echo=FALSE,warning=FALSE}
library(knitr)
```
```