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.

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

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.

R Markdown exercises part 1

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

Create a new R Markdown file (.Rmd) in RStudio.

Exercise 2

Insert a YAML Header with title, author and date of your choice at the top of your .Rmd script.

Exercise 3

Display the summary of “cars” dataset in your report. **HINT**: Use summary().

Exercise 4

Make a plot of the “cars” dataset under the summary you just created. **HINT**: Use plot().

Exercise 5

Create a small experimental dataframe and dipslay it in your report. **HINT**: Use data.frame().

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 6

Hide the code from your report. **HINT**: Use echo.

Exercise 7

Load the package “knitr” in your .Rmd file. and hide the code chunk. **HINT**: Use echo.

Exercise 8

Hide the warning message that appeared in your report. **HINT**: Use warning.

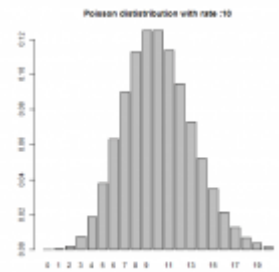
Exercise 9

Set fig.width and fig.height of your plot to 5.

Exercise 10

Change the file format of your plot from .png to .svg. **HINT**: Use dev.

Hacking statistics or: How I Learned to Stop Worrying About Calculus and Love Stats Exercises (Part-4)



Statistics are often taught in school by and for people who like Mathematics. As a consequence, in those class emphasis is put on leaning equations, solving calculus problems and creating mathematics models instead of building an intuition for probabilistic problems. But, if you read this, you know a bit of R programming and have access to a computer that is really good at computing stuff! So let’s learn how we can tackle useful statistic problems by writing simple R query and how to think in probabilistic terms.

Until now, in this series of exercise sets, we have used only continuous probability distributions, which are functions defined on all the real numbers on a certain interval. As a consequence, random variable who have those distributions can assume an infinity of values. However, a lot of random situations only have a finite amount of possible outcome and using a continuous probability distributions to analyze them is not really useful. In today set, we'll introduce the concept of discrete probability functions, which can be used in those situations and some examples of problems in which they can be used.

Answers to the exercises are available here.

Exercise 1
Just as continuous probability distributions are characterized by a probability density function discrete probability functions are characterized by a probability mass function which gives the probability that a random variable is equal to one value.

The first probability mass function we will use today is the binomial distribution, which is used to simulate n iterations of a random process who can either result in a success, with a probability of p, or a failure, with a probability of (1-p). Basically, if you want to simulate something like a coins flip, the binomial distribution is the tool you need.

Suppose you roll a 20 sided dice 200 times and you want to know the probability to get a 20 exactly five times on your rolls. Use the dbinom(n, size, prob) function to compute this probability.

Exercise 2
For the binomial distribution, the individual events are independents, meaning that the probability of realization of two events can be calculated by adding the probability of realization of both event. This principle can be generalize to

any number of events. For example, the probability of getting three tails or less when you flip a coins 10 time is equal to the probability of getting 1 tails plus the probability of getting 2 tails plus the probability of getting 3 tails.

Knowing this, use the dbinom() function to compute the probability of getting six correct responses at a test made of 10 questions which have true or false for answer if you answer randomly. Then, use the pbinom() function to compute the cumulative probability function of the binomial distribution in that situation.

Exercise 3

Another consequence of the independence of events is that if we know the probability of realization of a set of events we can compute the probability of realization of one of his subset by subtracting the probability of the unwanted event. For example, the probability of getting two or three tails when you flip a coins 10 time is equal to the probability of getting at least 3 tails minus the probability of getting 1 tails.

Knowing this, compute the probability of getting 6 or more correct answer on the test described in the previous exercise.

Exercise 4

Let’s say that in an experiment a success is defined as

getting a 1 if you roll a 20 sided die. Use the barplot() function to represent the probability of getting from 0 to 10 success if you roll the die 10 times. What happened to the barplot if you roll a 10 sided die instead? If you roll a 3 sided die?

Exercise 5
Another discrete probability distribution close to the binomial distribution is the Poisson distribution, which give the probability of a number of events to occur during a fixed amount of time if we know the average rate of his occurrence. For example, we could use this distribution to estimate the amount of visitor who goes on a website if we know the average number of visitor per second. In this case, we must assume two things: first that the website has visitor from around the world since the rate of visitor must be constant around the day and two that when a visitor is coming on the site he is not influenced by the last visitor since a process can be expressed by the Poisson distribution if the events are independent from each other.

Use the dpois() function to estimate the probability of having 85 visitors on a website in the next hour if in average 80 individual connect on the site per hour. What is the probability of getting 2000 unique visitors on the website in a day?

Exercise 6
Poisson distribution can be also used to compute the probability of an event occurring in an amount of space, as long as the unit of the average rate is compatible with the unit of measure of the space you use. Suppose that a fishing boat catch 1/2 ton of fish when his net goes through 5 squares kilometers of sea. If the boat combed 20 square kilometer, what is the probability that they catch 5 tons of fish?

Exercise 7
Until now, we used the Poisson distribution to compute the

probability of observing precisely n occurrences of an event. In practice, we are often interested in knowing the probability that an event occur n times or less. To do so we can use the ppois() function to compute the cumulative Poisson distribution. If we are interested in knowing what is the probability of observing strictly more than n occurrences, we can use this function and set the parameter lower to FALSE.

In the situation of exercise 5, what is the probability that the boat caught 5 tons of fish or less? What is the probability that the caught more than 5 tons of fish?

Note that, just as in a binomial experiment, the events in a Poisson process are independant, so you can add or subtract probability of event to compute the probability of a particular set of events.

Exercise 8

Draw the Poisson distribution for average rate of 1,3,5 and 10.

Exercise 9

The last discrete probability distribution we will use today is the negative binomial distribution which give the probability of observing a certain number of success before observing a fixed number of failures. For example, imagine that a professional football player will retire at the end of the season. This player has scored 495 goals in his career and would really want to meet the 500 goal mark before retiring. If he is set to play 8 games until the end of the season and score one goal every three games in average, we can use the negative binomial distribution to compute the probability that he will meet his goal on his last game, supposing that he won’t score more than one goal per game.

Use the dnbinom() function to compute this probability. In this case, the number of success is 5, the probability of success is 1/3 and the number of failures is 3.

Exercise 10

Like for the Poisson distribution, R give us the option to compute the cumulative negative binomial distribution with the function pnbinom(). Again, the lower.tail parameter than give you the option to compute the probability of realizing more than n success if he is set to TRUE.

In the situation of the last exercise, what is the probability that the football player will score at most 5 goals in before the end of his career.

Data Visualization with googleVis exercises part 8

Annotation & Sankey Charts

In the eighth part of our series we are going to learn about the features of some interesting types of charts. More specifically we will talk about Annotation and Sankey 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

As you already know, the first thing you have to do is install

and load the googleVis package with:
install.packages("googleVis")
library(googleVis)

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.

Annotation chart

It is quite simple to create an Annotation Chart with googleVis. We will use the "Stocks" dataset. You can see the variables of your dataset with head().
Look at the example below to create a simple Annotation Chart:
AnnoC <- gvisAnnotationChart(Stock)
plot(AnnoC)

Exercise 1

Create a list named "AnnoC" and pass to it the "Stock" dataset as an annotation chart. **HINT**: Use gvisAnnotationChart().

Exercise 2

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

Set the variables

As you can see the annotation chart you built is empty so we have to fill it with some information. We will use the variables of ths "Stock" dataset for this purpose like this:
datevar="Date",
numvar="Value",
idvar="Device",
titlevar="Title",
annotationvar="Annotation"

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 3

Use the example above to fill your annotation chart with the proper information and plot the chart.

Dimensions

You can use height and width to change the dimensions of the annotation chart.
options=list(width=500, height=300)

Exercise 4

Set height to 700, width to 500 and plot the chart. **HINT**: Use list().

Colours

You can change the colours of the lines with:
options=list(colors="['green','yellow']")

Exercise 5

Set the line colours to green and red and plot the chart.

With the fill option you can color the relevant areas and adjust how filled these areas will be. Look at the example:
options=list(colors="['green','yellow']",fill=20)

Exercise 6

Set fill to 50 and plot the chart.

Exercise 7

Now set fill to 150 to see the difference and plot the chart.

Sankey Chart

Before creating a sankey chart we have to create a data frame that will help us as an example, so copy and paste this code to crate the data frame “exp” first:

```
exp <- data.frame(From=c(rep("A",3), rep("B", 3)),
To=c(rep(c("X", "Y", "Z"),2)),
Weight=c(6,9,7,9,3,1))
```

As you can see we created a data frame with the variabls “From”, “To” and “Weight”. You can use head() in order to see them.
It is quite simple to create an Sankey Chart with googleVis. We will use the “exp” data frame.
Look at the example below to create a simple Sankey Chart:

```
SankeyC <- gvisSankey(exp )
plot(SankeyC)
```

Exercise 8

Create a list named “SankeyC” and pass to it the “exp” dataset as a sankey chart. **HINT**: Use gvisSankey().

Exercise 9

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

You can change the link colours with:

```
options=list(sankey="{link: {color: { fill: 'red' } }
```

Exercise 10

Color the links of ths sankey chart green and plot it.

Data visualization with googleVis exercises part 7

Table, Org Chart & Tree Map

In the seventh part of our series we are going to learn about the features of some interesting types of charts. More specifically we will talk about Table, Org Chart and Tree Map.

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

As you already know, the first thing you have to do is install and load the googleVis package with:

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

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

Table

It is quite simple to create a Table with googleVis. We will use the "Stock" dataset.
Look at the example below to create a simple table:

```
TableC <- gvisTable(Stock)
plot(TableC)
```

Exercise 1

Create a list named “TableC” and pass to it the “Stock” dataset as a table. **HINT**: Use gvisTable().

Exercise 2

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

Table with pages

To add pages to your table use:
```
options=list(page='enable')
```

Exercise 3

Add pages to the table you just created and plot it. **HINT**: Use list().

Org chart

It is quite simple to create an Org Chart with googleVis. We will use the “Regions” dataset. You can see the variables of your dataset with head().
Look at the example below to create a simple Org Chart:
```
OrgC <- gvisOrgChart(Regions )
plot(OrgC)
```

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

Create a list named “OrgC” and pass to it the “Regions” dataset as an org chart. **HINT**: Use gvisOrgChart().

Exercise 5

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

Dimensions

You can adjust the dimensions of the org chart with these options:

```
options=list(width=600, height=250,
size='large')
```

Exercise 6

Adjust the dimensions of your org chart. Set height to 300, width to 550 and size to medium and plot it.

Tree Map

It is quite simple to create a Tree Map with googleVis. We will use the “Regions” dataset.
Look at the example below to create a simple Tree Map:

```
TreeC <- gvisTreeMap(Regions)
plot(TreeC)
```

Exercise 7

Create a list named “TreeC” and pass to it the “Regions” dataset as an org chart. **HINT**: Use gvisTreeMap().

Exercise 8

Plot the the tree map. **HINT**: Use plot().

You can decide tha dependents variables of your dataset by

selecting it. In the example above the dependent variable was “Val”. To choose “Fac” follow the example:

```
TreeC <- gvisTreeMap(Regions,
"Region", "Parent",
"Fac")
plot(TreeC)
```

Exercise 9

Set “Fac” as your dependent variable, plot the tree map and see the difference.

Font size

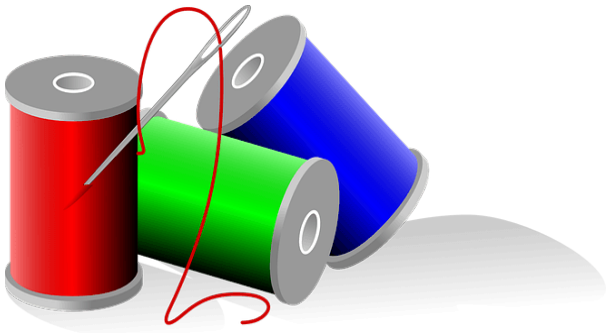
Obviously you can change the font size of your tree map simply with:

```
options=list(fontSize=10)
```

Exercise 10

Set the size of your font to 20 and plot your tree map. **HINT**: Use fontSize.

Hacking strings with stringr



This is first of the set of exercise on string manipulation with stringr

Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Exercise 1
use a stringr function to merge this 3 strings .
x <- "I AM SAM. I AM SAM. SAM I AM"
y <- "THAT SAM-I-AM! THAT SAM-I-AM! I DO NOT LIKE THAT SAM-I-AM!"
z <- ""DO WOULD YOU LIKE GREEN EGGS AND HAM?"

Exercise 2

Now use a vector which contains x,y,z and NA and make it a single sentence using paste ,do the same by the same function you used for exercise1 .Can you spot the difference .

Exercise 3

Install the babynames dataset ,find the vector of length of the babynames using stringr functions. You may wonder nchar can do the same so why not use that ,try finding out the difference and let me know in the comments.

Exercise 4

We often use substr to get part of the string ,in stringr world there exist a much powerful function which does almost the same thing . Create a string name with your name .
Use str_sub to get the last character and the last 5 characters .

Exercise 5

In mtcars dataset rownames, find all cars of the brand Merc .

Analytics/Text Mining Using R. In this course you will learn how create, analyse and finally visualize your text based data source. Having all the steps easily outlined will be a great reference source for future work.

Exercise 6

Use the same mtcars rownames ,find the total number of times “e” appears in that .

Exercise 7

Suppose you have a string like this
j <- "The_quick_brown_fox_jumps_over_the_lazy_dog"
split it in words using a stringr function

Exercise 8

On the same string I need the first word splitted but the rest intact ,help me to achieve that

Exercise 9

Now for on the same string J
a> I want the first “_” replaced by “–”
b> I want all the “_” replaced by “–”

Exercise 10

Many of the times ,you don’t want NA to appear when you do some string manipulation but its sometimes necessary to replace NA as a character(rather than remove it) ,stringr provides a useful tool for that.
Now if I have a vector like this ,
na_string_vec <- c("The_quick_brown_fox_jumps_over_the_lazy_dog",NA)

How can you turn the NA into a character string .

Data Visualization with googleVis exercises part 6

Geographical Charts

In part 6 of this series we are going to see some amazing geographical charts that googleVis provides.

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

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

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

Geo Chart

It is quite simple to create a Geo Chart with googleVis. We will use the "Exports" dataset. First let's take a look at it with head(Exports). As you can see there are three variables ("Country", "Profit", "Online") which we are going to use

later.
Look at the example below to create a simple geo chart:

```
Geo=gvisGeoChart(Exports )
plot(Geo)
```

Exercise 1

Create a list named “GeoC” and pass to it the “Exports” dataset as a geo chart. **HINT**: Use gvisGeoChart().

Exercise 2

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

Furthermore you can add much more information in your chart by using the locationvar and colorvar options to color the countries according to the their profit. Look at the example below.

```
Geo=gvisGeoChart(Exports,
locationvar="Country",
colorvar="Profit")
plot(Geo)
```

Exercise 3

Color the countries of your geo chart according to their profit and plot it. **HINT**: Use locationvar and colorvar.

Google Maps

It is quite simple to create a Google Map with googleVis. We will use the “Andrew” dataset. First let’s take a look at it with head(Andrew) to see its variables. Look at the example below to create a simple google map:

```
GoogleMap <- gvisMap(Andrew)
plot(GoogleMap)
```

Exercise 4

Create a list named “GoogleMap” and pass to it the “Andrew”

dataset as a google map. **HINT**: Use gvisMap().

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 5

Plot the the google map. **HINT**: Use plot().

As you can see there ane no data points on it as we did not select something yet. We have to select the latitude and longtitude variables for the dataset like the example below.
GoogleMap <- gvisMap(Andrew,"LatLong")

Exercise 6

Display the map by addind the “LatLong” variable to your list and plot it.

Exercise 7

Display the “Tip” variable on your google map just like you displayed the “LatLong” and plot it.

There are some useful options that gvisMap() provides to you that can enhance your map. Check the example below.
options=list(showTip=TRUE,
showLine=TRUE,
mapType='terrain',
useMapTypeControl=TRUE)

Exercise 8

Deactivate the Tip information from your map, plot the map and then enable it again. **HINT**: Use showTip.

Exercise 9

Enable useMapTypeControl and plot the map.

Exercise 10

Set the mapType by default to “terrain” and plot the map.