ggvis Exercises (Part-2)

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 and make a scatterplot. **HINT**: Use ggvis() and layer_points().

Exercise 2

Add a slider to the scatterplot of Exercise 1 that sets the point size from 10 to 100. **HINT**: Use input_slider().

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 3

Add a slider to the scatterplot of Exercise 1 that sets the point opacity from 0 to 1. **HINT**: Use input_slider().

Exercise 4

Create a histogram of the variable “Horsepower” of the “Cars93” data set. **HINT**: Use layer_histograms().

Exercise 5

Set the width and the center of the histogram bins you just created to 10.

Exercise 6

Add 2 sliders to the histogram you just created, one for width and the other for center with values from 0 to 10 and set the step to 1. **HINT**: Use input_slider().

Exercise 7

Add the labels “Width” and “Center” to the two sliders respectively. **HINT**: Use label.

Exercise 8

Create a scatterplot of the variables “Horsepower” and “MPG.city” of the “Cars93” dataset with size = 10 and opacity = 0.5.

Exercise 9

Add to the scatterplot you just created a function which will set the size with the left and right keyboard controls. **HINT**: Use left_right().

Exercise 10

Add interactivity to the scatterplot you just created using a function that shows the value of the "Horsepower" when you "mouseover" a certain point. **HINT**: Use add_tooltip().

---

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 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().

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 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.

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

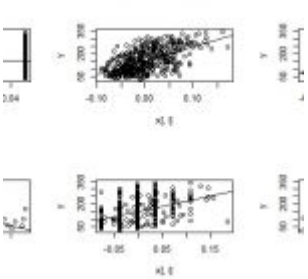
Enable useMapTypeControl and plot the map.

Exercise 10

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

---

iPlots exercises



INTRODUCTION

iPlots is a package which provides interactive statistical graphics, written in Java. You can find many interesting plots such as histograms, barcharts, scatterplots, boxplots, fluctuation diagrams, parallel coordinates plots and spineplots. The amazing part is that all of these plots support querying, linked highlighting, color brushing, and interactive changing of parameters.

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

Install and call the packages iplots and MASS in your working environment and then attach the dataset “Cars93”.

Exercise 2

Create a mosaic plot of the variables “AirBags”, “Cylinders” and “Origin” of the “Cars93” dataset. **HINT**: Use imosaic().

Exercise 3

Create a barchart of the variable “Fuel.tank.capacity” of the “Cars93” dataset. **HINT**: Use ibar().

Exercise 4

Get a spineplot of the barchart you created in Exercise 3. **HINT**: Use spineplot.

Exercise 5

See how the variable “Type” is ordered. **HINT**: Use levels().

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

- Work extensively with different packages to visualize your data
- Learn what visualizations exist for your specific use case
- And much more

Exercise 6

Reverse the order of the variable “Type”, name it “Type2” and check the order of “Type2”. **HINT**: Use ordered() and levels().

Exercise 7

Plot the barcharts of “Type” and “Type2” and spot the difference. **HINT**: Use ibar().

Exercise 8

Make a Parallel Coordinate Plot for all the continuous variables of “Cars93”. **HINT**: Use ipcp().

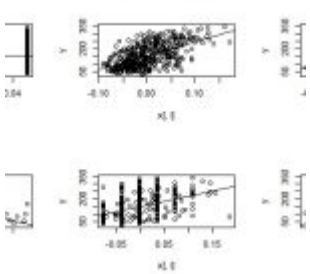
Exercise 9

Make a parallel boxplot for all “price” variables. **HINT**: Use ibox().

Exercise 10

Split the boxplot for “Wheelbase” by number of “Cylinders”. **HINT**: Use ibox().

How to create visualizations with iPlots package in R



INTRODUCTION

iPlots is a package which provides interactive statistical graphics, written in Java. You can find many interesting plots such as histograms, barcharts, scatterplots, boxplots, fluctuation diagrams, parallel coordinates plots and spineplots. The amazing part is that all of these plots support querying, linked highlighting, color brushing, and

interactive changing of parameters.

Furthermore, iPlots includes an API for managing plots and adding user-defined objects, such as lines or polygons to a plot.

PACKAGES & DATA

In order to use the iPlots package we have to install and call it. Moreover we need a dataset 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("iplots")
install.packages("MASS")
library(iplots)
library(MASS)
data("Cars93")
attach(Cars93)
```

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

PLOTS

Mosaic Plot

Some of the most important features that mosaic plots in iPlots provide are standard selection, highlighting, color brushing, reordering of variables and addition or exclusion of variables. An example of a mosaic plot with iPlots follows:

```
library(iplots)
library(MASS)
data("Cars93")
attach(Cars93)
imosaic(data.frame(AirBags,Cylinders,Origin))
```

iPlots through Model View also allows same Binsize, Fluctuation Diagram and Multiple Barchart. In order to rearrange the variables use the four arrow keys

Note that you can gain further information about your plot using -mouse-over and ctlr.

Barcharts

Barcharts in iPlots also feature Spineplots (use ctrl-s or the “View” menu to switch between the two representations). See the example below.

```
library(MASS)
data(Cars93)
attach(Cars93)
ibar(Cylinders)
```

To get a spineplot via the command line use:

```
ibar(Cylinders, isSpine=T)
```

You can reorder Bars with two ways. Firstly by using the options in the “View” menu and secondly by -dragging a bar to the position you wish.
Of course you can order the categories through R in a barchart. Look at the example below:

```
levels(AirBags)
[1] "Driver & Passenger" "Driver only" "None"

AirBags2 <- ordered(AirBags,
c("None", "Driver only", "Driver & Passenger"))

levels(AirBags2)
[1] "None" "Driver only" "Driver & Passenger"

ibar(AirBags)
ibar(AirBags2)
```

Parallel Plots

I. Parallel Coordinate Plot

A parallel coordinate plot connects all cases by lines. Look at the example below which creates a Parallel Coordinate Plot for the continuous variables “Cylinders” and “Passengers” of the “Cars93” dataset.

```
ipcp(Cars93[c(Cylinders, Passengers)])
```

Parallel Boxplot

Parallel boxplots are used to compare distributions of variables. Look how to create a parallel boxplot for all “MPG” variables of the”Cars93″ dataset

```
ibox(Cars93[c(7,8)])
```

All options can be found in the “View” menu of the plot. Note, that the scale is only displayed when all variables share the same scale!

III. Boxplot y by x

Boxplots y by x show boxplots by group. If ibox() is called with a continuous variable and a factor, a boxplot y by x is created. Look at the example to see how to split “Horsepower” by “Passengers”.

```
ibox(Horsepower, Passengers)
```

Note that a boxplot y by x always uses the same scale for all boxplots for a proper comparison.

Now, let’s move on to the first set of real exercises on the iPlots package!