MM 217: Descriptive statistics using R

In the getting started tutorial, towards the end of the tutorial, we did a couple of plots using R. In this tutorial, we will try and use the graphing and plotting capabilities of R to learn about descriptive statistics. In addition, we will also do some basic statistical analysis of the data to find out the centres and spreads of data.

plot() is the generic function for generating plots. Consider the first plot that you generated using the plot function in the last tutorial. It used a built-in data set called *islands*. Let us go through the commands you used and understand the details.

```
large.islands<-head(sort(islands,decreasing=TRUE),12)
plot(large.islands, main = "Land area of continents and islands", ylab = "Land area in square miles")
text(large.islands, labels=names(large.islands),adj=c(0.5,1))
```

In the first line, the data base *islands* was sorted in decreasing size, the first 12 data points from the top was chosen and a list called *large.islands* was generated. In the next line, the generated data was plotted and given a title and ylabel. In the third line, we have asked the data to be labelled by writing the name of the island next to the data point. Of coures, more information on plotting can be obtained using the command *help(plot)*. It is a good practice to use the help command and read up the given information; even though, initially, you may not understand all the information that help gives, as you learn more and more, you will become better at understanding the given details. So, I strongly recommend that at this point you should try the command help(plot). You may come back to this command as often as you feel like!

For our exercises, we also need data to work with. I have created a data called TerrorActionData and stored it as a csv file. The data looks as shown below. As you can see, this is the same data that we used in our lecture the other day.

Month	Year	Number of cases	Month with year
June	2019	44	June, 2019
May	2019	12	May, 2019
April	2019	19	April, 2019
March	2019	21	March, 2019
February	2019	31	Feb, 2019
January	2019	19	Jan, 2019
November	2018	17	Nov, 2018
September	2018	23	Sept, 2018
August	2018	16	Aug, 2018
April	2018	7	April, 2018
March	2018	23	March, 2018
February	2018	9	Feb, 2018
December	2017	2	Dec, 2017
November	2017	6	Nov, 2017
August	2017	22	Aug, 2017
June	2017	7	June, 2017
April	2017	27	April, 2017
March	2017	9	March, 2017
December	2016	10	Dec, 2016
November	2016	7	Nov, 2016
September	2016	2	Sept, 2016
August	2016	6	Aug, 2016
July	2016	38	July, 2016
June	2016	28	June, 2016

For all the exercises below, let us use this data. To use this data, first, we have to read this data. The following command reads this data from the csv file format.

```
X <- read.csv("TerrorActionData.csv")
```

After this, you can check that the data is read properly by typing X on the R window.

Now, let us do some analysis on the data. First, we want to extract the third column that consists of the number of cases that has been approved. Let us separate that data from the given data for carrying out further analysis:

```
x <- X[,3]
```

Again, by typing x on the R window you can confirm that the extraction is correct and complete. Let us now calculate the mean, median and standard deviation of the data.

```
mean(x)
median(x)
sd(x)
```

Note that there is no in-built function in R to get the mode. In fact, there is a command called mode and if you run on x, it justs tells you the format in which the data is stored. Use help(mode) to get more information on this aspect. However, it is easy to write a short function to get the mode of the given data. The following function does that.

```
CalcMode <- function(a) {
Unqa<- unique(a)
Unqa[which.max(tabulate(match(a,Unqa)))]
}
```

Here, find out what the function unique does using help(unique). The second line of the function finds out which data is repeated the maximum number of times. Now you can call this function to get the mode of the data:

```
CalcMode(x)
```

Now, using the information on mean, mode, median and standard deviation, can you comment on whether there is a qualitative change in number of terror cases approved in June 2019?

Plotting exercises

We will use the data to carry out some plotting exercises. Let us first generate the data for plotting by separating each of the columns out:

```
x<-X[,1]
y<-X[,2]
z<-X[,3]
```

These three commands take the first, second, and third column, respectively, and store them in the variables x, y, and z.

For plotting, let us use a package called *lattice*. To do so, we first load the package using the *library* function:

```
library(lattice)
```

Use the command help(library) to learn more about library.

Now, use the following command to generate a scatter plot of the data:

```
xyplot(x~z | y)
```

Also try the following command:

```
xyplot(y~z | x)
```

What are the differences between the two plots?

What happens, by the way, if you used the following command?

```
plot(X)
```

Now, let us add some trend lines to the data. To do so, we are going to use the full data that we loaded from the csv file.

```
xyplot(Month~Number.of.cases|Year, data=X,type=c("p","r"))
```

Here, the data is sorted according to the year, and the plot is consisting of points (indicated by “p”) and the regression line (indicated by “r”).

Now, let us consider the other types of plotting of the same data. Try the following commands and compare and contrast the plots that you get. In each case, use the help function and learn more about the commands.

```
bwplot(Month~Number.of.cases|Year, data=X)
barchart(Month~Number.of.cases|Year, data=X)
histogram(z)
levelplot(z~y*x)
cloud(Number.of.cases~Month*Year, data=X)
dotplot(~Number.of.cases | Year, data=X)
dotplot(~Number.of.cases | Month, data=X)
```

This gives a general idea of the different kinds of plots that are possible, even though this is not the exhaustive list.

Assignment / Home work: There are also some minor issues with these plots. For example, consider the last command: in the plot, the months appear alphabetically and not according to their chronological order. That is, the plots are in the order April, August, December, Februrary, ... and not January, February, March, ... Can we fix this? Similarly, in the bwplot, the plots are not labeled after the years but just the word Year is written. Can we fix this?