Exploratory Data Analysis in R for beginners (Part 1)

A Step-by-step Approach From Data Importing to Cleaning and Visualization

Joe Tran Oct 9, 2019 · 9 min read ★

Photo by Lukas Blazek on Unsplash

Exploratory Data Analysis (**EDA**) is the process of analyzing and visualizing the data to get a better understanding of the data and glean insight from it. There are various steps involved when doing EDA but the following are the common steps that a data analyst can take when performing EDA:

1. Import the data
2. Clean the data
3. Process the data
4. Visualize the data

What would you expect to find in this article?

This article focuses on EDA of a dataset, which means that it would involve all the steps mentioned above. Therefore, this article will walk you through all the steps required and the tools used in each step. So you would expect to find the followings in this article:

1. **Tidyverse** package for tidying up the data set
2. **ggplot2** package for visualizations
3. Some other basic functions to manipulate data like strsplit (), cbind (), matrix () and so on.

For beginners to EDA, *if you do not have a lot of time* and *do not know where to start*, I would recommend you to start with **Tidyverse** and **ggplot2**. You can do almost everything about EDA with these 2 packages. There are

various resources online like DataCamp, Setscholars, and books like Introduction to Data Science and so on.

In this article, I would walk you through the process of EDA through the analysis of the PISA score dataset which is available here.

Let's get started, folks!!

Importing the data

Before importing the data into R for analysis, let's look at how the data looks like:

Country Name	Country Code	Series Name	Series Code	2013 [YR2013]	2014 [YR2014]	2015 [YR2015]
Albania	ALB	PISA: Mean performance on the mathematics scale	LO.PISA.MAT	..	..	413.157
Albania	ALB	PISA: Mean performance on the mathematics scale. Female	LO.PISA.MAT.FE	..	..	417.7500295
Albania	ALB	PISA: Mean performance on the mathematics scale. Male	LO.PISA.MAT.MA	..	..	408.5454587
Albania	ALB	PISA: Mean performance on the reading scale	LO.PISA.REA	..	..	405.2588
Albania	ALB	PISA: Mean performance on the reading scale. Female	LO.PISA.REA.FE	..	..	434.6396255
Albania	ALB	PISA: Mean performance on the reading scale. Male	LO.PISA.REA.MA	..	..	375.7591992
Albania	ALB	PISA: Mean performance on the science scale	LO.PISA.SCI	..	..	427.225
Albania	ALB	PISA: Mean performance on the science scale. Female	LO.PISA.SCI.FE	..	..	439.4429629
Albania	ALB	PISA: Mean performance on the science scale. Male	LO.PISA.SCI.MA	..	..	414.9576437
Algeria	DZA	PISA: Mean performance on the mathematics scale	LO.PISA.MAT	..	..	359.6062
Algeria	DZA	PISA: Mean performance on the mathematics scale. Female	LO.PISA.MAT.FE	..	..	363.0724791
Algeria	DZA	PISA: Mean performance on the mathematics scale. Male	LO.PISA.MAT.MA	..	..	356.4951057
Algeria	DZA	PISA: Mean performance on the reading scale	LO.PISA.REA	..	..	349.8593
Algeria	DZA	PISA: Mean performance on the reading scale. Female	LO.PISA.REA.FE	..	..	366.2081668
Algeria	DZA	PISA: Mean performance on the reading scale. Male	LO.PISA.REA.MA	..	..	335.1854359
Algeria	DZA	PISA: Mean performance on the science scale	LO.PISA.SCI	..	..	375.7451
Algeria	DZA	PISA: Mean performance on the science scale. Female	LO.PISA.SCI.FE	..	..	383.2209389
Algeria	DZA	PISA: Mean performance on the science scale. Male	LO.PISA.SCI.MA	..	..	369.0352338
Argentina	ARG	PISA: Mean performance on the mathematics scale	LO.PISA.MAT	..	..	409.0333
Argentina	ARG	PISA: Mean performance on the mathematics scale. Female	LO.PISA.MAT.FE	..	..	400.4431161

Argentina	ARG	PISA: Mean performance on the mathematics scale. Male	LO.PISA.MAT.MA	..	..	418.3883609
Argentina	ARG	PISA: Mean performance on the reading scale	LO.PISA.REA	..	..	425.3031

When importing this data into R, we want the last column to be 'numeric' and the rest to be 'factor'. With this in mind, let's look at the following 3 scenarios:

```
df.raw <- read.csv(file ='Pisa scores 2013 - 2015 Data.csv',
fileEncoding="UTF-8-BOM", na.strings = '..')

df.raw1 <- read.csv(file ='Pisa scores 2013 - 2015 Data.csv')

df.raw2 <- read.csv(file ='Pisa scores 2013 - 2015
Data.csv',na.strings = '..')
```

These are 3 ways of importing the data into R. Usually, one with go for the df.raw1 because it seems to be the most convenient way of importing the data. Let's see the structure of the imported data:

```
df.raw1 <- read.csv(file ='Pisa scores 2013 - 2015 Data.csv')

str(df.raw1)
```

```
'data.frame':   1166 obs. of  7 variables:
 $ i..Country.Name: Factor w/ 132 levels "","Albania","Algeria",..: 2 2 2 2 2 2 2 2 2 3 ...
 $ Country.Code   : Factor w/ 130 levels "","ALB","ARE",..: 2 2 2 2 2 2 2 2 2 35 ...
 $ Series.Name    : Factor w/ 10 levels "","PISA: Mean performance on the mathematics scale",..: 2 3 4
 5 6 7 8 9 10 2 ...
 $ Series.Code    : Factor w/ 10 levels "","LO.PISA.MAT",..: 2 3 4 5 6 7 8 9 10 2 ...
 $ X2013..YR2013. : Factor w/ 2 levels "","..": 2 2 2 2 2 2 2 2 2 2 ...
 $ X2014..YR2014. : Factor w/ 2 levels "","..": 2 2 2 2 2 2 2 2 2 2 ...
 $ X2015..YR2015. : Factor w/ 614 levels "","..","325.586561113527",..: 122 144 106 96 206 39 182 217 1
29 20 ...
```

There are 2 problems that we can spot immediately. The last column is ‘factor’ and not ‘numeric’ like what we desire. Secondly, the first column ‘Country name’ is encoded differently from the raw dataset.

Now let’s try the second case scenario

```
df.raw2 <- read.csv(file ='Pisa scores 2013 - 2015
Data.csv',na.strings = '..')


str(df.raw2)
```

```
'data.frame':   1166 obs. of  7 variables:
 $ i..Country.Name: Factor w/ 132 levels "","Albania","Algeria",..: 2 2 2 2 2 2 2 2 2 3 ...
 $ Country.Code   : Factor w/ 130 levels "","ALB","ARE",..: 2 2 2 2 2 2 2 2 2 35 ...
 $ Series.Name    : Factor w/ 10 levels "","PISA: Mean performance on the mathematics scale",..: 2 3 4
 5 6 7 8 9 10 2 ...
 $ Series.Code    : Factor w/ 10 levels "","LO.PISA.MAT",..: 2 3 4 5 6 7 8 9 10 2 ...
 $ X2013..YR2013. : logi  NA NA NA NA NA NA ...
 $ X2014..YR2014. : logi  NA NA NA NA NA NA ...
 $ X2015..YR2015. : num  413 418 409 405 435 ...
```

The last column is now ‘numeric’. However, the name of the first column is not imported correctly.

What about the last scenario?

```
df.raw <- read.csv(file ='Pisa scores 2013 - 2015 Data.csv',
fileEncoding="UTF-8-BOM", na.strings = '..')
str(df.raw)
```

```
'data.frame':   1166 obs. of  7 variables:
 $ Country.Name  : Factor w/ 132 levels "","Albania","Algeria",..: 2 2 2 2 2 2 2 2 2 3 ...
 $ Country.Code  : Factor w/ 130 levels "","ALB","ARE",..: 2 2 2 2 2 2 2 2 2 35 ...
 $ Series.Name   : Factor w/ 10 levels "","PISA: Mean performance on the mathematics scale",..: 2 3 4 5
6 7 8 9 10 2 ...
 $ Series.Code   : Factor w/ 10 levels "","LO.PISA.MAT",..: 2 3 4 5 6 7 8 9 10 2 ...
 $ X2013..YR2013.: logi  NA NA NA NA NA NA ...
 $ X2014..YR2014.: logi  NA NA NA NA NA NA ...
 $ X2015..YR2015.: num  413 418 409 405 435 ...
```

As you can see, the first column is now named properly and the last column is ‘numeric’.

`na.strings = '..'`

This allows R to replace those blanks in the dataset with NA. This will be useful and convenient later when we want to remove all the 'NA's.

```
fileEncoding="UTF-8-BOM"
```

This allows R, in the laymen term, to read the characters as correctly as they would appear on the raw dataset.

Cleaning and Processing the data

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

We want to do a few things to clean the dataset:

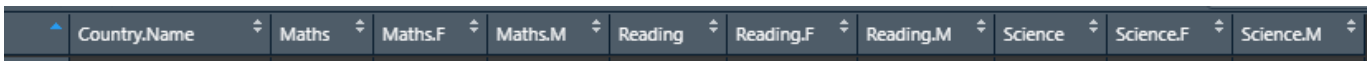
1. Make sure that each row in the dataset corresponds to ONLY one country: Use spread() function in tidyverse package

2. Make sure that only useful columns and rows are kept: Use drop_na() and data subsetting
3. Rename the Series Code column for meaningful interpretation: Use rename()

```
df <- df.raw[1:1161, c(1, 4, 7)] #select relevant rows and cols
    %>%  spread(key=Series.Code, value=X2015..YR2015.)
    %>%  rename(Maths = LO.PISA.MAT,
         Maths.F = LO.PISA.MAT.FE,
         Maths.M = LO.PISA.MAT.MA,
         Reading = LO.PISA.REA,
         Reading.F = LO.PISA.REA.FE,
         Reading.M = LO.PISA.REA.MA,
         Science = LO.PISA.SCI,
         Science.F = LO.PISA.SCI.FE,
         Science.M = LO.PISA.SCI.MA
         ) %>%
  drop_na()
```

Now let's see how the clean data looks like:

```
view(df)
```



1	Albania	413.1570	417.7500	408.5455	405.2588	434.6396	375.7592	427.2250	439.4430	414.9576
2	Algeria	359.6062	363.0725	356.4951	349.8593	366.2082	335.1854	375.7451	383.2209	369.0352
3	Argentina	409.0333	400.4431	418.3884	425.3031	432.9581	416.9666	432.2262	424.9944	440.1020
5	Australia	493.8962	490.9855	496.7613	502.9006	518.8658	487.1855	509.9939	508.9216	511.0493
6	Austria	496.7423	483.1330	510.0982	484.8656	495.0752	474.8460	495.0375	485.5268	504.3712
9	Belgium	506.9844	499.7390	514.0026	498.5242	506.6386	490.6642	501.9997	496.0319	507.7805
15	Brazil	377.0695	369.5493	385.0406	407.3486	418.5617	395.4633	400.6821	398.7000	402.7830
16	Bulgaria	441.1899	442.1631	440.3189	431.7175	456.5986	409.4498	445.7720	453.9011	438.4966
20	Canada	515.6474	511.1417	520.1661	526.6678	539.7624	513.5355	527.7047	527.1562	528.2548
22	Chile	422.6714	413.4490	431.7981	458.5709	464.5616	452.6422	446.9561	439.6174	454.2186
23	Colombia	389.6438	384.4883	395.3911	424.9052	432.2819	416.6816	415.7288	411.0316	420.9651
26	Costa Rica	400.2534	392.3129	408.4516	427.4875	434.8748	419.8605	419.6080	410.8349	428.6660
28	Croatia	464.0401	457.9612	470.5987	486.8632	499.5858	473.1367	475.3912	472.5863	478.4173
30	Cyprus	437.1443	439.5341	434.7064	442.8443	468.6583	416.8271	432.5964	440.9482	424.1478
31	Czech Republic	492.3254	488.6656	495.7942	487.2501	500.6527	474.5475	492.8300	488.3983	497.0304
32	Denmark	511.0876	506.3748	515.7565	499.8146	510.9516	488.7816	501.9369	498.9027	504.9427
33	Dominican Republic	327.7020	329.7459	325.5866	357.7377	372.7806	342.1682	331.6388	330.8290	332.4770
37	Estonia	519.5291	516.8728	522.0804	519.1429	533.3620	505.4863	534.1937	532.5228	535.7986
39	Finland	511.0769	514.9650	507.4528	526.4247	550.5112	503.9746	530.6612	540.5118	521.4797
40	France	492.9204	489.9540	495.9317	499.3061	513.7640	484.6293	494.9776	494.0342	495.9353
42	Georgia	403.8332	410.5960	397.7478	401.2881	431.8820	373.7585	411.1315	419.6164	403.4965
43	Germany	505.9713	497.5311	514.1177	509.1041	519.6741	498.9021	509.1406	503.8121	514.2837
45	Greece	453.6299	453.5732	453.6821	467.0395	486.4600	449.1362	454.8288	459.4177	450.5984
49	Hong Kong SAR, China	547.9310	546.7682	549.0658	526.6753	540.9844	512.7113	523.2774	523.7491	522.8172
50	Hungary	476.8309	472.7395	480.9055	469.5233	481.9596	457.1377	476.7475	475.2484	478.2405
51	Iceland	488.0332	488.5870	487.4457	481.5255	501.7167	460.1036	473.2301	474.6556	471.7177
52	Indonesia	386.1096	387.4450	384.7793	397.2595	408.9994	385.5642	403.0997	405.1289	401.0783
54	Ireland	503.7220	495.4450	511.5797	520.8148	526.9491	514.9914	502.5751	497.1740	507.7026
55	Israel	469.6695	465.5169	473.9902	478.9606	490.1650	467.3026	466.5528	464.4477	468.7432
56	Italy	489.7287	479.8237	499.7621	484.7580	492.7091	476.7038	480.5468	472.1190	489.0838
57	Japan	532.4399	525.4960	539.2673	515.9585	522.6553	509.3740	538.3948	531.5329	545.1415
58	Jordan	380.2590	387.3772	373.0013	408.1022	443.5964	371.9124	408.6691	427.9967	388.9627

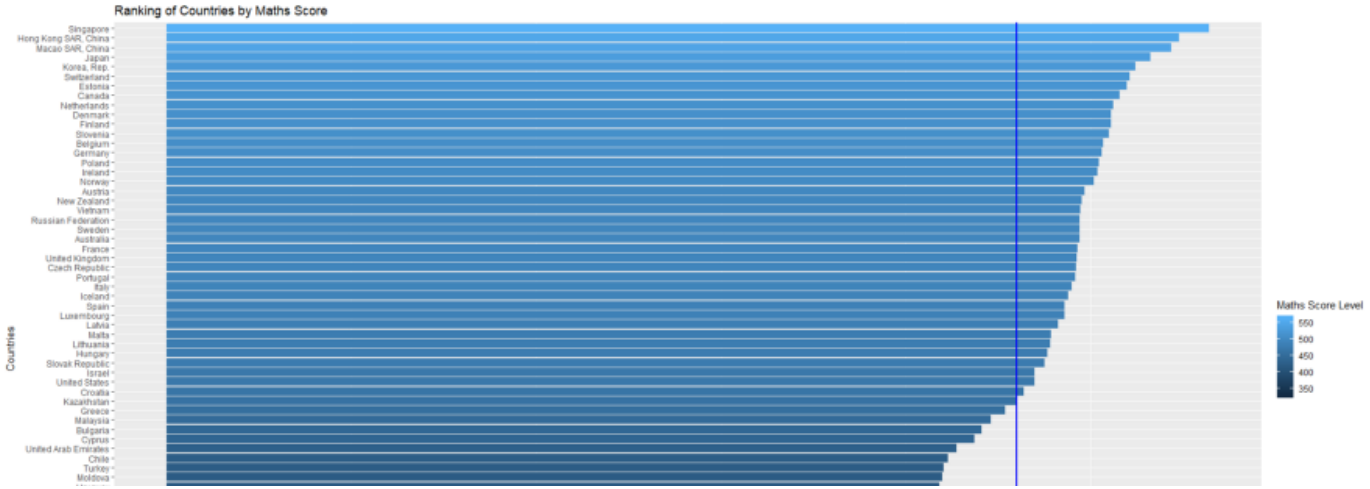
Visualizing the data

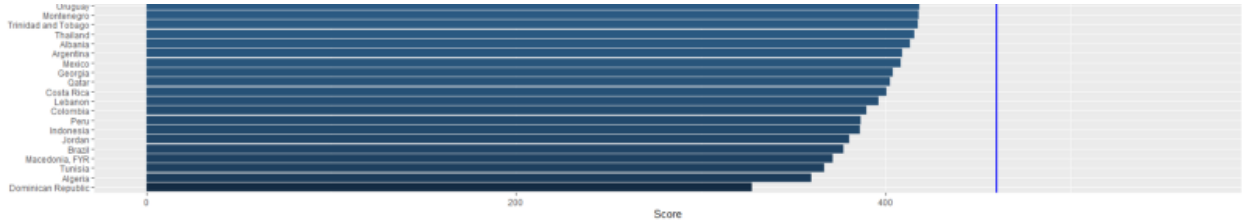
1. Barplot

```
install.packages("ggplot2")
library(ggplot2)


#Ranking of Maths Score by Countries


ggplot(data=df,aes(x=reorder(Country.Name,Maths),y=Maths)) +
  geom_bar(stat ='identity',aes(fill=Maths))+
  coord_flip() +
  theme_grey() +
  scale_fill_gradient(name="Maths Score Level")+
  labs(title = 'Ranking of Countries by Maths Score',
       y='Score',x='Countries')+
  geom_hline(yintercept = mean(df$Maths),size = 1, color = 'blue')
```





Similarly, we can rank by science score and reading score too, just change the name accordingly.

2. Boxplot

If we use the dataset above, we will not be able to draw a boxplot. This is because boxplot needs only 2 variables x and y but in the cleaned data that we have, there are so many variables. So we need to combine those into 2 variables. We name this as **df2**

```
df2 = df[,c(1,3,4,6,7,9,10)] %>%   # select relevant columns
  pivot_longer(c(2,3,4,5,6,7),names_to = 'Score')

view(df2)
```

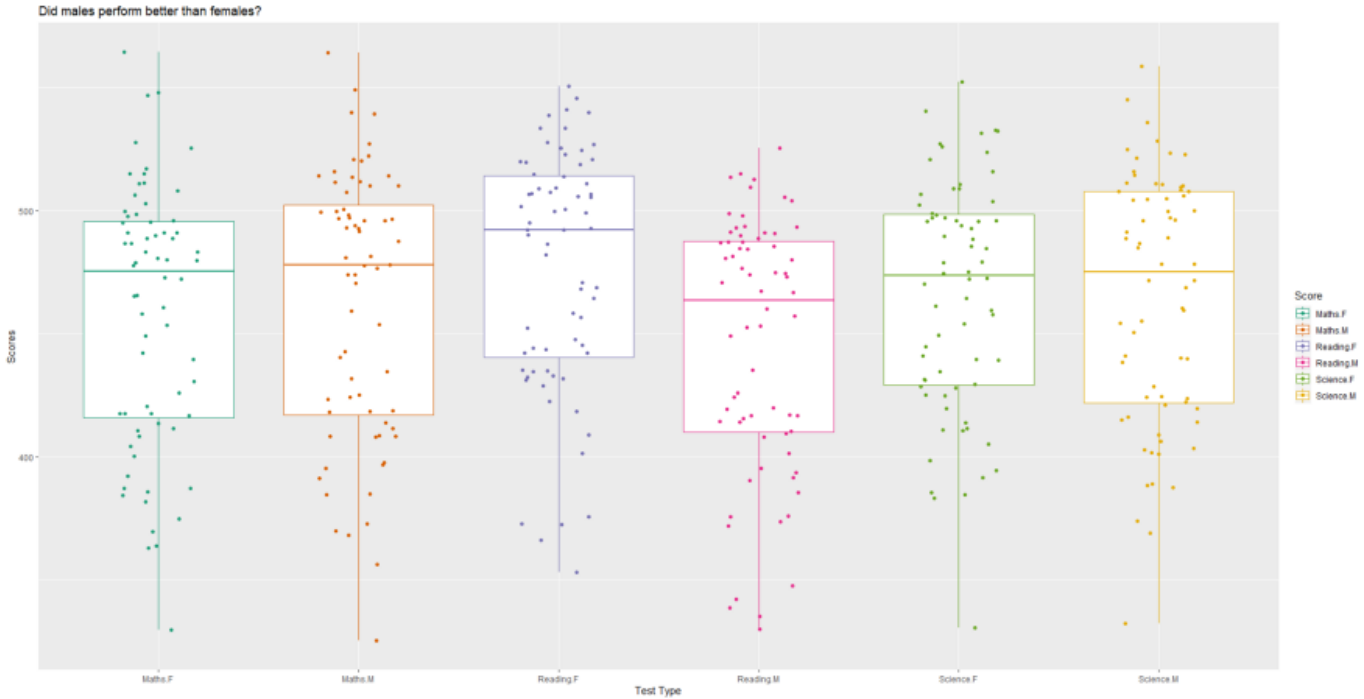
	Country.Name	Score	value
1	Albania	Maths.F	417.7500

2	Albania	Maths.M	408.5455
3	Albania	Reading.F	434.6396
4	Albania	Reading.M	375.7592
5	Albania	Science.F	439.4430
6	Albania	Science.M	414.9576
7	Algeria	Maths.F	363.0725
8	Algeria	Maths.M	356.4951
9	Algeria	Reading.F	366.2082
10	Algeria	Reading.M	335.1854
11	Algeria	Science.F	383.2209
12	Algeria	Science.M	369.0352
13	Argentina	Maths.F	400.4431
14	Argentina	Maths.M	418.3884
15	Argentina	Reading.F	432.9581
16	Argentina	Reading.M	416.9666
17	Argentina	Science.F	424.9944
18	Argentina	Science.M	440.1020
19	Australia	Maths.F	490.9855
20	Australia	Maths.M	496.7613
21	Australia	Reading.F	518.8658
22	Australia	Reading.M	487.1855
23	Australia	Science.F	508.9216
24	Australia	Science.M	511.0493
25	Austria	Maths.F	483.1330
26	Austria	Maths.M	510.0982
27	Austria	Reading.F	495.0752
28	Austria	Reading.M	474.8460
29	Austria	Science.F	485.5268
30	Austria	Science.M	504.3712
31	Belgium	Maths.F	499.7390

32	Belgium	Maths.M	514.0026
33	Belgium	Reading.F	506.6286

Great! Now we can make boxplots

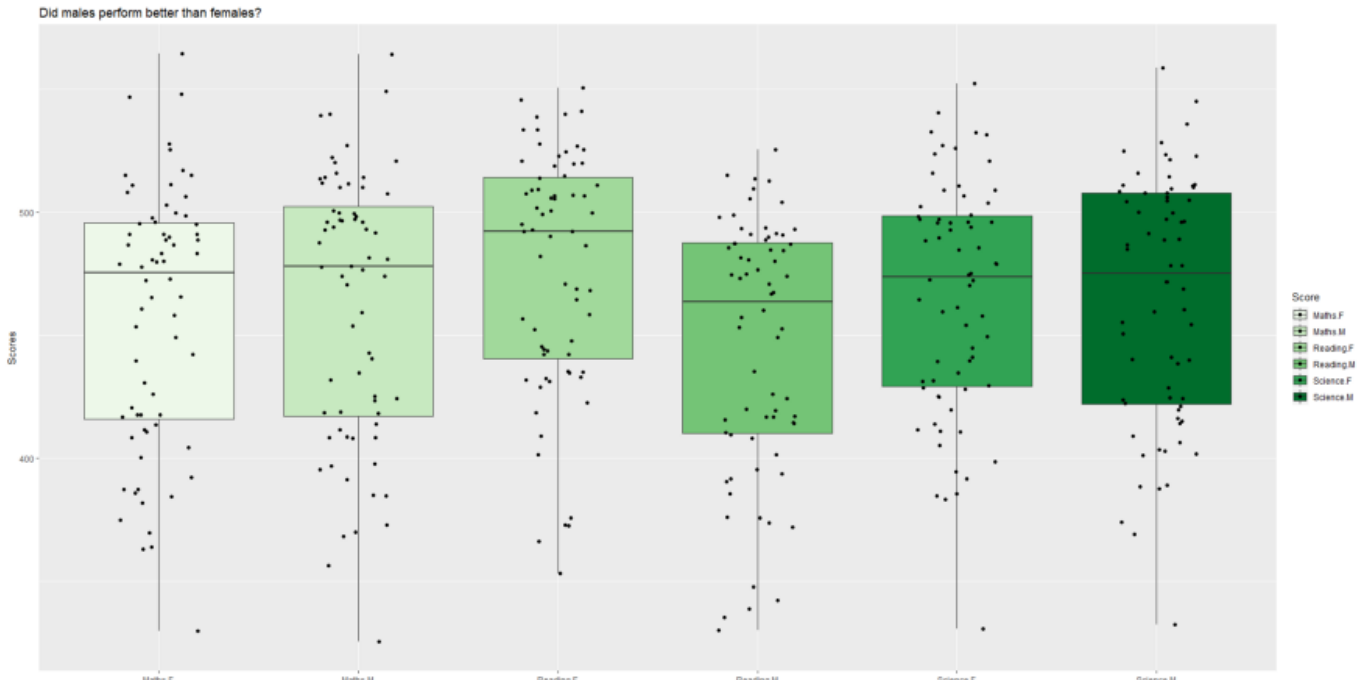
```
ggplot(data = df2, aes(x=Score,y=value, color=Score)) +
  geom_boxplot()+
  scale_color_brewer(palette="Dark2") +
  geom_jitter(shape=16, position=position_jitter(0.2))+
  labs(title = 'Did males perform better than females?',
       y='Scores',x='Test Type')
```



geom_jitter() allows you to plot the data points on the plot.

You can play around with the code above to get various plots. For example, I can change from 'color = Score' to 'fill=Score':

```
ggplot(data = df2, aes(x=Score,y=value, fill=Score)) +
  geom_boxplot()+
  scale_fill_brewer(palette="Green") +
  geom_jitter(shape=16, position=position_jitter(0.2))+
  labs(title = 'Did males perform better than females?',
       y='Scores',x='Test Type')
```



The plot looks a bit messy. A better visualisation would be to separate Subjects and Genders and plot them side by side.

How can we do that?

Since we want to separate Subjects and Genders from a column containing 'Subject.Gender' (e.g. Maths.F), we need to use strsplit () to do this job for us

```
S = numeric(408)     # create an empty vector
for (i in 1:length(df2$Score)) {
  S[i] = strsplit(df2$Score[i],".",fixed = TRUE)
}
```

Now S is a list of 408 components, each of which has 2 sub-components 'Subject' and 'Gender'. We need to transform S into a data frame with 1 column of **Subject** and 1 column of **Gender**. We will name this data frame as **df3**

```
df3 = S%>%unlist() %>% matrix(ncol = 2, byrow = TRUE)%>%
as.data.frame()

view(df3)
```

	V1	V2
1	Maths	F
2	Maths	M
3	Reading	F
4	Reading	M
5	Science	F
6	Science	M
7	Maths	F
8	Maths	M
9	Reading	F
10	Reading	M
11	Science	F
12	Science	M
13	Maths	F
14	Maths	M
15	Reading	F
16	Reading	M
17	Science	F
18	Science	M
19	Maths	F
20	Maths	M
21	Reading	F

22	Reading	M
23	Science	F
24	Science	M
25	Maths	F
26	Maths	M
27	Reading	F
28	Reading	M
29	Science	F
30	Science	M
31	Maths	F
32	Maths	M

We now need to combine this **df3** with **df2** we created earlier, and name the result as **df4**

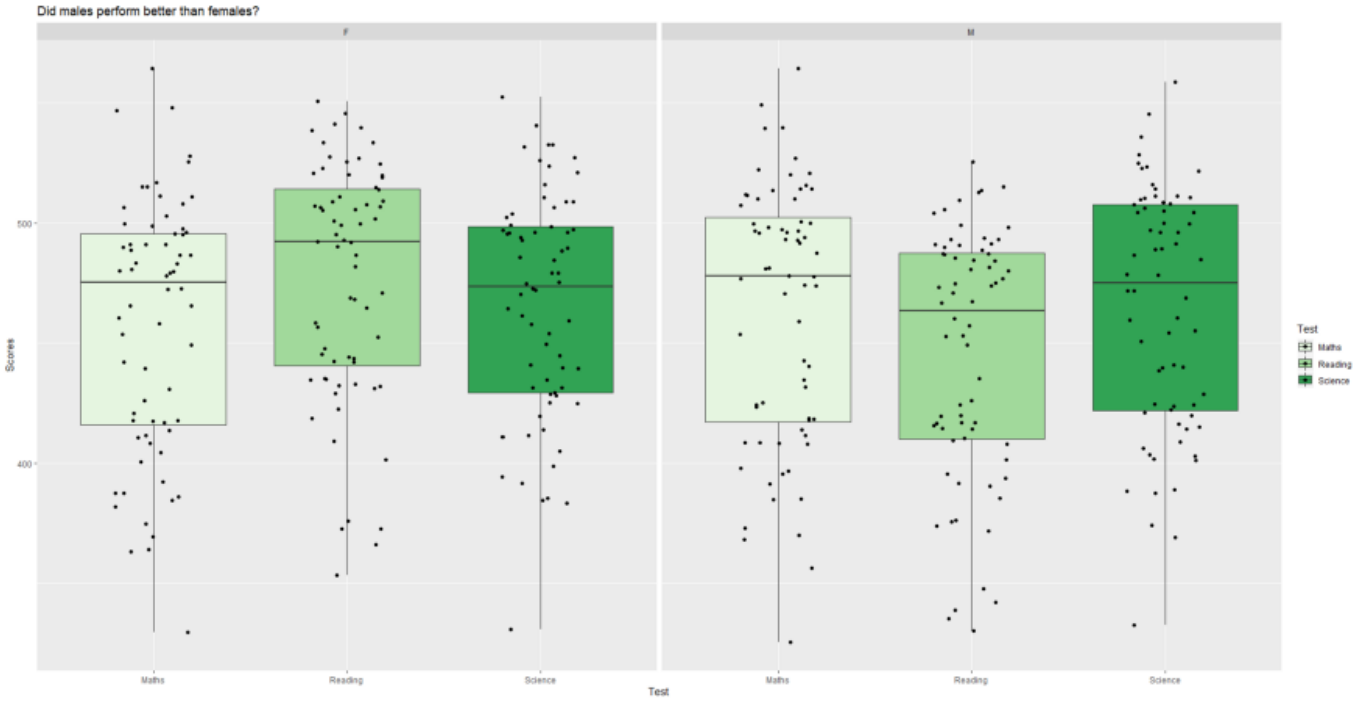
```
df4 = cbind(df2,df3)
colnames(df4) = c('Country','Score','Value','Test','Gender')
df4$Score = NULL # since the 'Score' column is redundant
view(df4)
```

	Country	Value	Test	Gender
1	Albania	417.7500	Maths	F
2	Albania	408.5455	Maths	M
3	Albania	434.6396	Reading	F
4	Albania	375.7592	Reading	M

5	Albania	439.4430	Science	F
6	Albania	414.9576	Science	M
7	Algeria	363.0725	Maths	F
8	Algeria	356.4951	Maths	M
9	Algeria	366.2082	Reading	F
10	Algeria	335.1854	Reading	M
11	Algeria	383.2209	Science	F
12	Algeria	369.0352	Science	M
13	Argentina	400.4431	Maths	F
14	Argentina	418.3884	Maths	M
15	Argentina	432.9581	Reading	F
16	Argentina	416.9666	Reading	M
17	Argentina	424.9944	Science	F
18	Argentina	440.1020	Science	M
19	Australia	490.9855	Maths	F
20	Australia	496.7613	Maths	M
21	Australia	518.8658	Reading	F
22	Australia	487.1855	Reading	M
23	Australia	508.9216	Science	F
24	Australia	511.0493	Science	M
25	Austria	483.1330	Maths	F
26	Austria	510.0982	Maths	M
27	Austria	495.0752	Reading	F
28	Austria	474.8460	Reading	M
29	Austria	485.5268	Science	F
30	Austria	504.3712	Science	M
31	Belgium	499.7390	Maths	F

Awesome! Now the data looks clean and neat. Let's create **multiple plots** with the use of **facet_wrap()** function in **ggplot2**

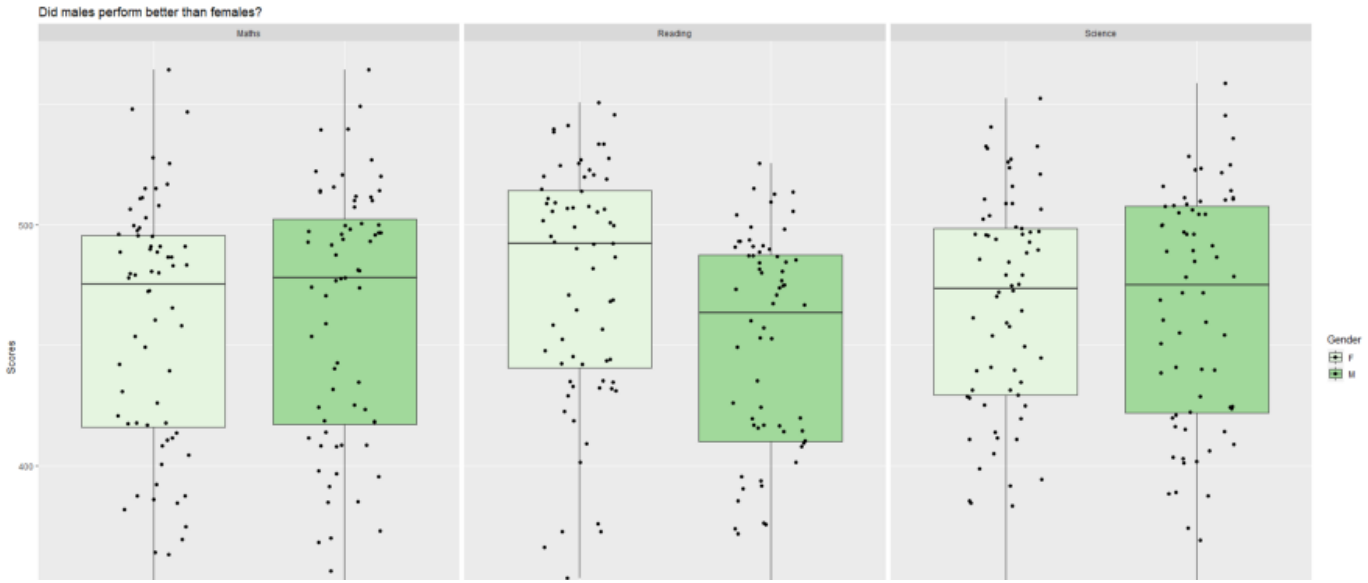
```
ggplot(data = df4, aes(x=Test,y=Value, fill=Test)) +
  geom_boxplot()+
  scale_fill_brewer(palette="Green") +
  geom_jitter(shape=16, position=position_jitter(0.2))+
  labs(title = 'Did males perform better than females?',
       y='Scores',x='Test')+
  facet_wrap(~Gender,nrow = 1)
```



Here, we categorized the plot by Gender, hence
facet_wrap(~Gender,nrow = 1)

We can also categorize the plot by Test by changing
facet_wrap(~Test,nrow = 1)

```
ggplot(data = df4, aes(x=Gender,y=Value, fill=Gender)) +
  geom_boxplot()+
  scale_fill_brewer(palette="Green") +
  geom_jitter(shape=16, position=position_jitter(0.2))+
  labs(title = 'Did males perform better than females?',
       y='Scores',x='')+
  facet_wrap(~Test,nrow = 1)
```



By looking at these plots, we can make some insights about the performance of males and females. Generally, males performed better in Science and Maths, but females performed better in Reading. However, it would be naive for us to make a conclusion only after looking at the boxplot. Let's dive deeper into the data and see any other insights we can get after manipulating the dataset.

Since I want to compare the performance of Males and Females in each subject across all participating countries, I would need to calculate the % difference in terms of the score for each subject between males and females and then plot it out to visualize.

How would I do this in R? Use mutate() function

Let's look at the original clean data set that we have, **df**

We will now use the mutate() function to calculate, for each country, the % difference between Males and Females for each subject.

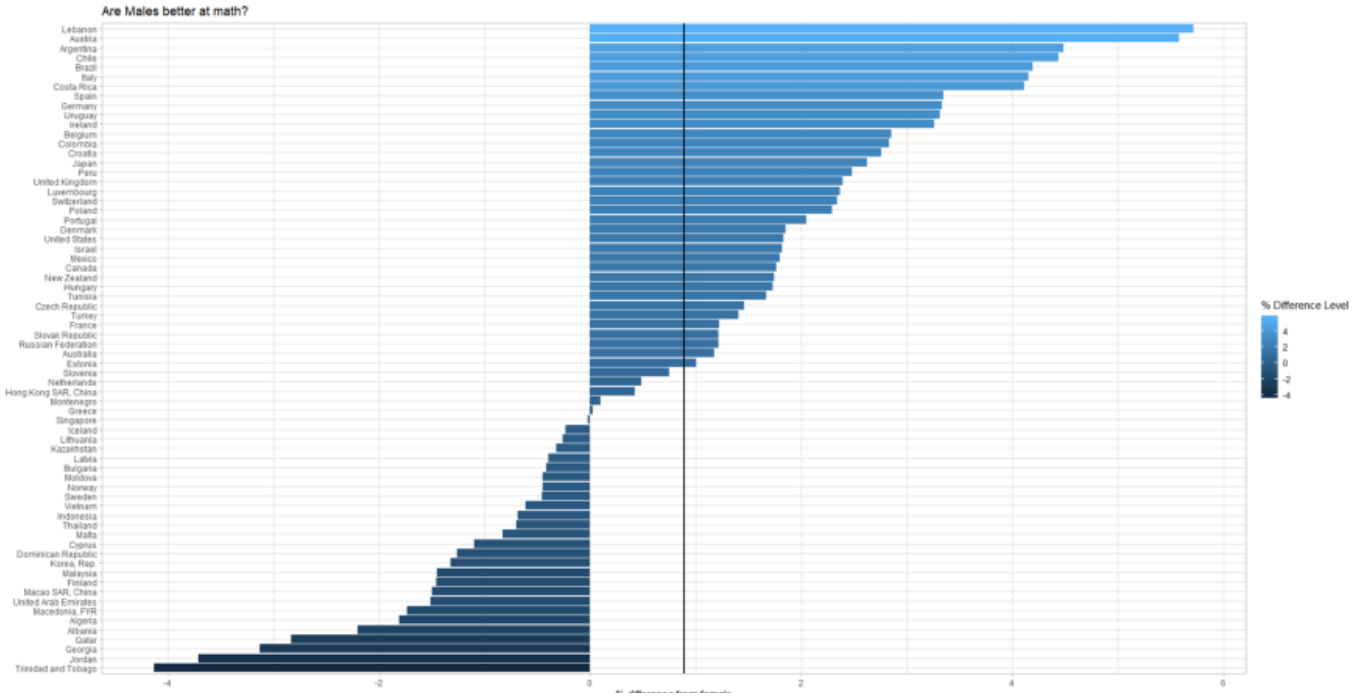
```
df = df %>% mutate(Maths.Diff = ((Maths.M - Maths.F)/Maths.F)*100,
        Reading.Diff = ((Reading.M - Reading.F)/Reading.F)*100,
        Science.Diff = ((Science.M - Science.F)/Science.F)*100,
        Total.Score = Maths + Reading + Science,
        Avg.Diff = (Maths.Diff+Reading.Diff+Science.Diff)/3
        )

view(df)
```

	Country.Name	Maths	Maths.F	Maths.M	Reading	Reading.F	Reading.M	Science	Science.F	Science.M	Maths.Dif	Reading.Dif	Science.Dif	Total.Score	Avg.Dif
1	Albania	413.1570	417.7500	408.5455	405.2588	434.6396	375.7592	427.2250	439.4430	414.9576	-2.20336807	-13.546953	-5.5718993	1245.641	-7.10740682
2	Algeria	359.6062	363.0725	356.4951	349.8593	366.2082	335.1854	375.7451	383.2209	369.0352	-1.81158688	-8.471338	-3.7017041	1085.211	-4.66154293
3	Argentina	409.0333	400.4431	418.3884	425.3031	432.9581	416.9666	432.2262	424.9944	440.1020	4.48134681	-3.693538	3.5547951	1266.563	1.44753454
5	Australia	493.8962	490.9855	496.7613	502.9006	518.8658	487.1855	509.9939	508.9216	511.0493	1.17637774	-6.105678	0.4180624	1506.791	-1.50374589
6	Austria	496.7423	483.1330	510.0982	484.8656	495.0752	474.8460	495.0375	485.5268	504.3712	5.58131786	-4.086078	3.8812368	1476.645	1.79215881
9	Belgium	506.9844	499.7390	514.0026	498.5242	506.6386	490.6642	501.9997	496.0319	507.7805	2.85419970	-3.153025	2.3685191	1507.508	0.68969785
15	Brazil	377.0695	369.5493	385.0406	407.3486	418.5617	395.4633	400.6821	398.7000	402.7830	4.19193941	-5.518524	1.0240679	1185.100	-0.10083227
16	Bulgaria	441.1899	442.1631	440.3189	431.7175	456.5986	409.4498	445.7720	453.9011	438.4966	-0.41708128	-10.326091	-3.3937938	1318.679	-4.71232193
20	Canada	515.6474	511.1417	520.1661	526.6678	539.7624	513.5355	527.7047	527.1562	528.2548	1.76553766	-4.858977	0.2083976	1570.020	-0.96168041
22	Chile	422.6714	413.4490	431.7981	458.5709	464.5616	452.6422	446.9561	439.6174	454.2186	4.43805793	-2.565731	3.3213455	1328.198	1.73122426
23	Colombia	389.6438	384.4883	395.3911	424.9052	432.2819	416.6816	415.7288	411.0316	420.9651	2.83566057	-3.608825	2.4167274	1230.278	0.54785415
26	Costa Rica	400.2534	392.3129	408.4516	427.4875	434.8748	419.8605	419.6080	410.8349	428.6660	4.11374502	-3.452567	4.3402113	1247.349	1.66712970
28	Croatia	464.0401	457.9612	470.5987	486.8632	499.5858	473.1367	475.3912	472.5863	478.4173	2.75950811	-5.294206	1.2338441	1426.294	-0.43361807
30	Cyprus	437.1443	439.5341	434.7064	442.8443	468.6583	416.8271	432.5964	440.9482	424.1478	-1.09834996	-11.059504	-3.8100543	1312.585	-5.32263594
31	Czech Republic	492.3254	488.6656	495.7942	487.2501	500.6527	474.5475	492.8300	488.3983	497.0304	1.45878076	-5.214233	1.7674354	1472.405	-0.66267223
32	Denmark	511.0876	506.3748	515.7565	499.8146	510.9516	488.7816	501.9369	498.9027	504.9427	1.85272237	-4.338958	1.2106588	1512.839	-0.42519213
33	Dominican Republic	327.7020	329.7459	325.5866	357.7377	372.7806	342.1682	331.6388	330.8290	332.4770	-1.26138622	-8.211916	0.4981496	1017.079	-2.99171751
37	Estonia	519.5291	516.8728	522.0804	519.1429	533.3620	505.4863	534.1937	532.5228	535.7986	1.00753236	-5.226405	0.6151492	1572.866	-1.20124122
39	Finland	511.0769	514.9650	507.4528	526.4247	550.5112	503.9746	530.6612	540.5118	521.4797	-1.45877024	-8.453340	-3.5211343	1568.163	-4.47774821
40	France	492.9204	489.9540	495.9317	499.3061	513.7640	484.6293	494.9776	494.0342	495.9353	1.22004196	-5.670840	0.3848057	1487.204	-1.35533066
42	Georgia	403.8332	410.5960	397.7478	401.2881	431.8820	373.7585	411.1315	419.6164	403.4965	-3.12916425	-13.458177	-3.8415909	1216.253	-6.80964401
43	Germany	505.9713	497.5311	514.1177	509.1041	519.6741	498.9021	509.1406	503.8121	514.2837	3.33377634	-3.997113	2.0784633	1524.216	0.47170880
45	Greece	453.6299	453.5732	453.6821	467.0395	486.4600	449.1362	454.8288	459.4177	450.5984	0.02399463	-7.672528	-1.9196790	1375.498	-3.18940428
49	Hong Kong SAR, China	547.9310	546.7682	549.0658	526.6753	540.9844	512.7113	523.2774	523.7491	522.8172	0.42022046	-5.226229	-0.1779279	1597.884	-1.66131228
50	Hungary	476.8309	472.7395	480.9055	469.5233	481.9596	457.1377	476.7475	475.2484	478.2405	1.72738617	-5.150190	0.6295895	1423.102	-0.93107137
51	Iceland	488.0332	488.5870	487.4457	481.5255	501.7167	460.1036	473.2301	474.6556	471.7177	-0.23359422	-8.294143	-0.6189646	1442.789	-3.04890065
52	Indonesia	386.1096	387.4450	384.7793	397.2595	408.9994	385.5642	403.0997	405.1289	401.0783	-0.68801937	-5.729900	-0.9998441	1186.469	-2.47258789

Now let's plot this out to visualize the data better

```
##### MATHS SCORE #####
ggplot(data=df, aes(x=reorder(Country.Name, Maths.Diff),
y=Maths.Diff)) +
  geom_bar(stat = "identity", aes(fill=Maths.Diff)) +
  coord_flip() +
  theme_light() +
  geom_hline(yintercept = mean(df$Maths.Diff), size=1, color="black")
+
  scale_fill_gradient(name="% Difference Level") +
  labs(title="Are Males better at math?", x="", y="% difference from
female")
```



This plot represents the % difference in scores using **Females** as **reference**. A **positive difference means males scored higher, while a negative difference means males scored lower**. The black line represents the mean difference across all countries.

Some interesting insights one can draw from here:

- In general, Males performed better than Females in Maths, especially in most of the DCs, males scored generally higher than females
- Interestingly, in Singapore and Hongkong, females and males performed equally well, with the difference in score is around 0 in each of these countries. This is good insight maybe for policy-makers because we do not want a huge gap between males and females performance in education.

We can do the same thing for Reading and Science score.

3. Correlation Plot

```
df = df[,c(1,3,4,6,7,9,10)] #select relevant columns
```

To create correlation plot, simply use cor():

```
res = cor(df[,-1]) # -1 here means we look at all columns except the
first column

res
```

	Maths.F	Maths.M	Reading.F	Reading.M	Science.F	Science.M
Maths.F	1.0000000	0.9845874	0.9377498	0.9177645	0.9711420	0.9547097
Maths.M	0.9845874	1.0000000	0.9312678	0.9468078	0.9576479	0.9758210
Reading.F	0.9377498	0.9312678	1.0000000	0.9663211	0.9555957	0.9440488
Reading.M	0.9177645	0.9468078	0.9663211	1.0000000	0.9283812	0.9692920
Science.F	0.9711420	0.9576479	0.9555957	0.9283812	1.0000000	0.9736539
Science.M	0.9547097	0.9758210	0.9440488	0.9692920	0.9736539	1.0000000

We can calculate p-value to see whether the correlation is significant

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

res2 <- rcorr(as.matrix(df[,-1]))
```

```
           Maths.F Maths.M Reading.F Reading.M Science.F Science.M
Maths.F       1.00    0.98      0.94      0.92      0.97      0.95
Maths.M       0.98    1.00      0.93      0.95      0.96      0.98
Reading.F     0.94    0.93      1.00      0.97      0.96      0.94
Reading.M     0.92    0.95      0.97      1.00      0.93      0.97
Science.F     0.97    0.96      0.96      0.93      1.00      0.97
Science.M     0.95    0.98      0.94      0.97      0.97      1.00

n= 68


P
           Maths.F Maths.M Reading.F Reading.M Science.F Science.M
Maths.F              0       0         0         0         0
Maths.M     0                0         0         0         0
Reading.F   0        0                0         0         0
Reading.M   0        0       0                  0         0
Science.F   0        0       0         0                   0
Science.M   0        0       0         0         0
```

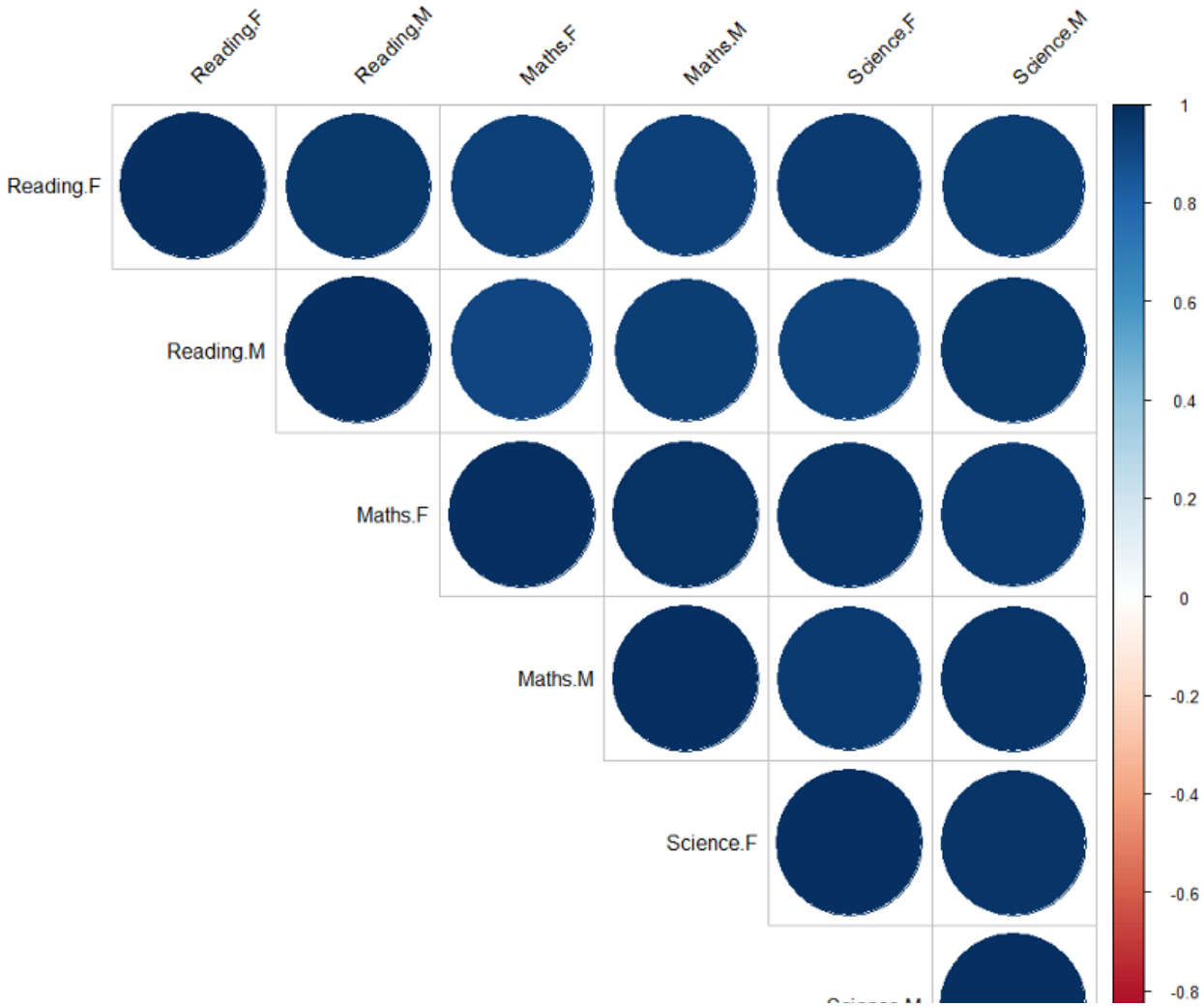
The smaller the p-value, the more significant the correlation.

The purpose of this section about correlation plot is to introduce to you how to calculate correlation between variables in R. For this dataset, it is obvious that all the variables are correlated

To visualize

```
install.packages("corrplot")

library(corrplot)
corrplot(res, type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)
```



The stronger the color and the bigger the size, the higher the correlation. The result is similar to the one we got earlier: All the variables are intercorrelated.

That is it! Hope you guys enjoyed and picked up something from this article. While this guide is not exhaustive, it will more or less give you some ideas of how to do some basic EDA with R.

If you have any questions, feel free to put them down in the comment section below. Thank you for your read. Have a great day and happy programming!!!

> *Check out Part 2 of this series here*
>
> *Check out my other posts on Random Forest and Forecasting if you are interested!*

Data Science Eda Towards Data Science Data Visualization