1.5 Exploratory Data Analysis

Objectives:

- Explore your data
- Visualise your data

Contents:

- Exploratory functions
- Data visualisation

Exploratory functions

The first thing to do before doing any analysis is to explore your data. This is an important step whether you want to explore a dataset that has been given to you (secondary analysis) or you want to check the data from your own experiment.

R comes with simple commands to preview and summarise the dataset, but you can also load some packages that give more detailed or different summaries (skimr, DataExplorer). Previewing data becomes important when dealing with a big dataset and you cannot open it or visualise it all at once.

Suppose you have been given the `happiness report` dataset from 2016 and been asked to report what's in before digging into the analysis.

Let's explore the data together.

1. First, you need to read the data from the happiness_report.xlsx file in the 1.5 folder.

```
#Read happiness files for 2016
#make sure you closed the xls file to read it in R
happiness_2016=read_excel("happiness_report.xlsx", sheet = "2016")
```

2. Use the functions `head()` and `tail()` to discover the first and the last 6 rows of the dataset. You can change the number of rows you want to see by adding it as an argument in the function `head()`, remember you can always ask R for help with the function `help(head)` or `?head`. You can also visualise a random sample from the data.

```
#Discover the first 6 rows of your df
> head(happiness_2016)
# A tibble: 6 x 13
```

```
  Country Region `Happiness Rank` `Happiness Scor~ `Lower Confiden~
  <chr>   <chr>             <dbl>            <dbl>            <dbl>
1 Denmark Weste~                1             7.53             7.46
2 Switze~ Weste~                2             7.51             7.43
3 Iceland Weste~                3             7.50             7.33
4 Norway  Weste~                4             7.50             7.42
5 Finland Weste~                5             7.41             7.35
6 Canada  North~                6             7.40             7.34
# ... with 8 more variables: `Upper Confidence Interval` <dbl>,
#   `Economy (GDP per Capita)` <dbl>, Family <dbl>, `Health (Life
#   Expectancy)` <dbl>, Freedom <dbl>, `Trust (Government
#   Corruption)` <dbl>, Generosity <dbl>, `Dystopia Residual` <dbl>
```

```
#Discover the first 10 rows of your df
head(happiness_2016, 10)
# A tibble: 10 x 13
   Country Region `Happiness Rank` `Happiness Scor~ `Lower Confiden~
   <chr>   <chr>             <dbl>            <dbl>            <dbl>
 1 Denmark Weste~                1             7.53             7.46
 2 Switze~ Weste~                2             7.51             7.43
 3 Iceland Weste~                3             7.50             7.33
 4 Norway  Weste~                4             7.50             7.42
 5 Finland Weste~                5             7.41             7.35
 6 Canada  North~                6             7.40             7.34
 7 Nether~ Weste~                7             7.34             7.28
 8 New Ze~ Austr~                8             7.33             7.26
 9 Austra~ Austr~                9             7.31             7.24
10 Sweden  Weste~               10             7.29             7.23
```

```
#Discover the bottom 6 rows of your df
tail(happiness_2016)
# A tibble: 6 x 13
  Country Region `Happiness Rank` `Happiness Scor~ `Lower Confiden~
  <chr>   <chr>             <dbl>            <dbl>            <dbl>
1 Rwanda  Sub-S~              152             3.52             3.44
2 Benin   Sub-S~              153             3.48             3.40
3 Afghan~ South~              154             3.36             3.29
4 Togo    Sub-S~              155             3.30             3.19
5 Syria   Middl~              156             3.07             2.94
6 Burundi Sub-S~              157             2.90             2.73
# ... with 8 more variables: `Upper Confidence Interval` <dbl>,
#   `Economy (GDP per Capita)` <dbl>, Family <dbl>, `Health (Life
#   Expectancy)` <dbl>, Freedom <dbl>, `Trust (Government
#   Corruption)` <dbl>, Generosity <dbl>, `Dystopia Residual` <dbl>
```

```
# View a random sample of 10 row of the data
# use the sample function and nrow (number of rows)
happiness_2016[sample(nrow(happiness_2016),10),]
A tibble: 10 x 13
   Country Region `Happiness Rank` `Happiness Scor~ `Lower Confiden~
   <chr>   <chr>             <dbl>            <dbl>            <dbl>
 1 Czech ~ Centr~               27             6.60             6.52
 2 Mexico  Latin~               21             6.78             6.68
 3 Camero~ Sub-S~              114             4.51             4.42
 4 Malawi  Sub-S~              132             4.16             4.04
 5 Iraq    Middl~              112             4.58             4.45
 6 Malays~ South~               47             6.00             5.92
 7 Senegal Sub-S~              128             4.22             4.15
 8 Yemen   Middl~              147             3.72             3.62
 9 Kuwait  Middl~               41             6.24             6.15
10 India   South~              118             4.40             4.35
# ... with 8 more variables: `Upper Confidence Interval` <dbl>,
#   `Economy (GDP per Capita)` <dbl>, Family <dbl>, `Health (Life
#   Expectancy)` <dbl>, Freedom <dbl>, `Trust (Government
#   Corruption)` <dbl>, Generosity <dbl>, `Dystopia Residual` <dbl>
```

3. Explore the dataset using the `str()` function, to see the structure of your data. This gives you each variable's name and class and a sample of the records.

```
#Look at the structure of your df
> str(happiness_2016)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame': 157 obs. of  13 variables:
 $ Country                      : chr  "Denmark" "Switzerland" "Iceland" "Norway" ...
 $ Region                       : chr  "Western Europe" "Western Europe" "Western Europe" "Western Europe" ...
 $ Happiness Rank               : num  1 2 3 4 5 6 7 8 9 10 ...
 $ Happiness Score              : num  7.53 7.51 7.5 7.5 7.41 ...
 $ Lower Confidence Interval    : num  7.46 7.43 7.33 7.42 7.35 ...
 $ Upper Confidence Interval    : num  7.59 7.59 7.67 7.58 7.47 ...
 $ Economy (GDP per Capita)     : num  1.44 1.53 1.43 1.58 1.41 ...
 $ Family                       : num  1.16 1.15 1.18 1.13 1.13 ...
 $ Health (Life Expectancy)     : num  0.795 0.863 0.867 0.796 0.811 ...
 $ Freedom                      : num  0.579 0.586 0.566 0.596 0.571 ...
 $ Trust (Government Corruption): num  0.445 0.412 0.15 0.358 0.41 ...
 $ Generosity                   : num  0.362 0.281 0.477 0.379 0.255 ...
 $ Dystopia Residual            : num  2.74 2.69 2.83 2.66 2.83 ...
>
```

4. You can extract the dimensions of the table with the `dim()` function. This shows us that the data has 157 rows and 13 columns.

```
#Extract the dimensions
> dim(happiness_2016)
[1] 157  13
```

5. Look at the names of the variables.

```
> #Want to know all the variables and their names?
> colnames(happiness_2016)
 [1] "Country"
 [2] "Region"
 [3] "Happiness Rank"
 [4] "Happiness Score"
 [5] "Lower Confidence Interval"
 [6] "Upper Confidence Interval"
 [7] "Economy (GDP per Capita)"
 [8] "Family"
 [9] "Health (Life Expectancy)"
[10] "Freedom"
[11] "Trust (Government Corruption)"
[12] "Generosity"
[13] "Dystopia Residual"
```

6. Now let's **summarize** the data and discover **basic descriptive statistics** with the function `summary()`. This might not be the best output when you are dealing with a large number of variables. The statistics are presented for each variable in the order of appearance in the dataset and depending on the class of the variable. For example, for each numeric variable, it displays `min`, `1st quartile`, `median`, `mean`, `3rd quartile` and `max`.

```
summary(happiness_2016)
   Country             Region          Happiness Rank  
 Length:157         Length:157         Min.   :  1.00  
 Class :character   Class :character   1st Qu.: 40.00  
 Mode  :character   Mode  :character   Median : 79.00  
                                       Mean   : 78.98  
                                       3rd Qu.:118.00  
                                       Max.   :157.00  
 Happiness Score Lower Confidence Interval
 Min.   :2.905   Min.   :2.732            
 1st Qu.:4.404   1st Qu.:4.327            
 Median :5.314   Median :5.237            
 Mean   :5.382   Mean   :5.282            
 3rd Qu.:6.269   3rd Qu.:6.154            
 Max.   :7.526   Max.   :7.460            
 Upper Confidence Interval Economy (GDP per Capita)
 Min.   :3.078             Min.   :0.0000          
 1st Qu.:4.465             1st Qu.:0.6702          
 Median :5.419             Median :1.0278          
 Mean   :5.482             Mean   :0.9539          
 3rd Qu.:6.434             3rd Qu.:1.2796          
 Max.   :7.669             Max.   :1.8243          
     Family       Health (Life Expectancy)    Freedom      
 Min.   :0.0000   Min.   :0.0000           Min.   :0.0000  
 1st Qu.:0.6418   1st Qu.:0.3829           1st Qu.:0.2575  
 Median :0.8414   Median :0.5966           Median :0.3975  
 Mean   :0.7936   Mean   :0.5576           Mean   :0.3710  
 3rd Qu.:1.0215   3rd Qu.:0.7299           3rd Qu.:0.4845  
 Max.   :1.1833   Max.   :0.9528           Max.   :0.6085  
 Trust (Government Corruption)   Generosity    
 Min.   :0.00000               Min.   :0.0000  
 1st Qu.:0.06126               1st Qu.:0.1546  
 Median :0.10547               Median :0.2225  
 Mean   :0.13762               Mean   :0.2426  
 3rd Qu.:0.17554               3rd Qu.:0.3119  
 Max.   :0.50521               Max.   :0.8197  
 Dystopia Residual
 Min.   :0.8179   
 1st Qu.:2.0317   
 Median :2.2907   
 Mean   :2.3258   
 3rd Qu.:2.6646   
 Max.   :3.8377   
```

7. What about missing values? Try another display of the summary of the dataset with the `skim()` function. First, install and load the package `skimr`. It is a good addition to `summary()` that I find more user friendly as the variables are displayed by type. In addition to the mean, it also displays, sd, more quantile information, missing values and an inline histogram for each variable.

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

> #UserFriendly summary and display of missing values
> skim(happiness_2016)
-- Data Summary ------------------------
                           Values        
Name                       happiness_2016
Number of rows             157           
Number of columns          13            
```

```
────────────────────────
Column type frequency:
  character                2
  numeric                  11
────────────────────────
Group variables            None

-- Variable type: character -----------------------------------
# A tibble: 2 x 8
  skim_variable n_missing complete_rate   min   max empty
* <chr>             <int>         <dbl> <int> <int> <int>
1 Country               0             1     4    23     0
2 Region                0             1    12    31     0
  n_unique whitespace
*    <int>      <int>
1      157          0
2       10          0

-- Variable type: numeric -------------------------------------
# A tibble: 11 x 11
   skim_variable                 n_missing complete_rate   mean
 * <chr>                             <int>         <dbl>  <dbl>
 1 Happiness Rank                        0             1 79.0
 2 Happiness Score                       0             1  5.38
 3 Lower Confidence Interval             0             1  5.28
 4 Upper Confidence Interval             0             1  5.48
 5 Economy (GDP per Capita)              0             1  0.954
 6 Family                                0             1  0.794
 7 Health (Life Expectancy)              0             1  0.558
 8 Freedom                               0             1  0.371
 9 Trust (Government Corruption)         0             1  0.138
10 Generosity                            0             1  0.243
11 Dystopia Residual                     0             1  2.33
       sd     p0     p25    p50     p75    p100 hist
 *  <dbl> <dbl>   <dbl>  <dbl>   <dbl>   <dbl> <chr>
 1 45.5   1      40      79     118     157    ▇▇▇▇▇
 2  1.14  2.90    4.40    5.31    6.27    7.53 ▂▆▇▇▅
 3  1.15  2.73    4.33    5.24    6.15    7.46 ▂▆▇▇▅
 4  1.14  3.08    4.46    5.42    6.43    7.67 ▃▇▇▇▅
 5  0.413 0       0.670   1.03    1.28    1.82 ▃▃▇▇▂
 6  0.267 0       0.642   0.841   1.02    1.18 ▁▂▅▇▇
 7  0.229 0       0.383   0.597   0.730   0.953 ▂▃▅▇▅
 8  0.146 0       0.257   0.397   0.485   0.608 ▁▃▅▇▆
 9  0.111 0       0.0613  0.105   0.176   0.505 ▇▆▂▁▁
10  0.134 0       0.155   0.222   0.312   0.820 ▆▇▂▁▁
11  0.542 0.818   2.03    2.29    2.66    3.84 ▁▃▇▃▁
```

8. To check the missing data, you can use the function `is.na()` or `complete.cases()`, which evaluates to true if the row has no missing data. You can explore the missing data in the whole dataset or report it by case/observation (row) or by variable (column). You can use the logical `NOT` operator: `!` to explore the cases that are incomplete (i.e. `NOT` complete).

There are different ways of dealing with missing data, but you should always consider carefully whether you remove them or replace them, for example, `na.omit()` remove all the incomplete cases/observation.

```
> #Finding missing values in your df

# in the whole df
> sum(is.na(happiness_2016)) # use sum to count the number
[1] 0

# by column
> colSums(is.na(happiness_2016))
                      Country                        Region
                            0                             0
               Happiness Rank               Happiness Score
                            0                             0
    Lower Confidence Interval     Upper Confidence Interval
```

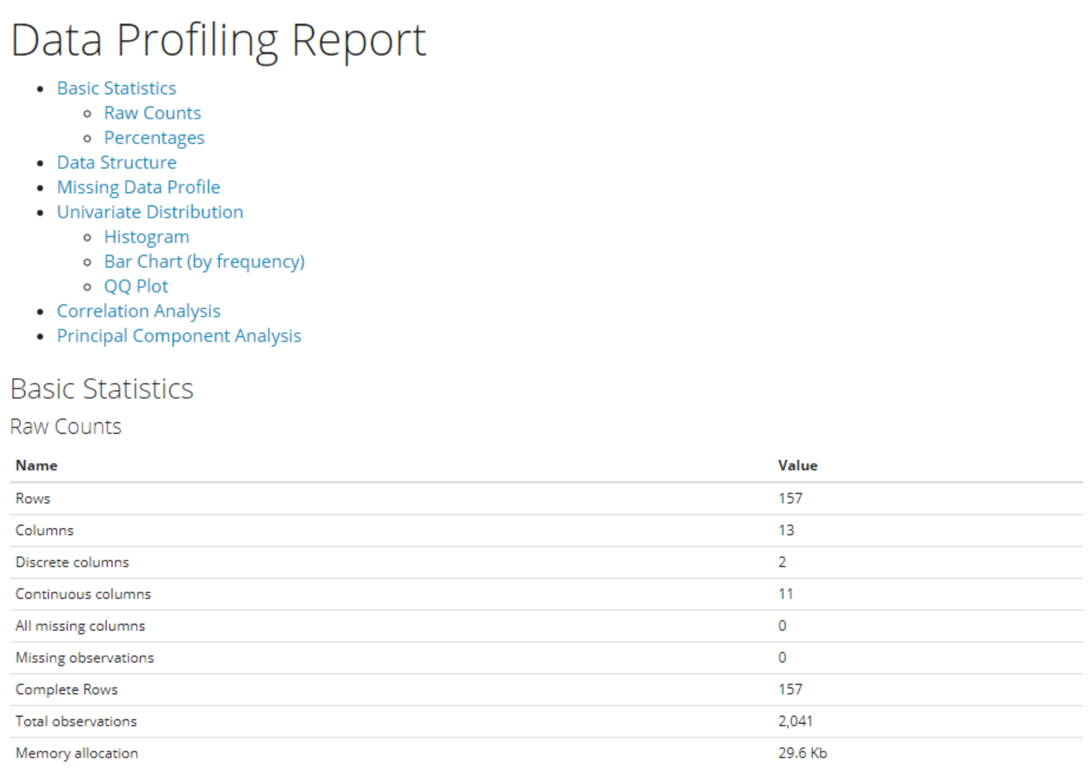
```
                            0                              0
       Economy (GDP per Capita)                       Family
                            0                              0
       Health (Life Expectancy)                      Freedom
                            0                              0
  Trust (Government Corruption)                   Generosity
                            0                              0
              Dystopia Residual
                            0
   # by rows
  > happiness_2016[!complete.cases(happiness_2016),] # returns all NON complete rows because we use !
  # A tibble: 0 x 13
  # ... with 13 variables: Country <chr>, Region <chr>,
  #   `Happiness Rank` <dbl>, `Happiness Score` <dbl>, `Lower
  #   Confidence Interval` <dbl>, `Upper Confidence
  #   Interval` <dbl>, `Economy (GDP per Capita)` <dbl>,
  #   Family <dbl>, `Health (Life Expectancy)` <dbl>,
  #   Freedom <dbl>, `Trust (Government Corruption)` <dbl>,
  #   Generosity <dbl>, `Dystopia Residual` <dbl>
```

9. Use `table()` to build a contingency table of the counts of observation in each factor. In this example, how many countries are there in each region?

```
> table(happiness_2016$Region)

      Australia and New Zealand     Central and Eastern Europe
                              2                             29
                   Eastern Asia    Latin America and Caribbean
                              6                             24
Middle East and Northern Africa                  North America
                             19                              2
             Southeastern Asia                  Southern Asia
                              9                              7
             Sub-Saharan Africa                 Western Europe
                             38                             21
```

10. Finally, you can produce a ready-made report as an html file with the basic statistics, structure, missing data, distribution visualisations, correlation matrix and principal component analysis for your data frame! And all of this with just one line of code `create_report(df)` from the package `DataExplorer`. This can be really handy when you need a quick wrap-up of your dataset, but it is also important to know how to access each of the statistics, as explained above.

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

#Create a report
create_report(happiness_2016)
```



Data Profiling Report

- Basic Statistics
  - Raw Counts
  - Percentages
- Data Structure
- Missing Data Profile
- Univariate Distribution
  - Histogram
  - Bar Chart (by frequency)
  - QQ Plot
- Correlation Analysis
- Principal Component Analysis

Basic Statistics

Raw Counts

Name	Value
Rows	157
Columns	13
Discrete columns	2
Continuous columns	11
All missing columns	0
Missing observations	0
Complete Rows	157
Total observations	2,041
Memory allocation	29.6 Kb

Data visualisation

Visualising your data helps you to understand and summarise the data. This is primordial for exploring data and raising questions.

- **R base graphs**

R comes with ready made function to create simple graphs such as

```
plot() #scatter plot
boxplot() # boxplot
hist() # histogram
barplot() # bar plot
line() # line plot
```

Before producing any figures, let's fix the issues we noticed during the data exploration, check that **variable names** have no spaces and check that the **variable type** is correct.

```
#rename variables to remove the space (you can also use mutate function from dplyr, will come later)
names(happiness_2016)[names(happiness_2016)=="Happiness Score"] <- "Happiness_Score"
names(happiness_2016)[names(happiness_2016)=="Happiness Rank"] <- "Happiness_Rank"

#transform region into a factor
happiness_2016$Region<-as.factor(happiness_2016$Region)
```

Exercise 1

Is there any patterns in the score of Happiness (or is the score normally distributed)? Visualise the variation in the score of happiness between the different countries.

Hint: A histogram is a good way to visualise a continuous variable.

Exercise 2

Which region has the highest Happiness score?

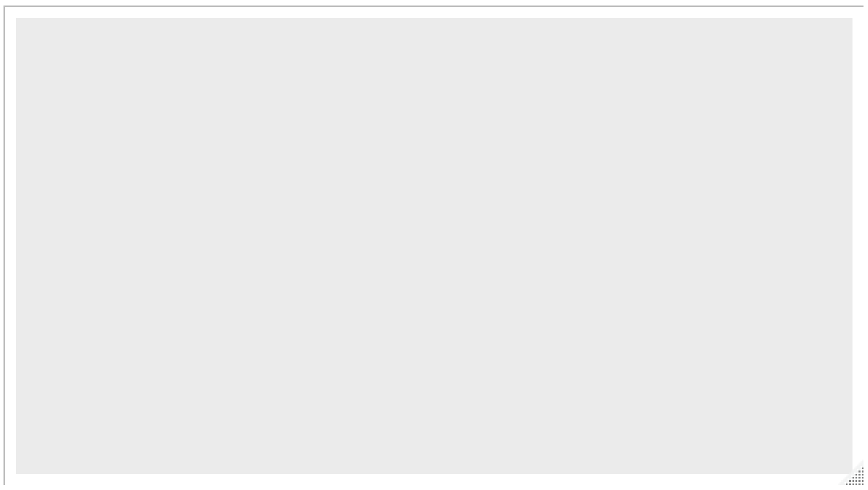
Hint: Use boxplots to visualise the variation by region.

- **GGPLOT2 graphs**

The best way to build elegant graphs is to use the ggplot2 package. This requires a bit of understanding of the grammar of graphics. Here is how to build a simple and elegant graph step by step like a construction game.

1. Create an empty graph by calling the function `ggplot()`.

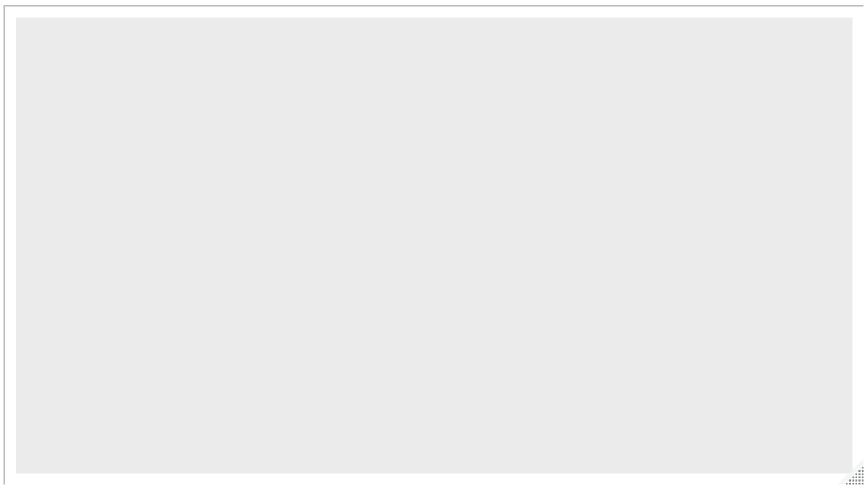
```
ggplot()
```



This produces an empty space

2. Specify the data you want to use by setting the first argument `data`.

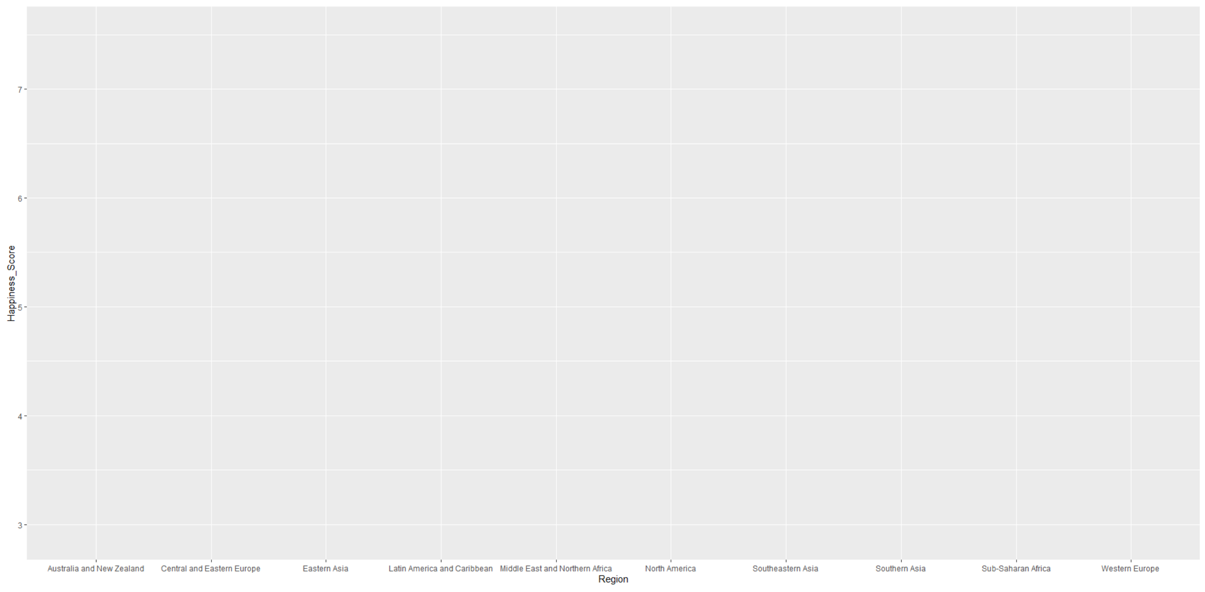
```
ggplot(data=happiness_2016))
```



This still gives us an empty space, so let's add the variables!

3. Add the variables you want to plot in x and y using the aes() "aesthetic" argument, which maps the variable into a plan.

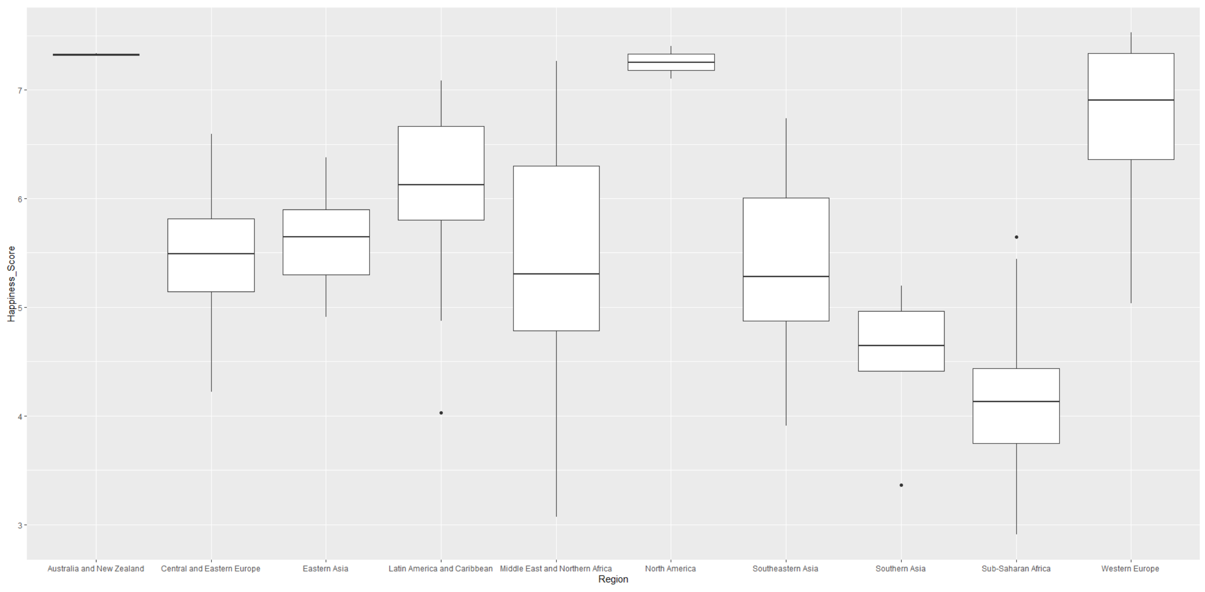
```
ggplot(data=happiness_2016, aes(x = Region, y = Happiness_Score))
```



This creates gridlines and axis labels but no data as we did not tell R how to plot it. You can check we have the 10 regions and the range for the Happiness score.

4. Add boxplots to the graph by using `geom_boxplot()`

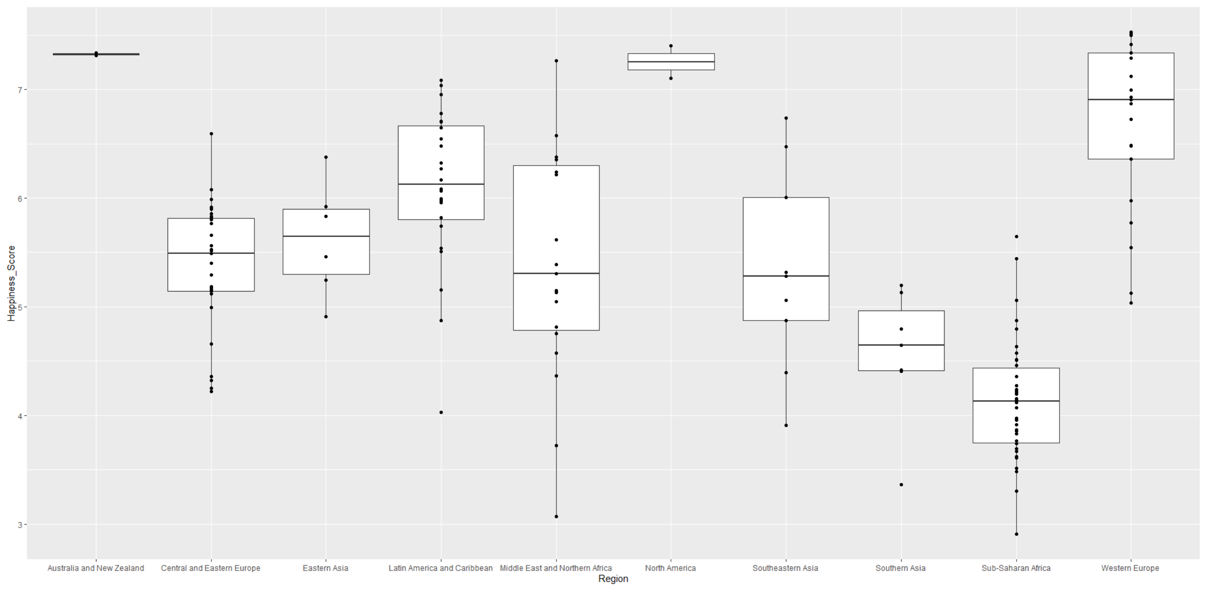
```
ggplot(data=happiness_2016, aes(x = Region, y = Happiness_Score))+
  geom_boxplot()
```



The problem here is that each region does not include the same number of countries. We need to take that into account to avoid misleading conclusions.

5. Add points for each of the countries in each region. To do this, we need to tell R to add another layer of data. Specifically, the country values (our observations or id) and to represent them by points, using `geom_point`.

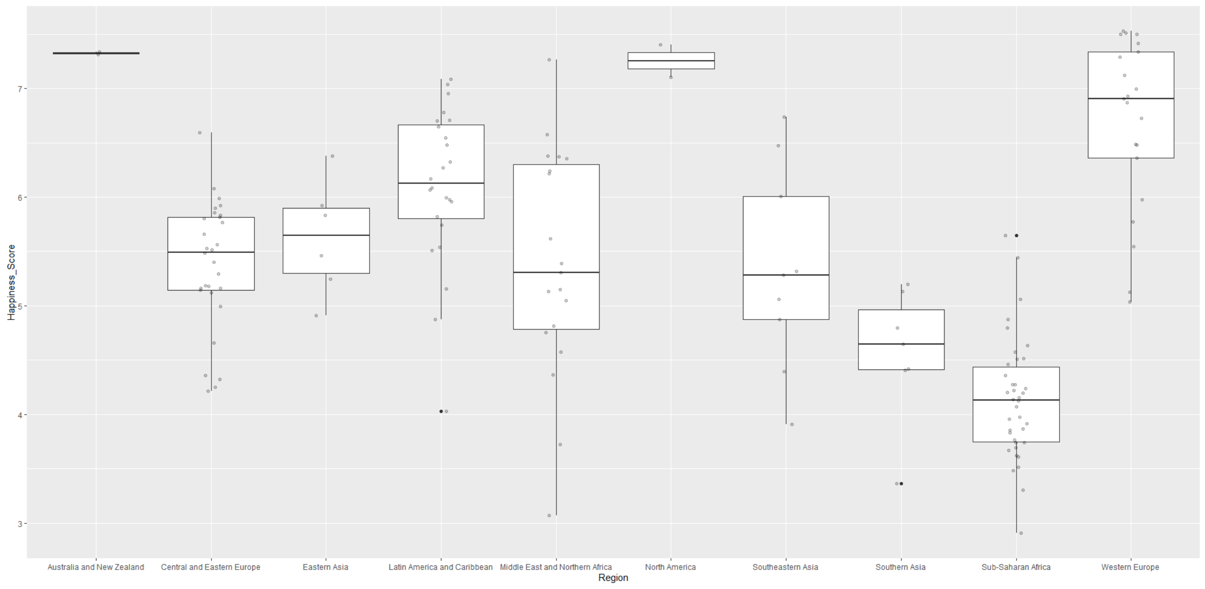
```
ggplot(data=happiness_2016, aes(x = Region, y = Happiness_Score))+
  geom_boxplot()+
  geom_point()
```



It is showing us that some regions are represented by more countries than others, but it is not clear as some points are overlapping.

6. To avoid overlap, use jitter to see more clearly all the countries per region. The argument is the function `geom_jitter` allows us to adjust the width of the points and their transparency (`alpha` close to 0 gives transparent points, whereas the transparency is none when `alpha`=1)

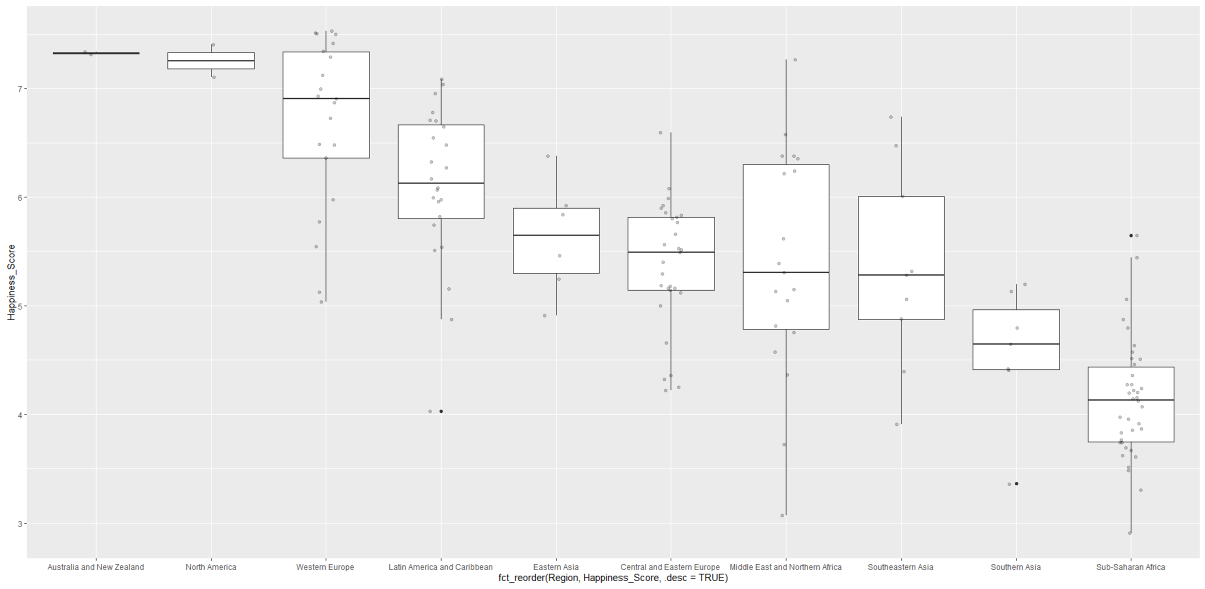
```
ggplot(data=happiness_2016, aes(x = Region, y = Happiness_Score))+
  geom_boxplot()+
  geom_jitter(width=0.1,alpha=0.2)
```



This is better, now it will be good to see directly which region have the greatest happiness score.

7. Reorder the region by descending score of Happiness. To do that, quickly use the function `fct_reorder` () from the package `forcats` (for categorical variables) directly in the plot. We need to add `.desc=TRUE` to ensure the data are ranked from the highest to the lowest score.

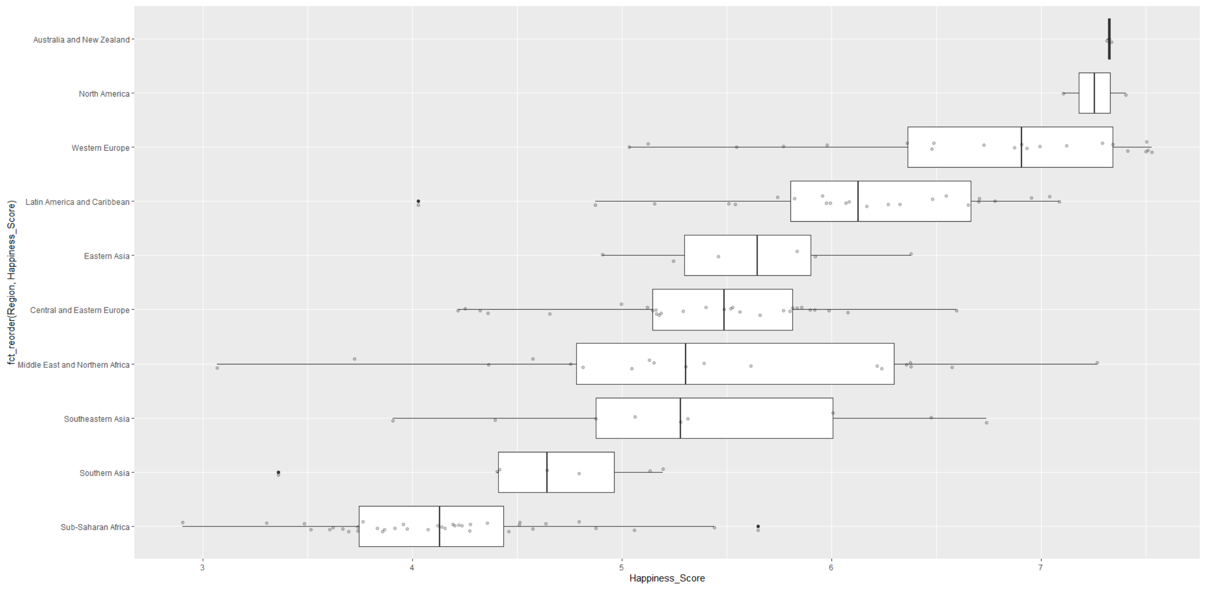
```
install.packages("forcats")
library(forcats)

ggplot(data=happiness_2016, aes(x = fct_reorder(Region, Happiness_Score , .desc=TRUE),
      y = Happiness_Score))+
  geom_boxplot()+
  geom_jitter(width=0.1,alpha=0.2)
```



This is better, but the regions are not easy to read. When dealing with lots of factor variables, a nice trick is to flip the axis to display the factors vertically.

8. Let's flip the coordinates using `coord_flip()`. We also need to remove `.desc=TRUE` to get the highest rank on top.

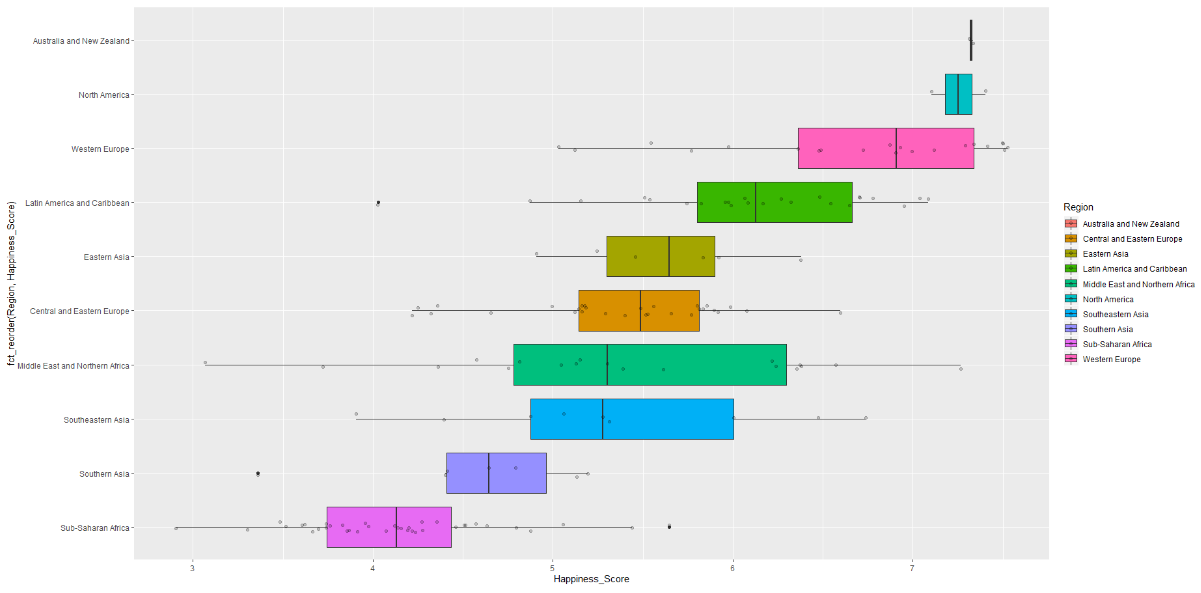
```
ggplot(data=happiness_2016, aes(x = fct_reorder(Region, Happiness_Score ),
      y = Happiness_Score))+
  geom_boxplot()+
  geom_jitter(width=0.1,alpha=0.2)+
  coord_flip()
```



This graph has all it needs to explain the data, but we can now make it more readable. Let's talk about design!

9. Adding colours will help to differentiate the regions. To colour by region we need to tell R in the aesthetics `aes()` to fill the boxplot by regions.

```
ggplot(data=happiness_2016, aes(x = fct_reorder(Region, Happiness_Score ),
       y = Happiness_Score, fill = Region))+
  geom_boxplot()+
  geom_jitter(width=0.1,alpha=0.2)+
  coord_flip()
```

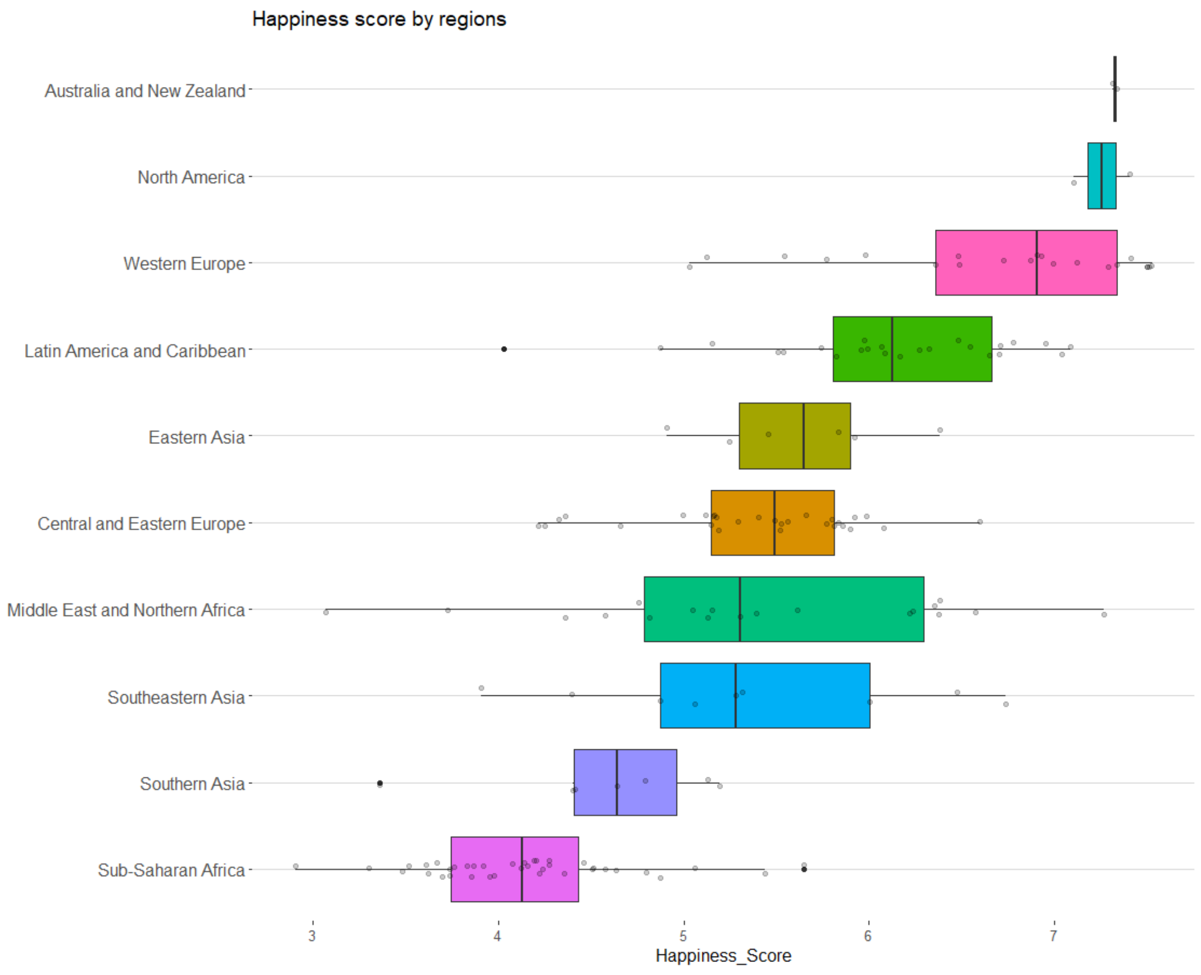


This is getting better, now let's improve the legend, add a title and remove the gridlines.

10. Let's polish the graph by

- removing unnecessary components (regions legends `theme(legend.position = "none")`, y-axis title `theme(axis.title.y = element_blank())`).
- adding a title: simply use `ggtitle`("Happiness score by regions").
- finetuning the graph: we can use a theme from the `ggthemes` package called `theme_hc()`.
- increasing the size of the text: `theme(axis.text.y=element_text(size=12)`

```
ggplot(data=happiness_2016, aes(x = fct_reorder(Region, Happiness_Score ), y = Happiness_Score, fill = Region))+
  geom_boxplot()+
  geom_jitter(width=0.1,alpha=0.2)+
  coord_flip()+
  ggtitle("Happiness score by regions")+
  theme_hc()+
  theme(legend.position = "none")+
  theme(axis.text.y = element_text(size = 12))+
  theme(axis.title.x = element_text(size = 12))+
  theme(axis.title.y = element_blank())
```



This is clear and eye catching! Good Job!

Summary

You learned:

- How to preview your dataset
- How to create descriptive statistics
- How to explore the structure of your dataset and summarise it
- How to make quick graph with base plot
- How to create a ggplot step by step

Solutions

```
#Solution exercise 1

#rename hapiness score to remove the space
names(happiness_2016)[names(happiness_2016)=="Happiness Score"] <- "Happiness_Score"
names(happiness_2016)[names(happiness_2016)=="Happiness Rank"] <- "Happiness_Rank"

#transform region into a factor
happiness_2016$Region<-as.factor(happiness_2016$Region)
happiness_2016$Country<-as.factor(happiness_2016$Country)

hist(happiness_2016$Happiness_Score)
```

```
#Solution exercise 2
boxplot(Happiness_Score ~ Region, data=happiness_2016)
```

Additional exercises

Use the code from step 10 and make some small tweaks to get

1 - The density for each region instead of a boxplot

2- Change the fill to black and weight

Materials used in this course can be cloned directly by clicking here. Alternatively, you can view the repository online:

> *https://github.com/RJODRISCOLL/Introduction-to-R*

For any help and advice, We can be contacted at:

> pspjo@leeds.ac.uk, *R.ODriscoll@leeds.ac.uk*