Lab Exercise-3

This lab introduces the basic understating of data transformation and Graph plotting.

A. Transformation of variables, subsets of datasets: Consider the dataset cherry

1. *(Transformation of variables in a dataset)* Construct a new dataset, `cherry1`, with the command

```
cherry1 <- transform(cherry, logVolume=log(Volume), logGirth=log(Girth))
```

and notice that the name appears in the upper right box of RStudio. Inspect the new dataset, either by clicking on the name in the upper right box, or by one of the commands

```
cherry1
head(cherry1)
hist(cherry1$logVolume)
```

Notice that `log` is the natural logarithm. If you prefer the logarithm with base 2 or 10, you should use the functions `log2` or `log10` instead og `log`.

2. *(Subset of a dataset)* Try the following commands (one at a time). Make sure you understand the output.

```
cherry[3,]
cherry[3:5,]
cherry[-c(2,4),]
```

Construct and inspect each of the following datasets (one at at time). Make sure you understand the content of each dataset.

```
subset(cherry, Height>70)
subset(cherry, Height>=70)
subset(cherry, Height==80)
subset(cherry, Height==80, select=c(Girth,Volume))
subset(cherry, Height>80 & Girth>15)
subset(cherry, Height>80 | Girth>15)
```

B. Merging datasets

It sometimes happens that data comes from different sources (files), and should be merged before analysis. Consider the dataset cherry

1. *(Merging data, new datalines)* Assume that data from two more trees are made available. Try the following commands that construct a dataset with the new data and merge the two datasets:

```
newData <- data.frame(Girth=c(11.5, 17.0), Height=c(71, 75), Volume=c(22, 40))
newData
allData <- rbind(cherry, newData)
allData
```

2. *(Merging data, new variables)* Assume that the precipitation at the location of each tree has also been registered and saved in a variable called `precipitation`. The variable below contains random numbers with mean 50 and standard deviation 10, and are meaningless — yet the commands illustrate the merging of a new variable and an existing dataset.

```
precipitation <- rnorm(n=31, mean=50, sd=10)
precipitation
allData2 <- cbind(cherry, precipitation)
allData2
```

   Of course the variables should be ordered the same way in the dataset and the new variable. See (Martinussen *et al.*, 2012, Section 4.2) for more advanced merging using the `merge` function.

C. Working with data: Tomato yields?

Consider an experiment with tomatoes. Three different varieties and four different seed densities have been tested, and there are three replications for each of the 12 combinations. The yield has been registered for each of the 36 field plots. The data are saved in the file `tomatoes.xlsx`

1. Create a dataset called `tomatoes`, say, in R, and inspect the data with the functions `plot` and `summary`. Which variables are quantitative and which variables are categorical?
2. Make a histogram of the yield variable. Moreover, compute the mean, median and standard deviation of the yield variable.
3. Try the following commands, and explain the output:

```
table(variety)
table(density)
table(density, variety)
```

4. Make a new dataset with two more variables: one with the squareroot of the yield values and one with the logarithm of the yield values.

5. Make a dataset which only contains the datalines corresponding to the variety Ife (use `variety=='Ife'` in a `subset` command).

6. Make a dataset containing datalines for the variety Pusa with density less than 25000. What is the median of the corresponding yield values?

7. Make a dataset without observation numbers 5 and 24. Make a histogram of the yield values from this dataset.

D. Graphs- Basic high-level plots

This exercise introduces some basic high-level plots, which can be produced with very simple Commands. Here we use the default options (which are often fine), but it is an important point that graphs can easily be modified.
Consider first the dataset cherry from Lab-Exercise-1, and recall that it has variables Girth, Height and Volume.

1. *(Scatterplots)* The most important graphics function `plot` function. It does different things depending on the type of argument(s) supplied to the function.

   Try the commands and explain what you see:

   ```
   plot(cherry)
   plot(Girth, Volume)
   plot(log(Girth), log(Volume))
   plot(Height)
   ```

2. *(Histograms and boxplots)* Try the commands

   ```
   hist(Volume)
   boxplot(Volume)
   ```

3. *(Barplots)* The temperature in New York was measured daily for five month (May to September). The average temperatures in degrees Fahrenheit were 65.5, 79.1, 83.9, 84.0, and 76.9, respectively. Try the commands

   ```
   temp <- c(65.5, 79.1, 83.9, 84.0, 76.9)
   barplot(temp, names=5:9)
   ```

E. Modifications of scatter plots

The temperature in New York was measured daily for five month (May to September). The data from July and August are saved in the files `ny-temp.xlsx` and `ny-temp.csv`. There are three variable: `Month` with values 7 and 8, `Day` with values 1–31, and `Temp` with the temperature in degrees Fahrenheit.

1. Read the data into a dataset, `NYtemp`. Then make two sub-dataset, `july` and `august` with the data from July and August, respectively.

2. *(Plotting symbols, lines, axes, title)* Try the following commands and explain what happens:

```
attach(july)
plot(Day, Temp)
plot(Day, Temp, pch=16)
plot(Day, Temp, pch=2)
plot(Day, Temp, type="l")
plot(Day, Temp, type="b")
plot(Day, Temp, type="l", lty=2)
plot(Day, Temp, type="l", lwd=2)
plot(Day, Temp, type="l", col="blue")
plot(Day, Temp, type="l", xlab="Day in month", ylab="Temperature (F)",
     main="Temperatur in New York")
plot(Day, Temp, type="l", cex.lab=1.3)
plot(Day, Temp, type="l", cex.axis=1.3)
plot(Day, Temp, type="l", xlim=c(0,40), ylim=c(50,100))
detach(july)
```

3. *(Adding data points)* Try the following commands and explain what you see:

```
range(july$Temp)
range(august$Temp)
plot(july$Day, july$Temp, type="l", ylim=c(70,100))
points(august$Day, august$Temp, col="red")
plot(july$Day, july$Temp, type="l", ylim=c(70,120))
lines(august$Day, august$Temp, col="green")
```

4. *(Legends)* Add a legend to the previous plot:

```
legend(5,120, c("July", "August"), text.col=c("Black","Green"))
```

5. *(Adding straight lines)* Try the commands:

```
plot(july$Day, july$Temp, type="l", ylim=c(70,100))
abline(h=73, col="red")
abline(v=14, col="blue")
abline(84.652, -0.0468)
```

Notice how the final command adds a line to the plot with intercept and slope as specified in the command. The actual values are the estimates from a linear regression

6. *(Points coloured according to third variable)* In the R package `MASS` there is a dataset called `cats` with variables `Sex`, `Bwt` (body weight in kg), `Hwt` (in g). Run the following commands:

```
library(MASS)
data(cats)
plot(cats)
```

Then try the command and explain what you see:

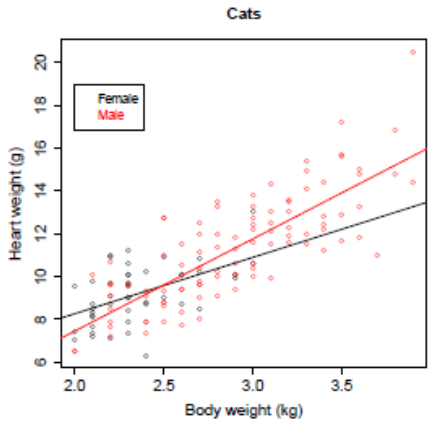

Figure 1: Scatterplot with regression lines for male and female cats.

```
plot(cats$Bwt, cats$Hwt, col=cats$Sex)
```

7. Modify the figure such that it looks similar to Figure 1. The lines are regression lines for linear regression models fitted to male and female cats, respectively, namely

$$\text{Male cats:} \quad \texttt{HWt} = -1.184 + 4.313 \cdot \texttt{Bwt}$$
$$\text{Female cats:} \quad \texttt{HWt} = 2.981 + 2.636 \cdot \texttt{Bwt}$$

F. Modifications of histograms, parallel boxplot

1. In the R package `MASS` there is a dataset called `cats`. Run the following commands:

```
library(MASS)
data(cats)
plot(cats)
```

The variables `Bwt` and `Hwt` contain the weight of the body (kg) and the heart (g), respectively. There are both male and female cats.

2. *(Modification of histograms)* Try the following commands and explain what happens:

```
attach(cats)
hist(Hwt[Sex=="M"])
hist(Hwt[Sex=="M"], prob=T)
hist(Hwt[Sex=="M"], breaks=c(5,10,15,20,25))  ## Not nice
```

Use the `xlab` and `main` arguments to change the x-label and the title to something more appropriate.

3. *(Parallell boxplots)* Try the following commands and explain what happens:

```
boxplot(Hwt[Sex=="F"])
boxplot(Hwt[Sex=="M"])
boxplot(Hwt ~ Sex)
```

G. Exporting graphics

Use the temperature data

There are (at least) two different ways to save your plots as files as `pdf` files that can be used for other documents. The plots can also be saved in other formats, and it is possible to set the size (width and height) of the graphs if the default is not satisfactory.

1. *(Save an existing plot)* Try the following commands:

```
plot(july$Day, july$Temp)
dev.print(file="plotfile1.pdf", device=pdf)
```

   The file `plotfile1.pdf` should now appear in the working directory. You can get information about the current working directory with the command `getwd()`. Find the file and open it.

2. *(Plot directly in a pdf file)* Try the following commands:

```
pdf("plotfile3.pdf")
plot(july$Day, july$Temp, col="blue")
dev.off()
```

   The commands open a file in the working directory, makes the graph in the file, and closes the file again. Find the file and open it.