Theme 9. Example in R

Exercise 1.

0. Define the type of the variables

Because the variables have infinite values spectra then they're the numeric type.

1. Prepare the data to get into R. Watch the video how to import data in Google Sheet - https://youtu.be/sy-j-sqd0t8

After the data has been successfully imported use commands as shown below:

```
Input = ("Glucose level    Body weight
             4       70
             6       72
             6       90
             2       80
             2       102
             4       92
             5       94
             4       81
             4       78
             6       84
             6       100
             7       90
             7       93
             3       83
             3       81
             3       72
             3       69
             2       72
             2       68
             2       120
             4       108
             3       97
             3       82
             2       93")
DF = as.data.frame(read.table(textConnection(Input), header = TRUE, sep =
"\t"))
```

This will allow you to work with data from the single variable of type data.frame. To access specific variable in the data.frame use the $ sign. For example, to get the Glucose level one have to use `DF$Glucose.level`

Alternatively, one can construct data frame from the column vectors by using commands:

```
glucose_lvl <- c(4,6,6,2,2,4,5,4,4,6,6,7,7,3,3,3,3,2,2,2,4,3,3,2)
body_weight <- c(70,72,90,80,102,92,94,81,78,84,100,90,93,83,81,72,69,72,68
,120,108,97,82,93)
DF =data.frame(Glucose.lvl = glucose_lvl, Body.weight = body_weight)
```

2. Calculate mean, median, mode

To get the **mean** one has to use command `mean`

```
mean(DF$Glucose.lvl)
```

Result:

```
> mean(DF$Glucose.lvl)
[1] 3.875
```

```
mean(DF$Body.weight)
```

Result:

```
> mean(DF$Body.weight)
[1] 86.29167
```

Median can be calculated by use of command `median`

```
median(DF$Glucose.lvl)
```

Result:

```
> median(DF$Glucose.lvl)
[1] 3.5
```

```
median(DF$Glucose.lvl)
```

Result:

```
> median(DF$Body.weight)
[1] 83.5
```

Mode can be obtained in two ways: through the `table` command or via the use of `Mode` command from the `DescTools` library

```
table(DF$Glucose.lvl)
```

Result:

```
> table(DF$Glucose.lvl)

2 3 4 5 6 7
6 6 5 1 4 2
```

As one can see there are two values that has the max frequency of 6 which means variable Glucose level has 2 modes: 2, 3.

```
library(DescTools)    # This command can be executed only once to get
                      # the needed functions
Mode(DF$Glucose.lvl)
```

Result:

```
> library(DescTools)
> Mode(DF$Glucose.lvl)
[1] 2 3
attr(,"freq")
[1] 6
```

As you can see we've got the same result with a different function. It's up to the reader (you) to obtain the mode(s) for the Body weight variable.

3. Calculate range, standard deviation, variance, standard error of mean, standard error of median

To get range one has to subtract the minimum value (`min`) of a variable from its maximum value (`max`).

```
max(DF$Glucose.lvl) - min(DF$Glucose.lvl)
```

Result:

```
> max(DF$Glucose.lvl) - min(DF$Glucose.lvl)
[1] 5
```

Alternatively range can be obtained via the `Range` function from the `DescTools` library.

```
Range(DF$Glucose.lvl)
```

Result:

```
> Range(DF$Glucose.lvl)
[1] 5
attr(,"bounds")
[1] 2 7
```

Standard deviation is being calculated through the `sd` function

```
sd(DF$Glucose.lvl)
```

Result:

```
> sd(DF$Glucose.lvl)
[1] 1.676241
```

```
sd(DF$Body.weight)
```

Result:

```
> sd(DF$Body.weight)
[1] 13.31387
```

Variance can be calculated through the `var` function

```
var(DF$Glucose.lvl)
```

Result:

```
> var(DF$Glucose.lvl)
[1] 2.809783
```

```
var(DF$Body.weight)
```

Result:

```
> var(DF$Body.weight)
[1] 177.2591
```

Standard error of mean can be obtained in two ways:

a. Manually by combining in expression the standard deviation (`sd`), square root (`sqrt`) and sample size (`length`) calculation functions

```
sd(DF$Glucose.lvl) / sqrt(length(DF$Glucose.lvl))
```

Result:

```
> sd(DF$Glucose.lvl) / sqrt(length(DF$Glucose.lvl))
[1] 0.3421612
```

b. Using function `MeanSE` from the `DescTools` library

```
DescTools::MeanSE(DF$Glucose.lvl)
```

Result:

```
> DescTools::MeanSE(DF$Glucose.lvl)
[1] 0.3421612
```

The way the `MeanSE` function being called in this example is just another way of telling R that you wish to use specific function from the specific library. But it's not mandatory and you could safely use the short form: `MeanSE(DF$Glucose.lvl)` and it would still provide you with the correct result.

```
> MeanSE(DF$Glucose.lvl)
[1] 0.3421612
```

4. Calculate confidence intervals of mean, confidence intervals of median

Confidence interval of mean can be gotten in two ways:

a. From one sample t-test

```
t.test(DF$Body.weight)$conf.int
```

Result:

```
> t.test(DF$Body.weight)$conf.int
[1] 80.66971 91.91362
attr(,"conf.level")
[1] 0.95
```

Lower (or left) bound of confidence interval of mean is the first value (80.67 in this example). Upper (or right) bound of confidence interval of mean is the second value (91.91 in this example). The confidence level is 95% (0.95).

b. From `MeanCI` function of `DescTools` library

```
MeanCI(DF$Body.weight)
```

```
> MeanCI(DF$Body.weight)
    mean   lwr.ci   upr.ci
86.29167 80.66971 91.91362
```

lwr.ci is a lower (or left) bound of the confidence interval of mean. upr.ci is an upper (or right) bound of the confidence interval of mean. The default confidence level for the MeanCI function is 0.95.

Confidence interval of median can be obtained by `MedianCI` function from `DescTools` library

```
MedianCI(DF$Glucose.lvl)
```

```
> MedianCI(DF$Glucose.lvl)
median lwr.ci upr.ci
   3.5    3.0    5.0
attr(,"conf.level")
[1] 0.9773442
```

lwr.ci is a lower (or left) bound of the confidence interval of median. upr.ci is an upper (or right) bound of the confidence interval of median. The default confidence level for the MedianCI function is 0.95.

5. Check if the samples follow normal distribution

To check if the sample follows the normal distribution one has to apply the Shapiro-Wilk test. It can be done via the `shapiro.test` function. If the sample follows normal distribution, then the Shapiro-Wilk test will give p-value greater than the significance level of 0.05. If the opposite is true, then the sample doesn't follow the normal distribution.

```
shapiro.test(DF$Glucose.lvl)
```

Result:

```
> shapiro.test(DF$Glucose.lvl)

        Shapiro-Wilk normality test

data:  DF$Glucose.lvl
W = 0.87708, p-value = 0.007263
```

As we can see Glucose level variable doesn't follow normal distribution because it's p-value is lower than the significance level of 0.05. We can confirm this by plotting the histogram of the Glucose level variable. To do so we have to use `hist` function. Result of the function is presented on figure 1.

```
hist(DF$Glucose.lvl, main = "Glucose level", xlab = "Value")
```

main = “Glucose level” – title of the histogram plot

xlab = “Value” – title of the horizontal axis

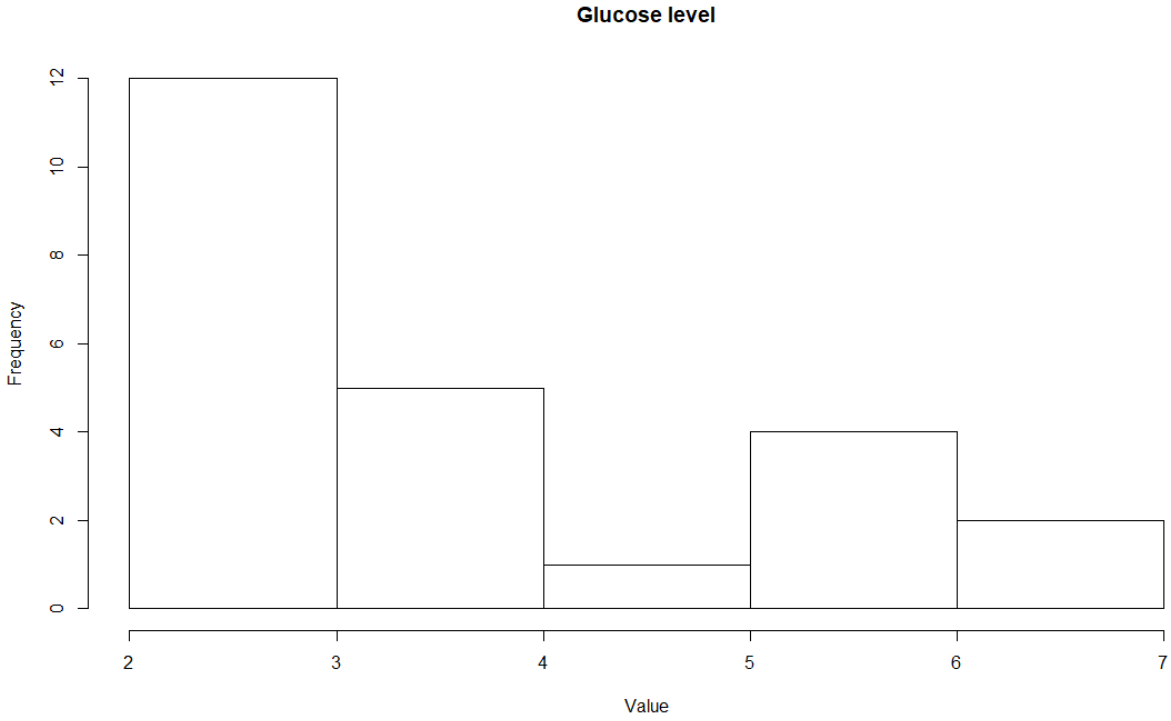

Fig.1. Histogram of the Glucose level variable. The variable frequencies are skewed towards left and hence evidence against variable being normally distributed.

For the Body weight we have the opposite case.

```
> shapiro.test(DF$Body.weight)

        Shapiro-Wilk normality test

data:  DF$Body.weight
W = 0.95007, p-value = 0.2719
```

As one can see the Body weight’s p-value exceeds the significance level of 0.05 which means Body weight variable follows the normal distribution. The histogram of the Body weight variable is plotted in figure 2.

```
hist(DF$Body.weight, main = "Body weight", xlab = "Value")
```

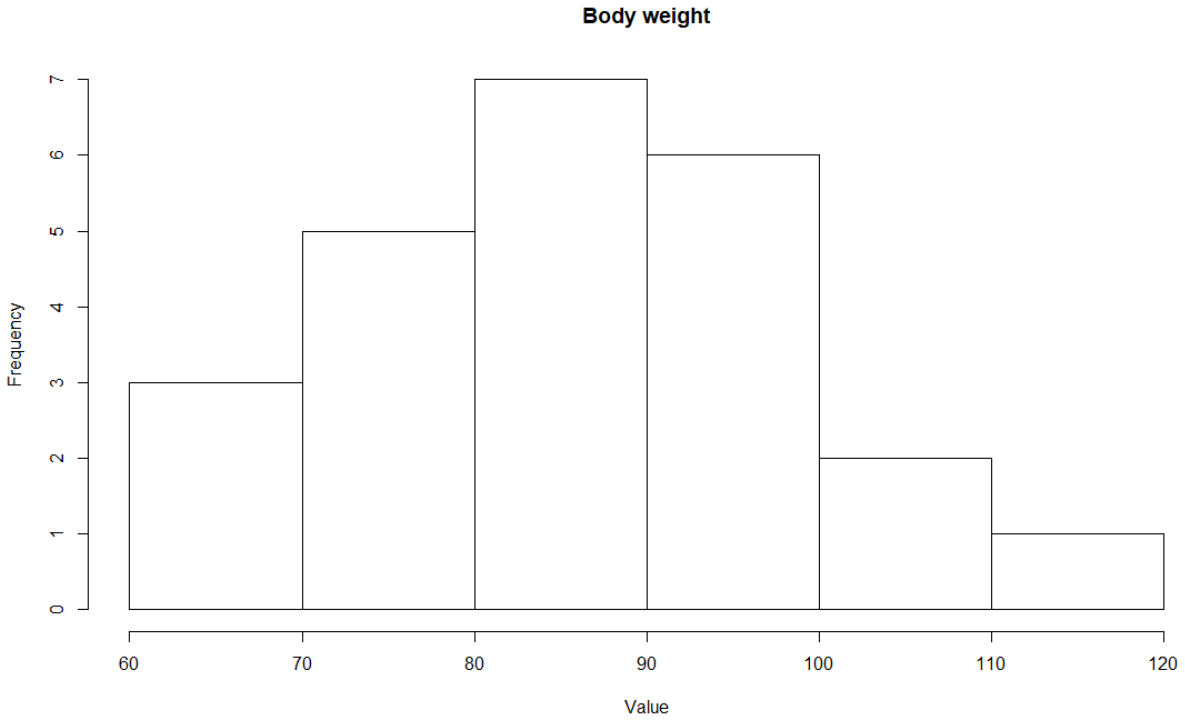

Fig.2. Histogram of Body weight variable. The variable frequencies are distributed in a bell-like pattern which is in agreement with the Shapiro-Wilk test results.

6. Gathering the results

We should garner the results in a table. To do so we could have stored the results in a separate variables (the preferred way) or we can produce an expression which combines all of the required functions.

R variable *gl_summary* to hold calculation results for the Glucose level:

```
gl_summary <- list(
Name = "Glucose level",
Size = length(DF$Glucose.lvl),
Min = min(DF$Glucose.lvl), Max = max(DF$Glucose.lvl),
Mean = mean(DF$Glucose.lvl),
Median = median(DF$Glucose.lvl),
SD = sd(DF$Glucose.lvl), Var = var(DF$Glucose.lvl),
SEM = DescTools::MeanSE(DF$Glucose.lvl),
lwr.CI = t.test(DF$Glucose.lvl)$conf.int[1],
upr.CI = t.test(DF$Glucose.lvl)$conf.int[2])
```

Variable *gl_summary* is a list of pairs *tag* = *value*. *Tag* serves only as a name for the *value* and doesn't aff ect calculations, but instead one can address the specific value by its tag.

R variable *bw_summary* to hold calculation results for the Body weight:

```
bw_summary <- list(
Name = "Body weight",
Size = length(DF$Body.weight),
Min = min(DF$Body.weight),
Max = max(DF$Body.weight),
Mean = mean(DF$Body.weight),
Median = median(DF$Body.weight),
SD = sd(DF$Body.weight), Var = var(DF$Body.weight),
SEM = DescTools::MeanSE(DF$Body.weight),
lwr.CI = t.test(DF$Body.weight)$conf.int[1],
upr.CI = t.test(DF$Body.weight)$conf.int[2]
)
```

Final data frame to hold the calculation results is done through the merge of the above variables. During the merge process both variable can be treated either as rows or columns. To merge the variables as rows one has to use `rbind` function. To merge the variables as columns (preferred method) one has to use `cbind` function.

```
rbind(gl_summary,bw_summary)
Result:
> rbind(gl_summary,bw_summary)
           Name             Size Min Max Mean     Median SD       Var      SEM
lwr.CI   upr.CI
gl_summary "Glucose level" 24   2   7   3.875    3.5    1.676241 2.809783 0.3
421612 3.167186 4.582814
bw_summary "Body weight"   24   68  120 86.29167 83.5   13.31387 177.2591 2.7
17682  80.66971 91.91362
```

```
cbind(gl_summary,bw_summary)
Result:
> cbind(gl_summary,bw_summary)
       gl_summary      bw_summary
Name   "Glucose level" "Body weight"
Size   24              24
Min    2               68
Max    7               120
Mean   3.875           86.29167
Median 3.5             83.5
SD     1.676241        13.31387
Var    2.809783        177.2591
SEM    0.3421612       2.717682
lwr.CI 3.167186        80.66971
upr.CI 4.582814        91.91362
```

Store your calculations as a data frame

```
DF_Summary = as.data.frame(cbind(gl_summary, bw_summary))
```

Now you can access the specific value by addressing it through the $ sign. For example, to get the upper CI for the Body weight one has to execute `DF_Summary$bw_summary$upr.CI`