ICG Workshop Statistical Programming with R

This is the R code to perform the statistical analysis for the ACENET-Supported ICG Bioinformatics Workshop
The session to be held on May 24th 2023 at Dalhousie University.

Requirements:
For running the R script, first, RStudio needs to be pointed to the library in which the packages have previously been installed.
This will avoid long waiting times for installing all the packages.
To perform this, please run
.libPaths("/project/def-sponsor00/R/lib")
in RStudio.

Now, a library command can be used to load the packages.
library(ggplot2)
library(rio)
library(ggpubr)
library(pROC)

This should set the environment for the following steps 1, 2, and 3.

1 - Basic statistical tests

This step will be performed using the ToothGrowth dataset.
It is a native dataset in R, found in the datasets package (RStudio already has is pre-loaded).

You can visualize the all the data:
View(ToothGrowth)

Just the first 5 rows:
head(ToothGrowth)

Just the last 5 rows:
tail(ToothGrowth)

You can view the columns:
colnames(ToothGrowth)

You can view the type of each column (i.e., numeric, character, etc)
class(ToothGrowth$len)
class(ToothGrowth$supp)
class(ToothGrowth$dose)

Basic statistics can be explored on this data (for that, you need to know the type of variable you have).
The mean of the length of tooth:
mean(ToothGrowth$len)

The standard deviation:
sd(ToothGrowth$len)

The minimum value:
min(ToothGrowth$len)

The maximum value:
max(ToothGrowth$len)

We know that the variables dose and supp (i.e., dose of Vitamin C and supplement of vitamin C [orange juice, of tablets]) are not continuously numeric variables.
You can analyze the distribution of the observations according to the dose and supp variables using the table command:
table(ToothGrowth$supp)
table(ToothGrowth$dose)

This table command can be nestled:
table(table(ToothGrowth$dose, ToothGrowth$supp)

You can explore this clusters of variables (based on supp and/or dose) based on a function (i.e., mean, sum)
For this, we need to explore what a function is in R.

We see a function as a system of inputs(x) and outputs(y).
Let's say we want to see mean(function) of the len(inputs) grouped into different supplements of vitamin C (outputs).
x~y
For this, we use the aggregate() function in R.
It's syntax is aggregate(x~y, FUN)
The function might vary, 'mean', 'sd', 'sum'.

aggregate(ToothGrowth$len~ToothGrowth$supp, FUN = 'mean')

You can also have more than one input.
aggregate(ToothGrowth$len~ToothGrowth$supp+ToothGrowth$dose, FUN = 'mean')

Now, we can visualize our aggregate as boxplots using the ggplot2 library.

For that, we need to specify three key items (data, x axis, and y axis).
As the data is already prepared, we can use the aes() argument of a ggplot2 object.

```
ggplot(data = ToothGrowth, aes(x = supp, y = len))+
  geom_boxplot()
```

We can increment our plot to look better.

```
ggplot(data = ToothGrowth, aes(x = supp, y = len, color = supp))+
  geom_boxplot()+
  theme_pubr()
```

As we know our data, we have three different doses of the supplements OJ and VC. We can automatically have ggplot2 to have one plot per level of dose.

```
ggplot(data = ToothGrowth, aes(x = supp, y = len, color = supp))+
  geom_boxplot()+
  theme_pubr()+
  facet_grid(.~dose)
```

Now, we can perform statistical tests in our boxplots.

First, to choose the appropriate statistical tests, we need to understand the distribution of our data.
Also, please note that the statistical test from here on will be performed using the only continuous variable we have, i.e., ToothGrowth$len.

To do that, we can use the Shapiro-Wilk normality test:

```
shapiro.test(ToothGrowth$len).
```

It returned a p value of 0.1091.
For any p value greater than 0.05, we should assume the data follows normal distribution.
In this case, for comparing the average of ToothGrowth$len between OJ and VC, we can use a T test.
For that, we can add a stat_compare_means function in our ggplot2 object. We specify the statistical test (i.e., T test).

```
ggplot(data = ToothGrowth, aes(x = supp, y = len, color = supp))+
  geom_boxplot()+
  theme_pubr()+
  facet_grid(.~dose)+
  stat_compare_means(method = "t.test")
```

This shows us that there are significant differences in the length of tooth growth supplemented by vitamin C (coming from OJ) with the doses 0.5 and 1.

2 - Classification problems

For this section, we will use some data of mine. You can find it at https://raw.githubusercontent.com/gustavsganzerla/ICG_Workshop_R/main/data_covid.csv

We can directly import the data into R with the rio command.

```
data_covid <-
import("https://raw.githubusercontent.com/gustavsganzerla/ICG_Workshop_R/main/data_covid.csv")
```

Now, take some time (up to 10 minutes) to explore the data with the steps covered in session #1.

First, let's check the normality of neutro.

```
shapiro.test(data_covid$neutro)
```

The low p-value = 1.445e-13 indicates the data does not follow normal distribution. So, we can use a wilcox test.

Let's visualize our groups in a boxplot. In this case, we need a pairwise comparison in between each group.
We can tweak the stat_compare_means function to do the comparison between all.
We will run into a problem in which the MOF variable needs to be converted into a factor first as it is an integer.

```
class(data_covid$MOF)
data_covid$MOF <- as.factor(data_covid$MOF)

ggplot(data = data_covid, aes(x = Diagnosis, y = neutro))+
  geom_boxplot()+
  facet_grid(.~MOF)+
  stat_compare_means(comparisons = list(c("sepsis", "sepsis+covid"),
                                        c("sepsis", "septic shock"),
                                        c("sepsis+covid", "septic shock")),
                     method = "wilcox.test",
                     label = "p.signif")
```

We see that neutrophils seem to be higher in septic shock patients when their MOF is 0.
Meaning the patients are not going through a multi organ failure syndrome.
So, we can distinguish this group based on their number of neutrophils.
We can create a subset of our data considering the mof = 0

```
data_covid_mof_0 <- subset(data_covid, data_covid$MOF == 0)
```

To perform a classification task, we need to have a function (x~y).
But the x needs to be a continuous numerical variable.
While y needs to be a categorical variable.

In our data, have the variable neutro (neutro, i.e., neutrophils) as being a continuous numeric variable, let's use it as our x.
And the y variable can be the disease the patient has (i.e, diagnostic, sepsis, sepsis+covid, and septic shock).

There are many algorithms to perform a classification task. Most advanced methods use artificial intelligence.
We will start simple by using a logistic regression model. This is achieved by the glm function in R.
The main arguments are: the function x~y, and the data in which the variables are coming from.

```
model <- glm(Diagnosis == "septic shock" ~ neutro, data = mof_0)
```

Now, we want to assess our predictions whether they worked or not.
We will do that by performing a ROC analysis, which measures how our model reacts with different thresholds in the logistic function.

We will use the pROC package. We are applying our model to all instances in which the diagnosis is septic shock. Then, assessing how the model responded.

```
roc_obj <- roc(data_covid_mof_0$Diagnosis=="septic shock", predict(model))
```

For plotting purposes, we want to create a data frame with the elements in the ROC object.
It has two important columns we will need:
specificity = proportion of positives classified by the model that are actually positives.
Sensitivity = proportion of negatives classified by the model that are actually negatives.

```
roc_df <- data.frame(TPR = roc_obj$sensitivity, FPR = roc_obj$sensitivities)
```

Now, we can plot the ROC curve and see the Area Under the ROC (AUROC).

```
ggplot(data = roc_df, aes(x = 1-FPR, y = TPR))+
 geom_line()+
 annotate("text", x = 0.8, y = 0.2, label = paste("AUC = ", round(roc_obj$auc, 2)))+
 theme_pubr()
```

3. Regression

The data is available at the Github
https://raw.githubusercontent.com/gustavsganzerla/ICG_Workshop_R/main/diabetes.csv.

First, you can find information about it at
https://www.kaggle.com/datasets/mathchi/diabetes-data-set.

Our regression will be composed of a simple function. I will use BMI to predict glucose.

```
model <- lm(BMI ~ Glucose, data = data_diabetes)
```

If I check the summary of the model, we can see that the dependent variable glucose is significantly (*** three levels) capable of predicting the BMI.

```
summary(model)
```

We can extract the coefficients to build the equation that performs our prediction.
For that, we need the intercept and the slope of the regression function.

```
intercept <- coef(model)[1]
slope <- coef(model)[2]

equation <- paste("BMI =", round(intercept, 2), "+", round(slope, 2), "* Glucose")
print(equation)
```

"BMI = 25.4 + 0.05 * Glucose"

Now, we want to assess our prediction. Opposing to a classification model, regression tasks are not that straightforward to evaluate. In our case, we will use the root mean squared error (RMSE), which measures the average difference between the values predicted by our model and the actual values.

We obtain this metric by checking the residuals, which is the difference between observed x predicted. For that, we first need to create a new column in our data, containing the predictions of the model.

```
data_diabetes$predicted_BMI <- predict(model, newdata = data_diabetes)
```

Now, we can calculate the residuals for each observation

```
residuals <- data_diabetes$BMI - data_diabetes$predicted_BMI
```

Now, we calculate the RMSE

```
rmse <- sqrt(mean(residuals^2))
```


```
rmse <- sqrt(mean(residuals^2))
```