Exercice 4.
Matrix manipulation.

First create a script "exercise4.R" and save it to the "Rintro/day2" directory: you will save all your commands in it for that exercise.

1. Create matrix **m** as:
m <- matrix(c(seq(from=1, to=10, by=2), 5:1, rep(2017, 5)), ncol=3)

2. What does function **seq**() do?

3. What are the dimensions of **m** (number of rows and number of columns)?

4. Create matrix **m2** and **m3** as:
m2 <- matrix(c(seq(1, 10, 2), 5:1, rep(2017,5)), ncol=3)
m3 <- matrix(c(seq(1, 10, 2), 5:1, rep(2017,5)), ncol=3, byrow=T)

Why are **m2** and **m3** different?

5. Compute the sum of each row of **m3**:
rowSums()

6. Add column names "day", "month" and "year" to **m2**.
colnames()

7. Add row names A to E to **m2.**
rownames()

8. Select rows of **m2** if the month is superior or equal to 3.
Note: there are 2 different ways of doing this: using the column name or using the column index: try both!

9. Replace all elements that are 2017 with 0 in m3.

10. Multiply all elements of the **2d column** of **m3** by 7.

11. Add one column of **1s** to **m3**: save in **m4**.
cbind().

12. Replace all elements of **m4** that are superior to 2 with 0.

Exercise 5.
Data frame manipulation.

Exercise 5a

1. Create and enter directory **exercise5**.
*Note: remember **dir.create**() and **setwd**() functions.*

2. Copy file**/users/bi/public-docs/sbonnin/Rcourse/ex5_input.txt** into the current directory.
file.copy()
Note: file.copy copies a file from one directory to another.

3. Read in file **ex5_input.txt** the 2 following ways:
ex5_1 <- read.table("ex5_input.txt")
ex5_2 <- read.table("ex5_input.txt", header=T)

4. What difference do you see between **ex5_1** and **ex5_2**?
*Note: compute **colnames**() and **dim**() of each object.*

5. Check the structure of **ex5_2**.
str()

6. Now read in **ex5_input.txt** the following way:
ex5_3 <- **read.table**("ex5_input.txt", header=T, as.is=T)
What is the structure of **ex5_3**?
How does it differ from **ex5_2**?
*Note: read information about option **as.is** in **read.table**() help page and its default value.*

7. What are the column names of **ex5_3**?

8. Change the name of the first column of **ex5_3** to "Country"

9. How many countries are in the Eurozone, according to this table?
table()

10. Replace "**TRUE**" with "**yes**" and "**FALSE**" with "**no**".

11. How many country names contain the letter "**c**"?
Note remember grep(): see help page.

12. How many people live, according to that table:
* in the European union?
* in the Eurozone?

13. Sort **ex5_3** by increasing number of inhabitants per country. Store in **ex5_4.**
order()

14. Write **ex5_4** into file "**ex5_output.txt**".

write.table()
Note: look the help page of write.table, especially options col.names, row.names, sep, quote… Play with these parameters and view the different outputs.

Exercise 5b - OPTIONAL

1. Create two data frame **df1** and **df2** (do not forget to add the column names):

df1

id	age
1	14
2	12
3	15
4	10

df2

id	name
1	paul
2	helen
3	emily
4	john
5	mark

2. Merge df1 and df2 by their "id" column.
merge()
Create 2 different output: one will contain all rows from both data frames, the other ones will contain only the ones that have a common id.
Note: read about the "all" parameter in the help page of merge.

Exercise 6
Packages

1. Install and load packages **ggplot2** and **WriteXLS**

2. ggplot2 loads automatically the **diamonds** dataset in the working environment. What are the dimensions of **diamonds**?

3. Keep only columns **carat**, **cut**, **color** and **price**: store in object **diams1**
Read the help page of the diamonds dataset to understand what it contains.
Note: diamonds is a data frame: is.data.frame(diamonds) returns TRUE.

4. Install and load package **dplyr**.

5. Randomly sample **200 lines of diams1**: save in **diams** object.
sample_n()

6. Export diams into 2 files:
    a. diamonds200.txt – **write.table**()

b. diamonds200.xls – **WriteXLS**()

Note: read about and play with the different options of both functions and check the output files.

Exercise 7 (optional)

1. read file /users/bi/public-docs/sbonnin/Rcourse/gene_setA.txt" into object **de**. **read.table**()
Note: the first row of genes_de3.txt should NOT be read as the header.

2. Add column names to **de**.
Note: The first column is the ensemble gene id, the second column is the gene symbol and the third column is the adjusted p-value calculated for Treatment1 vs WT.

3. How many genes does **de** contain?

4. Select a subset of **de** of genes for which the adjusted p-value is < 0.05 and store it in **de_p005**
How many genes are found in the subset?

5. What is the minimum adjusted p-value in **de_p005**?

6. Remove the rows containing **NA** values in the **padj** column from **de_p005**.
*Note: read about **is.na**() function.*
How many genes are now in **de_p005**?
What is now the minimum adjusted p-value of **de_p005**?

7. Read file "/users/bi/sbonnin/Rcourse/genes_setB.txt" into object **de2** and repeat steps 2 to 6 for **de2**, so as to obtain a subset **de2_p005**.
Note: The first column is the ensemble gene id, the second column is the gene symbol and the third column is the adjusted p-value calculated for Treatment2 vs WT.

8. How many genes are found in common between **de_p005** and **de2_p005**? I.e. How many genes are significantly differentially expressed, based on adjusted p-value < 0.05, both when you treat the cell with Treatment1 or Treatment2?

9. Create a list de_list containing 2 elements: gene ids from de_p005 and gene ids from de2_p005.
Note: a list is simply created using the list() function that takes as argument(s) any other element(s)

10. Name the elements of **de_list** as Treatment1 and Treatment2.
Note: use the names() function as for the vectors.

11. Install and load package **VennDiagram**.

12. Create a simple 2-way Venn diagram using **de_list** as the main argument**.** **venn.diagram**()