Session #2 - Exercises

Adrian C Lo and Ludovic Telley

12/4/2019

INSTRUCTIONS

There are 3 segments with around 10 questions each that increase in difficulty. Fill in the answer within the code chunk. When you wish to test the code chunk, press the *green play button* on the right side of the code chunk to see your output.

The hints will guide you what functions are required. You can access further information with *?x* where x is the function name, e.g. ?print().

Also check this page, specifically sections 8.3 and 10.2 for additional help.

Quick Reminder

For additional (free) learning, visit this webpage

DNF-UNIL Github page

Click here for the resources from the first DNF coding club session.

R is a console where you can type all the calculations you want and see the result

```r
1 + 1
```

```
## [1] 2
```

```r
10 / 5
```

```
## [1] 2
```

```r
2^5
```

```
## [1] 32
```

```r
1 * 5
```

```
## [1] 5
```

You can assign value or text to an object

```r
a <- 2
b <- 3

a + b
```

```
## [1] 5
```

```r
c <- "text"
c
```

```
## [1] "text"
```

You can call a function. E.g, to do a sum

```r
# ?sum

sum(1, 2, 3, 4, 5)
```

```
## [1] 15
```

If you need help about a function

```
# ?c()
```

Exercises

BASIC OPERATIONS

1. Create a vector x containing the elements 0,1,2,3,4,5,6. Create a vector y containing the elements 0,1,2,-3,4,-5,6,10,200,300.

HINT: c()

```
#
```

2. Write an if-else statement to check the first element of x. If it is zero, output "This is good", else output "This is bad".

HINT: print() and logical operator ==

Also check here for extra help.

```
#
```

3. Write a for-loop that iterates over each element of vector x using print()

Also check here for extra help.

```
#
```

4. Write a for-loop that iterates over each element of vector x and print the cube of each value.

HINT: 2^3 = cube of 2 = 8

```
#
```

5. Write a for-loop that makes a right angle triangle pattern where each row is a number x repeating itself x times.

HINT: rep()

The pattern is:

1

22

333

4444

```
#
```

6. Write a for-loop that iterates over the inbuilt iris dataset that evaluates the number of characters in the column name. The output should be like:

Sepal.Length (12)

Sepal.Width (11)

Petal.Length (12)

Petal.Width (11)

Species (7)

HINT: paste0() and nchar()

```
#
```

7. Write a for-loop that iterates over the inbuilt iris dataset ad outputs for each flower/row as output:

Flower **1** is a **setosa** and has a Sepal.Length of **5.1**

```
#
```

8. Write a for-loop that iterates over each element of vector y and only outputs the positive numbers, i.e. ignoring the negative numbers.

HINT: next()

```
#
```

9. Write a for-loop that iterates over each element of vector y, till it reaches value 10. If it reaches value 10, stop the for-loop.

HINT: break()

```
#
```

10. Use the inbuilt rivers dataset and write a for-loop that checks each value. If the value is lower than 500, output "short river". If the river is more than 2000, output "long river". Everything else, output the original value.

```
#
```

READR and DPLYR

1. Import readr and dplyr packages/libraries

```
#
```

2. Import dataset tidy.csv and store it in an object called "mydata"

HINT: read_csv()

```
#
```

3. Check out mydata, how many rows and columns are there?

```
#
```

4. Genotype, treatment and gender should be categorical variables with some order (instead of the default alphabetical order). Transform these into order "WT","GM" for genotype, "CTR_X","TRT_X","TRT_Y","TRT_Z" for treatment and "M","F" for gender. Store it again in object "mydata".

HINT: mutate() and factor(. . . , levels = c(. . .)).

Example: factor(class, levels = c("level_1","level_2","level_3"))

```
#
```

5. Continue from mydata from question 4. Variables are unnormalized. Create three new columns with the variables multiplied by the norm column. Name the new columns var_x_norm, var_y_norm and var_z_norm. Store in an object called "mydata_t".

HINT: mutate()

```
#
```

6. Continue from mydata_t. Filter values that only contain male (gender == "M") samples, store this in an object called "mymales". Continue working with mymales untill question 10.

HINT: filter()

```
#
```

7. Continue from mymales. The summarise() function is very useful for collapsing a large dataset into a single observation. Summarise the mymales dataset to find the following columns / values: min_value_x_norm (minimum value for var_x_norm) and mean_value_x_norm (mean value for value_x_norm).

HINT: min() and mean()

```
#
```

8. In combination with group_by() function, you can summarise the value of interest by each group. For example, grouping according to genotype, you can find the values of question 7 for each genotype.

HINT: %>% makes the line of execution more readable.

Example 1:

function2(function1(x,y))

is more readable as (and equivalent to)

x %>% function1(y) %>% function2()

Example 2:

data %>% group_by(x) %>% summarise(. . .)

is equivalent to (but more readable than)

summarise(group_by(data, x), . . .)

```
#
```

9. Grouping can be performed on multiple levels. Try to find values of question 7 when grouping for genotype and then treatment. Summaries can also be stored in an object. Store this in an object called "mysummary"

```
#
```

10. Export your result with write_csv(). Call it "myfirstroutput.csv"

```
#
```

11. So far you have checked the values of question 7 for only "var_x_norm". Repeat this for each variable ("var_x_norm","var_y_norm","var_z_norm") the values of question 17 when grouping for genotype and then treatment.

```
#
```

GGPLOT2

1. Work with mydata_t from the previous segment. Create a boxplot with on the y-axis var_x_norm and on the x-axis genotype. How does your plot look like? What can you conclude from this?

HINT: ggplot(data, aes(x = x, y = y) + geom_boxplot()

```
#
```

2. The previous graph takes a general view of var_x_norm according to genotype. Add within aesthetics (aes) that treatment is defined by fill.

HINT: fill = . . .

```
#
```

3. Re-use the code from question 2 and adjust it slightly so you have violinplots instead of boxplots.

```
#
```

4. Re-use the code from question 2 and adjust it slightly so you have stacked bars instead of boxplots.

HINT: geom_col()

```
#
```

5. Re-use the code from question 4 and adjust it slightly to have side-by-side bars instead of stacked bars.

HINT: position = "dodge" inside geom_col()

```
#
```

6. Re-use the code from question 4 and adjust it slightly to have stacked bars of equal height, instead of stacked bars with different height.

HINT: position = "fill"

```
#
```

7. Create a scatter plot with on the y-axis var_x_norm and on the x-axis genotype. How does your plot look like?

HINT: geom_point()

```
#
```

8. All points are only segmented according to genotype, but no difference is made between treatments. Add within aesthetics (aes) that treatment is defined by color.

HINT: color = ...

```
#
```

9. Treatments are defined with different colors but are all on top of each other. To make more clear. Add inside geom_point() that position has to be changed. This can be done with the statement position = position_dodge(1). The number (0-1) determines how much space there is between the respective columns within genotype. Try out different values to see how it makes the graph more aesthetic/readable.

```
#
```

10. Besides colors and fills, shapes can also be used as an aesthetic to define categorical groups. Add shape = gender within aesthetics to see what happens. What can you see? How can the visibility be improved? See facet_wrap() or facet_grid() and try out.

```
#
```