

Επιστήμη Δεδομένων Υγείας 1

Εισαγωγή στην R

2024-2025

PART 1

The file **data_exercise.csv** contains data from a random sample of 200 individuals. The available variables are the estimated glomerular filtration rate (eGFR) (in milliliters of cleansed blood per minute per body surface (mL/min per $1.73 m^2$)), a reliable indicator of kidneys' function, the age (in years), the serum creatinine levels (in mg/dl), the gender and the race.

Using R programming language, complete the following tasks:

A. Import the dataset into R (use `read.table()` or `read.csv()` functions). Generate a random positive integer **k** from the set $\{100, 101, \dots, 180\}$. Then, pick a random subset of the dataset, of size **k**, and save it to an object named **data_sample**. Make sure your results are reproducible (*do not forget to set a seed!*). **This will be the dataset you are going to work from now on.** Create a new variable in the dataset, named **log_eGFR**, as the natural logarithm of eGFR. Calculate descriptive statistics for all the variables (mean(sd) or median(IQR) for the continuous ones, n(%) for the categorical ones; you are advised to convert each categorical variable to a factor, with appropriate labels for each label).

B. Construct a new variable in your dataset, named **cat_SC**, which takes the value of 1 when serum creatinine ≤ 0.8 , the value of 2 when $0.8 < \text{serum creatinine} < 1$ and the value of 3 when serum creatinine ≥ 1 . Provide the labels **low**, **normal**, **high** to each value respectively.

C. Produce the following plots:

- a histogram of *log_eGFR*,
- boxplots of *log_eGFR* with separate color according to the levels of *cat_SC*
- a scatterplot of *log_eGFR* to *serum creatinine*, with points colored according to *gender* variable.

For all the plots, provide titles, appropriate names for the axes and legends wherever is needed.

PART 2

A. Create a function, named *count_down*, that accepts a single natural number *n* as input and returns a vector with elements all the natural numbers starting from *n* down to 0. For example, if you input $n = 8$, the function should return $\{8, 7, 6, \dots, 1, 0\}$.

B. If x is a vector and $element$ is a numeric value we know that the command `sum(x == element)` essentially counts the appearances of $element$ in x . Without using the above command, but instead utilizing *for* loops and *if* statements, create a function, named `count_appearances1`, that accepts two arguments, a numeric vector x and a numeric value object named $element$ and returns the number of times that $element$ appears in x . Apply your function for $x = c(1, 1, 1, 1, 2, 2, 3)$, $element = 1$ (it has to return 4).

C. Extend the previous function for $element$ being a vector of numbers. That is, create a function named `count_appearances2` that accepts two arguments: a numeric vector named x and a *numeric* vector named $elements$. The function should return a vector of length equal to the length of $elements$, which contains the number of times that each number in $elements$ appears in x . Apply your function for $x = c(1, 1, 1, 1, 2, 2, 3)$, $elements = c(1, 2, 3, 4)$ (it has to return `c(4, 2, 1, 0)`).

D. Write a function, named `roll_dices`, that accepts as input a numeric value named k . The function has to simulate the rolling of 2 (independent and fair) dices until their sum equals k . It has to return the number of attempts that were needed until the dice roll sum was found to be equal to k . (**OPTIONAL:** Note that since the sum of two dices can only be a natural number out of $\{2, \dots, 12\}$, it will be a good behavior of your function to account for it, that is to be forced to stop (and perhaps return an informative message) if someone inputs a k other than them). **TIP:** Use appropriately the function `sample()` to simulate the rolling of a fair dice.

(OPTIONAL) E. Consider a vector $x = \{x_1, x_2, \dots, x_n\}$. We define the moving average of x of order k at the position p of the vector as the average of the subvector of x , $\{x_{p-k}, \dots, x_p, \dots, x_{p+k}\}$. If at the p -th position of the vector there aren't k elements on the left or on the right of x_p , we define the moving average for this position to be missing (*NA* in R). Create a function, named `moving_average`, that accepts two arguments: a numeric vector x and a natural number k . The function has to return a vector of the same length as x , which will contain in each position the moving average of x of order k . At the positions in which the moving average cannot be computed, according to our definition, the returned vector will contain *NAs*. So, for the vector $x = \{1, 4, 1, 4, 10\}$ the function `moving_average(x, k = 1)` will return `{NA, 2, 3, 5, NA}` and for the vector $x = \{1, 4, 1, 4, 10, 6, 9\}$ the function `moving_average(x, k = 2)` will return `{NA, NA, 4, 5, 6, NA, NA}`.

The deliverable for the assignment is an R script file with your R code for all questions. Separate with comments (use `#`) your answers between the different questions. Optionally, instead of the above, you can submit an Rmarkdown file (html, pdf or word) that combines code with results, comments, etc. (introduction and guide to Rmarkdown can be found [here](#) and [here](#)).