# 

# COMP 5070 Statistical Programming for Data Science

# Test 2: COVID-19 Simulation

* This exercise is a second part of the continuous assessment that is worth 25% of your overall grade in total.
* Your code should be submitted as an R-script using LearnOnline. Do not hardcode any paths on your computer in the code, as I should be able to load and run your code. Your code should load all required libraries BUT don’t install anything on my computer. You can assume that I have all packages already installed.
* The exercise is out of 100 marks. To obtain the maximum available marks you should aim to:

1. Code the requested program (60%). The correct code should use vectorization as much as possible. If you directly convert Python code from Exercise 1 into R, that will be marked as a Fail, due to a failure to use the main feature of R – vectorization.
2. Use a clear coding style (10%). Code clarity is an important part of your submission. Thus, you should choose meaningful variable names and adopt the use of comments - you don't need to comment every single line, as this will affect readability - however you should aim to comment at least each section of code.
3. Have the code run successfully (10%).
4. Output the information in a presentable manner as decided by yourself (10%).
5. Document code limitations including, but not limited to, the requested functionalities (10%).

This assessable exercise can be openly discussed within the group online and you are welcome to share tips and tricks (not entire programs, however).

Having said that, the ground rules are:

* If you use another person’s code in your file, please note the source and how much of the code is not yours.
* If you submit a program cobbled together by other peoples’ code with no, or little, original input from yourself, you will automatically receive a zero mark. The idea is to develop your own programming style with (or without) the help of others, however any code used should support your approach to how you write the program, not replace your own efforts.

Late submission will be penalized by 10-point deduction for each day or part of it after the due date.

If you’re unsure at any point, you’re welcomed to check with me.

# COVID-19 Simulation

In this exercise you should write a code to simulate COVID-19 infection in the society. You could see my presentation on how to code this simulation in Python. On the course web page go to “Past announcements” and then find “Week 3” and link for “Thursday presentation on Exercise 1 solution”.

Simulation rules are almost the same to what I coded using Python but with one interesting addition – partial immunity (see details below).

The main difference between the Python version and R-version of the code would be a use of vectorization. You should use as little as possible of loops. In fact, there should be just one loop for days-count. If anyone tries to use R and create the exact copy of my Python code but in R, the grade will be quite low.

Below are rules that should help you building a simulation model:

1. We assume that there are “N\_population” citizens in a population. “N\_population” is an input parameter. During development and testing you can have it small, but your code should be able to handle a reasonably large value for population size.
2. Every citizen has a health state – “healthy, sick, dead” (or you can code it as 0, 1, 2). As we start all citizens are alive and healthy.
3. To start the pandemic, you randomly mark a small number of citizens as “sick” – there can be a parameter for the number of initially infected citizens. You can start with 1 or 2 initial sick cases.
4. One iteration is one day. During the day, every citizen can meet a random number (say between 0 and 20 inclusive) of randomly selected citizens. You don’t really need to control all citizens as we are interested in meetings of sick people only. We don’t care if and how many healthy people meet each other.
5. Every sick citizen can stay sick and infectious for 10 days, hence you should have some counter for each sick citizen. After 10 days a sick citizen becomes healthy and stops spreading the virus.
6. Obviously, dead citizens cannot become sick, they don’t meet anyone and, as a result, cannot infect anyone.
7. During the day every sick citizen has a probability to die (mortality rate) from the disease with probability 0.2% (very low probability).
8. If a sick citizen does not die, then they can meet other people as per the rule 4 above and if they meet a healthy person, that person might become sick too and start infecting other people starting from the next day (that is an infection day is day 0 of their sickness). The probability for a citizen of becoming sick (infection rate) after a contact is 30% (this is quite high).
9. (New rule!) After surviving the infection and getting healthy, the person becomes immune. This is a partial immunity. It does not make the person “invincible” but reduces the chance of infection in future meetings with sick people. Take the immunity coefficient as 0.1. That is, a probability of being infected for immune person is ten times lower than for the person without an immunity. The infection rate for the immune person is the “original” infection rate multiplied by the immunity coefficient, e.g. (0.3\*0.1).   
   For the test, you can set immunity coefficient equal 1, which means no benefits of immunity and the result should be the same as in Python example I presented.
10. You should run this simulation for a number of days (iterations) and store each day results in a data frame: how many sick citizens in population in total, how many people died, how many new infections per day, something else you might find interesting or useful (e.g. R0). After completing the simulation, you should create a data visualisation of the history of infection.
11. You can run simulations for a predefined number of days, say 100 or 300 days, or till some natural outcome – all citizens get healthy, or all citizens die.
12. Keep all parameters in the beginning of the code, so you can easily change them without a need to change anything in the code.
13. Try to change parameters of the model – increase mortality rate (more dangerous virus scenario – SARS and MERC had 3% mortality); or decrease infection rate (say, 5% – wearing masks and social distance scenario); or increase the length of sickness period; or reduce the number of possible contacts for all citizens to a range between 0 and 2 (lockdown scenario). Make a brief comment (3-4 lines as comments at the end of the R-script) on what parameters have the strongest effect on the infection growth.

The main purpose of this exercise is to test your understanding of programming in R: using vectorization, indexing, plotting. For this exercise, weeks 7 and 8 materials are more than sufficient.

**Hint on getting a random event for a given probability**

If you know that some event has probability 75% (e.g., coin tossing on head – this is clearly a biased coin), then you generate a random number from a uniform distribution between 0 and 1, e.g.,

x <- runif(n = 1)

And then, if x < 0.75 is TRUE we assume coin to be on head; if FALSE we assume the coin is on tail.

Thanks to vectorization we can create a very large number of random numbers by changing parameter n and then test all these numbers if they are less than 0.75.

Beware of a relationship between percentage and proportion in probability: 75% means 0.75; 0.2% means 0.002.

**Hint on indexing**

If you have a collection of some numbers, say

x <- c(1,3,0,5,4,7,2)

You can check what numbers are greater than 3 and you will get a vector of logical values

x > 3

[1] FALSE FALSE FALSE TRUE TRUE TRUE FALSE

You might be interested what is an index for each value that is greater than 3. It can be done by function which()

which(x > 3)

[1] 4 5 6

For indexing, that is, to extract or to change values, you can use a vector of logical values or a vector of indexes – whatever works better for you in each situation.