**Project Exercise**

## January 2023

# Instructions & Notes

This is an exercise in data science programming methods. Use either the project\_dataset.csv (MIMIC-III based) or the framghm2.csv (BioLINCC Framingham based) datasets.

## Step 1: Design/split-up the programme on paper.

We want to run through the typical processes involved in a health data science project:

1. Read in the data
2. Change the column names, remove unnecessary columns
3. Remove / update bad data
   1. Missing data
   2. Wrong/inconsistent data
4. Decide which data to keep
   1. Range of available data
   2. Study design requirements
5. Explore the data
6. Describe the data
   1. Table 1 – how many participants / patients / records with / without feature
7. Analyse the data
   1. Statistical tests, regressions, adjustments for bias
8. Communicate your results
   1. Summarise the test results and interpret them
   2. Produce data visualisations

## Step 2: Create a project and organise the files.

Create an R project and decide where files of different types go. This will depend on the complexity of the project and the size of the project team.

## Step 3: Start / Clean down the project environment.

Clean the data environment, the plots, the console and restart your R session. Re-run your code to restore these to a known state.

## Step 4: Design each of the components associated with each task on paper.

Each component needs to have its functionality clearly defined and the inputs and outputs specified. Add comments to the code to explain these things.

Report the status of your dataset after each function or group so that you might spot problems if you run it again later. For example, if you read the file and you have 1,763 records of data, you might display some text indicating how many rows you found this time and that you expect to see 1,763. That way, when you run again later, if your data has been corrupted, you should spot that the actual number you have read is not 1,763.

## Step 5: Implement the tasks.

Each task could be implemented as a function. The functions need to be documented so that the users (other members of the team) can understand them. Ideally you should commit your code to version control (git) at this stage (or more often) with a suitable comment. You will work through this iteratively until a working 'main' programme is achieved. If necessary, further separate difficult tasks, or move onto new tasks if any prove easy.

You can push/pull your committed code to a github respository if you are collaborating on a large project in a team. This is fine for learning but is NOT appropriate for your assignment work!

## Step 6: Make sure the design fits together.

When you complete a component, make sure it is compatible with existing components and review their design if necessary. Make sure you update any comments.

## Step 7: Test the component.

Run the code with some test data or with a subset of the real data for which you know the outcome. Do this several times for several sets of data (decide on what you want to check – e.g. situations where the input data or results may be unrealistic).

## Step 8: Re-run the entire code and check that the whole thing works and that the existing components still work / produce the expected output.

Clear down your R environment and re-run the code. Check that all the counts of rows or other features are still correct. Add checks for the latest component.

## Step 9: Repeat until you have completed all the required components.

Run the final code and report the results. If you have git, commit this vesion with a clear label that this is the version used to generate your report.