A practical interdisciplinary PhD course on exploratory data analysis

Anna Górska

November 8, 2021

The aim of the entire course is to walk you through the general data-analysis pipeline. That is being able to write a decent (i.e. well structured, documented) code to: load + clean data, read/write data into a sensible file format, harness the power of plotting, generate output documents: excel, maybe latex. As you are already a programmer, and possibly know python programming language, you can to learn in a *project-based* manner, i.e. via solving the following exercises. Those are separate tasks that are intended to build upon each-other, so solve them within one *project*. Each section has a set of questions you should consider answering before the moving froward, I hope they will help you to create a high-quality data-analysis pipeline that would enable you to further build upon and extend.

Please see the dataset: `https://leoss.net/customs/public_data_set_analysis.html` coming from the LEOSS COVID project. Take a look at the data description and download a public databaset.

1 Exercise 1: warmup

Setup your IDE and make a first script. It has to have a main function and be able to read in parameters, so its possible to run it from the command line like this: `python3.9 main.py my_data.csv`. The program should say sth like this: `I'm going to work with my_data.csv file!`.

Checklist:

- *What are the naming conventions in python? How should your variables and functions be named? Check out PEP8.*
- *What is a proper way to include inline parameters?*

2 Exercise 2: read in the data

Add another script to your project, this script should be able to read the csv file. Import it in your main file, read the csv file, and print the basic statistics about it, such as (filling in with real data):

`There are .. rows and .. columns in my_data.csv file`

- `column A: type of variable: categorical/binary, val1: 59%, val2: 41%`
- `column B : type of variable: categorical/binary, val1: 59%, val2: 41%`
- `...`

Quality checklist:

- *What is a proper way to execute main?*
- *Any nice ways to print things in the command line?*
- *BASH: how to direct the outcome to the new file in bash?*

3 Exercise 3: write a new excel file

Make another script, this time to write to the excel file. The excel file should contain the above outcome, i.e. a description of the data. The outcome should be ready-to-use and well formatted so that you can send it directly to your collaborator. There should be a header with bold font. Color-code the categorical and numerical variables by changing the color of the cell of the name of the variable name.

Quality checklist:

- *Encapsulate! Your functions should be sweet and short (ideally 4 lines), should accept short list of parameters.*
- *Use list comprehension - checkout what is a list comprehension, and replace where possible, makes the code faster!*
- *Use lambdas - checkout what is a lambda function, and replace where possible, makes the code faster!*

4 Exercise 4: write a class for patient

Each row in the dataset corresponds to a patient. Write a class that represents a single patient. Write a python class that deals with a single patient. A patient should be able to introduce themselves: `Patient PID: .. , has .. years old, outcome: ...` Add a function that turns the excel file into a dictionary of patients, where the key is the PID, and the value is the object patient.

Add an option to the code that calls for the patient introduction, like this: `python3.8 main.py --func introduce --pid 237`

Quality checklist:

- *What is a nice way to read the parameters from the command line?*

5 Exercise 5: harness the power of the class

Compute and add the *derived* variable to the patient. e.g. BMI. Program another class, Patients, to deal with the analysis of the patient group.

6 Exercise 6: plotting!

Add plotting for the group of patients:

- Barcharts by different populations.
- Stack-charts for the binary/categorical variables.
- Maybe a heatmap, maybe a cluster-heatmap?
- A nice way to visualize the outcome?

Quality checklist:

- *Consider ink to information ratio.*
- *You might have used some package to program the plots. Consider using the matplotlib - I know its not the easiest package but enables a great deal of flexibility.*
- *Put all similar plots in one panel.*