Final Project

[Full mark: 100; 70% of module grade]

BEE1038: Introduction to Data Science in Economics

In this project, you will demonstrate your understanding and mastery of programming in Python using data science tools, in addition to showing your understanding of the different research methods that use big data.

What you learnt so far should cover almost everything you will need, so if you are stuck then read through the notebooks again. Some of your tutorials and exercises may be also relevant. For Problem 4, you may need to consult the "Bit by Bit" book[1]. If you are still unsure, then have a look online. *Google* and *Stack OverFlow* are your friends. You must use Python only; MATLAB and Excel are not acceptable.

The grade of this project will be calculated out of 100, and it will contribute 70% towards your overall grade in the module. The following aspects need to be shown:

- Manipulation of different data structures including dictionaries and data frames.
- Preparing and preprocessing data.
- Doing a basic plot, and changing plot markers, colors, etc.
- Improving and extending analysis.
- Showing an understanding of different research methods for using big data.
- Showing the ability to critique some scientific approaches.

Your submission will be a compressed file (.zip) containing:

1. A copy of your Python script named **solution_code.ipynb** (done in Jupyter Notebook). Your scripts must be sufficient to reproduce your answers to Problems 1-3.
2. Same copy printed as a PDF, **solution_code.pdf** [2]
3. A PDF file named **P4_answers.pdf** that contains your answers to Problem 4.

The deadline is **Monday 25th April at 3:00 PM (BST).**

Collaboration: You are encouraged to think about this project with other students or ask each other for help. If you do, you should keep the following in mind: 1) write your own code (no code copying from others), 2) Report the names of all people that you worked with in your submission, 3) if you received help from someone, write that explicitly, 4) **plagiarism of code or writeup will not be tolerated**, and 5) do not post your solutions online (even after the release of your marks). For those who want to

[1] https://www.bitbybitbook.com/en/1st-ed/preface/

[2] See here how to create a PDF from a .ipynb file: https://vle.exeter.ac.uk/mod/forum/discuss.php?d=194496

evidence your experience to recruiters, make sure you share a private link to your project/work (or undiscoverable link). **If I can find your answers online anytime until September this year, you will be reported for misconduct.**

The University takes poor academic practice and misconduct very seriously and expects all students to behave in a manner which upholds the principles of academic honesty. Please make sure you familiarise yourself with the general guidelines and rules from this link[3] and this link[4].

Problem 1 [25 marks]: Eurovision Winners Data Set

In this problem, you will use a data set called *eurovision_winners.csv* (see folder *data*). This data set has information about the winner of each year from 1957 till 2018 (both included), including Year, Host City, Winner, Song, etc.

Please follow the instructions below for your data analysis.

A. Load the *eurovision_winners.csv* file in your notebook, and print the data set to screen. Print the column headings.

B. Each row represents a winner in one year. This data set contains 65 rows, but the number of years included are 62 years (1957-2018). Do some investigation to explain where the 3 additional rows came from. Show your work in code (opening the .csv file is not an acceptable answer).

C. Create a new data set '*limitDF*' that contains only the following columns: ' Year', ' Points', ' Margin'.

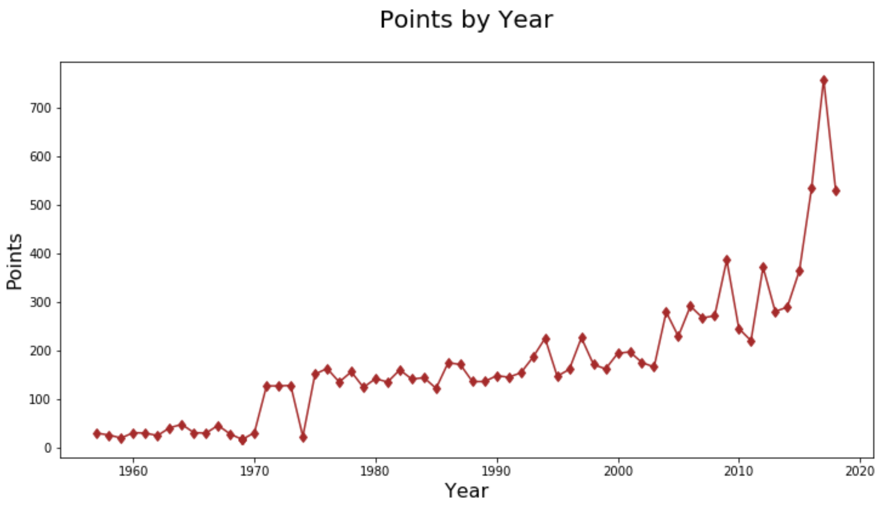
For the remaining questions, go back to the main data set.

D. Plot the histogram for the number of points by winners (*'Points'*), using 7 bins, alpha=0.7, green filling, and black edges for the bars. Add illustrative labels to x-axis and y-axis, and resize plot and axes' label font size to make the plot look visually acceptable.

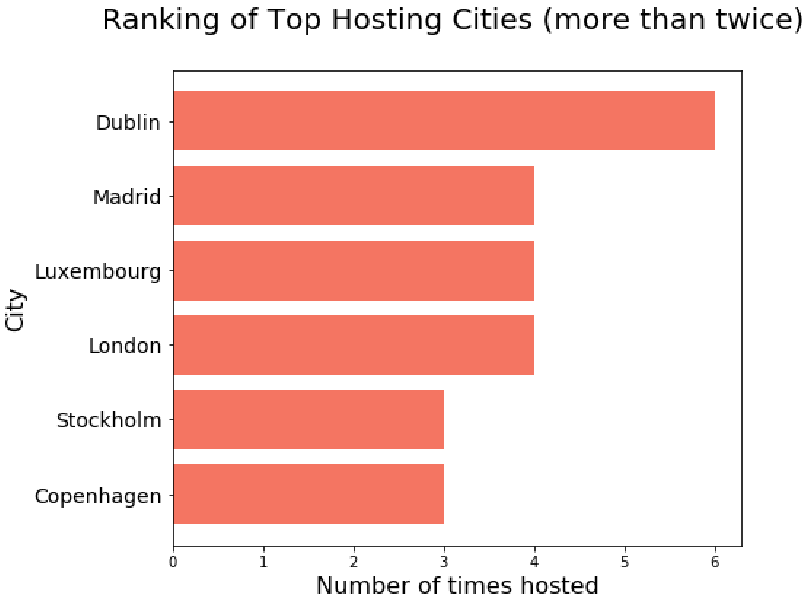
E. Plot the winner points (*'Points'*) over years (*'Year'*). Your plot should look exactly (or as close as possible) to the one below. you will get marks for reproducing the plot as accurately as possible, taking into consideration the steps undertaken to reach the final figure. How do you explain the rise in points over years? Write your explanation in a "Markdown" (textual) cell.

[3] http://as.exeter.ac.uk/academic-policy-standards/tqa-manual/aph/managingacademicmisconduct/

[4] https://vle.exeter.ac.uk/pluginfile.php/1794/course/section/27399/A%20Guide%20to%20Citing%2C%20Referencing%20and%20Avoiding%20Plagiarism%20V.2.0%202014.pdf



F. Reproduce the following plot: you will get marks for reproducing the plot as accurately as possible, taking into consideration the steps undertaken to reach the final figure.



G. In this part, we will see what song words are associated with the highest points. This is not going to be very interesting because the data is thin. But it would be a good exercise. There is a column named ‘Song’. Each song has multiple words (separated by “ “). For example, the words for the song in the first row are:

"Net als toen"

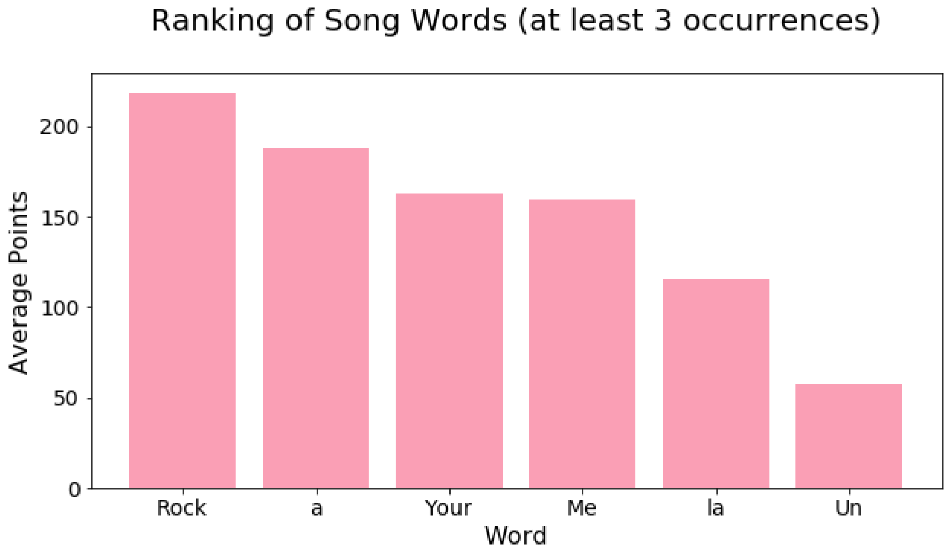
Which consists of 3 words: “Net”, “als”, and “toen”.

We need to do some pre-processing. Ideally, we want to create a new data set, *word_points*, that has two columns: *word*, and *points*. Each row will have only one word and the points of the winning song that has that word. For example, the points for the song in row 0 is 31. This means that each of the three words in this song will have these points in the new data set. I.e., the first three rows are: *('Net', 31), ('als', 31), ('toen', 31).* Here is one way to do it (other ways are acceptable; with some deductions for less efficient methods):

a. Write a function that takes one row and returns a list of 2-dimension tuples. For example, for row 0, it returns:

   [*('Net', 31), ('als', 31), ('toen', 31)*]

   Test your function on row 0.
b. Use the method apply() on all rows to produce a one list containing all 2-dim tuples
c. Convert the list of tuples into a data frame
d. Filter out words that were listed strictly fewer than 3 times
e. Calculate the mean points per word, and sort
f. Plot it like this one:



H. Use the data set to perform compelling extra analysis. You will get marks if you create a compelling and interesting visualisation and/or analysis. One plot (or one piece of analysis) is enough, but you can do more if they are all tied into one main idea. Please provide 1-2 sentences to explain your interesting analysis. Write it in a separate cell inside Jupyter (using Markdown).

Problem 2 [20 marks]: UEFA Euro 2020 Scores Data Set

UEFA Euro is the international men's football championship of Europe organised by (UEFA). The 2020 version took place during the summer of 2021 in multiple European cities, with the final being hosted at Wembley Stadium (London). The competition included 28 teams representing their countries, which played in two parts: Group Stage and Knockout Stage. In the Group Stage, the 28 teams were divided into 7 groups of 4 each. Each pair of teams in a group played one game, a total of 6 games per group. The 6 games were played in 3 Matchdays (2 games per group per Matchday). Matches usually assume a designated “home” team and an “away” team. In some cases, the home team is the host, in other cases, the host is a third country (but the away team will have to change their jersey if there’s a clash in colour with the home team). Each team earns points based on the results of each of the 3 games they play in their group. The points are used to calculate who should move on to the Knockout Stage. A match between two teams can end up in one of two ways:

a) Draw: both teams score the same number of goals. In this case, each team receives 1 point.

b) Win/Lose: one team scores more goals than the other. In this case, the winning team receives 3 points, while the losing receives 0 points.

At the end of the three Matchdays, teams are ranked in each group based on their total points. For the sake of this problem, we will assume that ties are broken first by goal difference (goals scored minus goals received), and then by goals scored.[5]

In this problem, you will use a data set called *eurocup_2020_results.csv* (see folder *data*). The data set contains the results for the UEFA Euro 2020 matches (summer 2021). This data set has information about 51 matches, including team names, number of goals by each, number of shots, etc. For this problem, you can ignore the Knockout Stage games (rows 0-14). You will write a function that calculates the final tables of each of the 7 groups given the results in the Group Stage only (rows 15-50). Please follow the instructions below.

A. Load the *eurocup_2020_results.csv* file in your notebook, and print the data set to screen. Print the column headings.

B. Print a list of the unique teams (no repetition) using ‘team_name_home’. Similar to what you did in the last assessment (Assignment 30%), you will notice that team names have an unnecessary leading space (e.g., “ Italy”). Write a code to remove the leading space from every row in the ‘team_name_home’ and ‘team_name_away’

[5] In reality the tie-breaking rules are slightly more complicated.

columns. Save changes to your data frame. Re-run code that generates unique team names to make sure it is solved now.

C. Create a new data set '*df_G'* that contains only data from the Group Stage (rows 15-50). For the remaining parts, we will work only with this data set.

D. Write a function calcScore() that takes one row as an input (call it *'row'*), and it returns two variables: numbers of points for the Home team and the Away team. You will use the scores from columns 'team_home_score' and 'team_away_score', which include the goals scored by the teams named in the columns 'team_name_home' and 'team_name_away'. The rules of point calculations is the following: If two teams score the same number of goals by the end of the game, each receives 1 point. Otherwise, the team that scores more goals, receives 3 points, while the losing team receives 0 points. The first line of your function will look like :

def calcScore(row):

Test your function on row 17: *calcScore(df_G.loc[17,:])*

E. Using the function you wrote, and using the method *apply()*, create two columns 'team_home_points' and 'team_away_points' that contain the number of points awarded for Home and Away teams respectively. [Hint: check *zip* and *lambda*]

F. Write a function *GetGroupTeams ()*, that takes as an input a team name (string). The function return the list of teams that are in the same group as the input team. Test that your function works correctly by calling it for 'England'

GetGroupTeams('England')

It should return the following (order is not important):

['England', 'Scotland', 'Croatia', 'Czech Republic']

G. Write a function *GetGroupRows()*, that takes as an input a team name (string). The function return a 6-row subset of the data frame containing the Group Stage matches by the teams in the same group as the input team. Test that your function works correctly by calling it for 'England'

GetGroupRows('England')

It should return a data frame of 6 rows, all showing games between the teams of England's group.

H. Write a function *GetTeamStats ()*, that takes as an input a team name (string). The function return a dictionary containing the number of points, the number of goals scored, and the number of goals conceded by the input team. Test that your function works correctly by calling it for ‘England’

GetTeamStats('England')

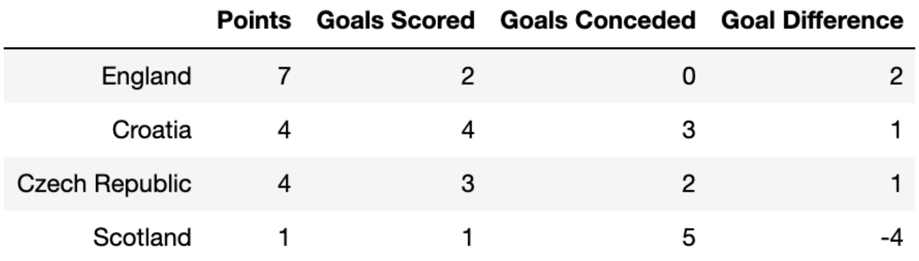
It should return the following (order is not important):

{'Points': 7, 'Goals Scored': 2, 'Goals Conceded': 0}

I. Write a function *GetGroupTable()*, that takes as an input a team name (string). The function returns a data frame with 4 rows (row for each team in the group of the input team), and 4 columns: ‘Points’, ‘Goals Scored’, ‘Goals Conceded’, and ‘Goal Difference’. The teams are ordered first by number of points, and in the case of a tie, by goal difference, then by goals scored. Test that your function works correctly by calling it for ‘England’

GetGroupTable ('England')

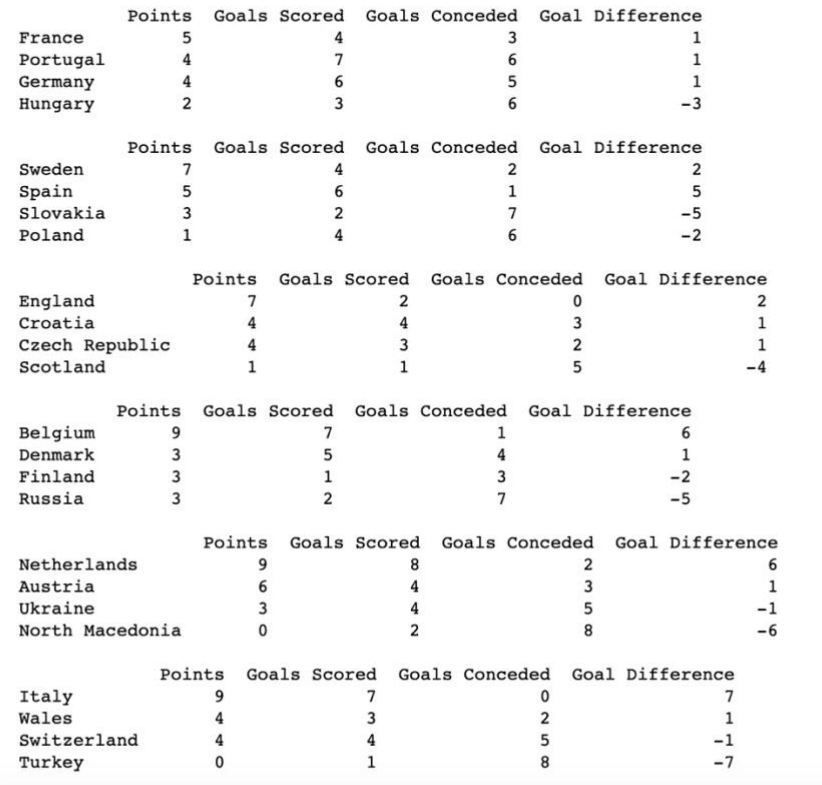
It should return the following:



	Points	Goals Scored	Goals Conceded	Goal Difference
England	7	2	0	2
Croatia	4	4	3	1
Czech Republic	4	3	2	1
Scotland	1	1	5	-4

Just to be sure, test out your function again by calling it for ‘Italy’

J. Using the function above, write a function *GetAllGroupTables()*, that returns the table for all 7 groups. The output should look like the following:



```
          Points  Goals Scored  Goals Conceded  Goal Difference
France         5             4               3                1
Portugal       4             7               6                1
Germany        4             6               5                1
Hungary        2             3               6               -3

          Points  Goals Scored  Goals Conceded  Goal Difference
Sweden         7             4               2                2
Spain          5             6               1                5
Slovakia       3             2               7               -5
Poland         1             4               6               -2

                Points  Goals Scored  Goals Conceded  Goal Difference
England              7             2               0                2
Croatia              4             4               3                1
Czech Republic       4             3               2                1
Scotland             1             1               5               -4

         Points  Goals Scored  Goals Conceded  Goal Difference
Belgium       9             7               1                6
Denmark       3             5               4                1
Finland       3             1               3               -2
Russia        3             2               7               -5

                 Points  Goals Scored  Goals Conceded  Goal Difference
Netherlands           9             8               2                6
Austria               6             4               3                1
Ukraine               3             4               5               -1
North Macedonia       0             2               8               -6

             Points  Goals Scored  Goals Conceded  Goal Difference
Italy             9             7               0                7
Wales             4             3               2                1
Switzerland       4             4               5               -1
Turkey            0             1               8               -7
```

K. Use the data set to perform compelling extra analysis. You will get marks if you create a compelling and interesting visualisation and/or analysis. One plot (or one piece of analysis) is enough, but you can do more if they are all tied into one main idea. Please provide 1-2 sentences to explain your interesting analysis. Write it in a separate cell inside Jupyter (using Markdown).

Problem 3 [25 marks]: COVID-19

In this problem, you will use a data set, named *owid-covid-data.csv* (see folder *data*), from "Our World in Data" website.[6] The data is about COVID-19 cases in all countries on every day since 1 January 2020 until 13 March 2022. The website provides many visualisations related to COVID in different countries and over days.[7] The data you will find in data folder is downloaded from this link which has more information about the data.[8]

Please follow the instructions below for your data analysis.

A. Load the *owid-covid-data.csv* file in your notebook, and print the data set to screen. Print the column headings.

B. Using the column *date*, print the number of unique dates in the data set.

C. Using the column *iso_code*, print the number of unique countries in the data set.

D. From C, you will notice that there are too many countries. As it turns out, there are 15 values in *iso_code* that do not correspond to countries. Let's remove all rows that correspond to those values. They all start with the string "OWID_". Re-run the code from C again, it should be 15 fewer. Make sure you reset the index of your data frame.

E. For the rest of this problem, we will focus on part of the data; the dates after the introduction of vaccination. Create a new data frame, *df_v,* for which the column 'new_vaccinations' is not null. Using the column '*date'*, find the earliest date in your new data frame. **Unless explicitly specified, use the data frame *df_v* for all remaining questions.**

F. Using the column *'total_vaccinations_per_hundred'*, create a new column *'total_vaccinations_per_million'.*

G. Using the column *'date'*, create a new column *'DaysSince3Dec20', which contains the number of days between 3 December 2020 and the current date of the row.*

H. As you may have noticed already, each country is represented by many rows (each row represents one day for one country. Some columns have a different value every day per

[6] https://ourworldindata.org

[7] see this for example about vaccination progress in different countries: https://ourworldindata.org/covid-vaccinations

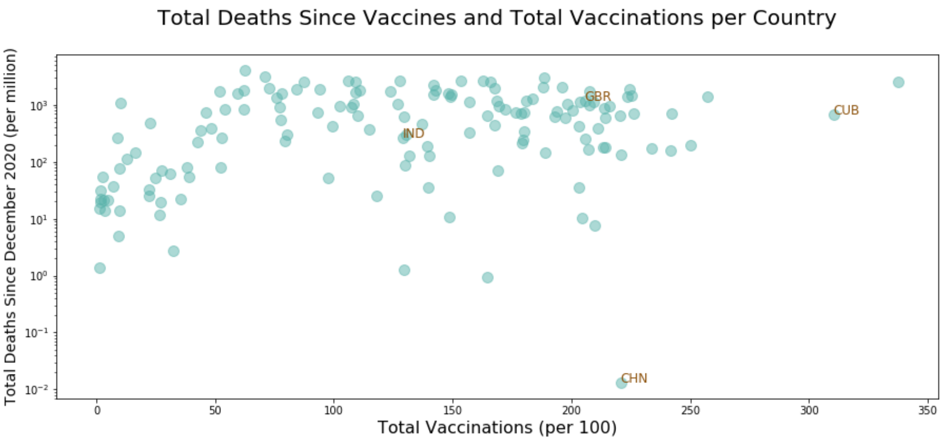
[8] See this for general info: https://github.com/owid/covid-19-data/tree/master/public/data and this for codebook: https://github.com/owid/covid-19-data/blob/master/public/data/owid-covid-codebook.csv

country. These are the COVID-related columns (e.g., 'total_deaths_per_million', 'total_cases_per_million'), others have the same value per country repeated every day (e.g., 'diabetes_prevalence', 'female_smokers', 'male_smokers'). Using *groupby*, we want to create a new data frame *'df_country'* that contains data at the level of each country (i.e., each row represents one country). The new data frame should contain one value for each country for the following columns (and it only has these columns):

'iso_code',
'total_vaccinations_per_hundred' (on the latest date for that country),
'total_deaths_per_million' (on the earliest date for that country), and
'total_deaths_per_million' (on the latest date for that country).

Create a new column, *TotalDeathPerMSinceVac*, the difference (subtraction) between the last two columns.

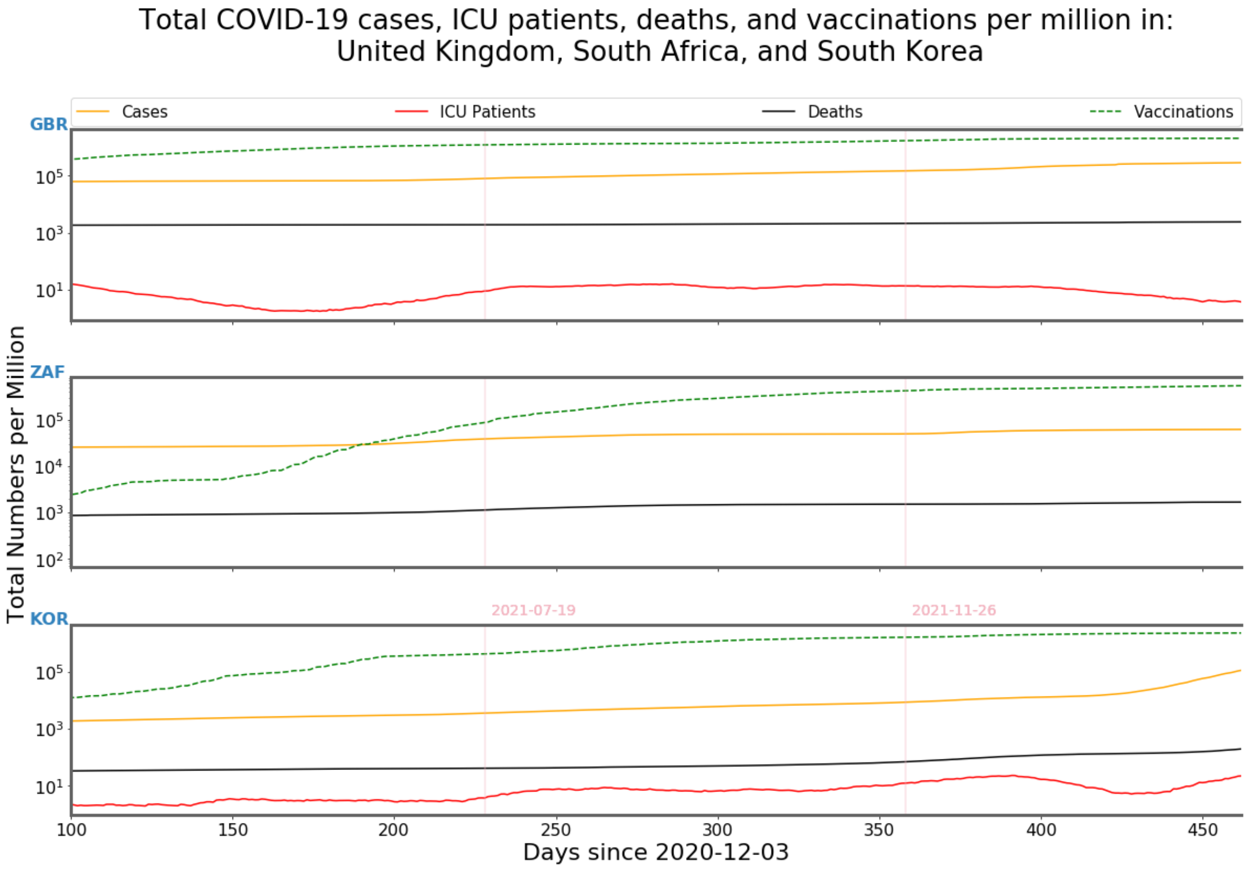
I. Using *'df_country'*, reproduce the following plot: you will get marks for reproducing the plot as accurately as possible, taking into consideration the steps undertaken to reach the final figure.



J. Create a new data frame *'df_GBR'* that contains data only for the UK (using iso_code= GBR; should be 424 rows).

K. In *'df_GBR'*, create a new column *'total_vaccinations_per_hundred_smoothed'*, which contains the moving average of the column 'total_vaccinations_per_hundred', with a window size of 7 i.e., moving average of row x is calculated from rows x-3, x-2, x-1, x, x+1, x+2, and x+3. For x=1, 2, 3 the moving average is calculated using the first 4, 5, 6 rows, respectively (same for the last three rows).

L. Reproduce the plot shown below, as accurately as possible.



The plot is self-explanatory. You will get marks for reproducing the plot as accurately as possible, taking into consideration the steps undertaken to reach the final figure.

M. Use the data set to perform compelling extra analysis. You will get marks if you create a compelling and interesting visualisation and/or analysis. One plot (or one piece of analysis) is enough, but you can do more if they are all tied into one main idea. Please provide 1-2 sentences to explain your interesting analysis. Write it in a separate cell inside Jupyter (using Markdown).

Problem 4 [30 marks]:

In this problem, you will demonstrate your understanding for the materials covered in the module from “Bit by Bit” book. Your answers **should not exceed 1000 words in total**. Write your answers in a separate PDF file. Consider the following research question and answer the questions below:

> *Does employing a harsher punishment result in a better rule following?*

A. Hypothesis: Before collecting any data, and using your intuition and prior knowledge, do you think the answer to the above question is:
   a. Yes, it does
   b. No, it results in a worse rule following
   c. No, it has no effect
   d. It depends
   e. Impossible to tell
   f. Other [elaborate]

B. Justifications: Lay down some reasons that support your answer to the previous question.

C. Constructs: what are some potential measures for each of the following:
   a. Harsh punishment
   b. Rule following

D. Methodologically speaking, what is a good research design for answering the above question? Provide a research plan that uses one or more of the four research designs (Chapters 2-5) in the book.

E. What are some ethical issues you should be aware of when trying to answer this research question or when following the research design you outlined? How to deal with them?

F. You see a published study showing that countries with more severe punishment rules (e.g., caning, whipping) showed lower cases and deaths during COVID-19. The study authors concluded that national rules with stricter punishment are a necessary tool in the face of international-level and country-level disasters because they make citizens more likely to stick to the rules and harmonise their actions. Discuss the potential problems of this post in terms of *validity* (Section 4.4.1).