Final Project

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

BEE2041: Data Science in Economics

In this project, you will demonstrate your mastery of programming using data science tools.

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.
- Preparing and preprocessing data.
- Producing high quality visualisations.
- Improving and extending analysis.
- Utilising the above skills to perform high quality projects.

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

1. A directory/folder called **p1_git** that contains files and folder as asked in Problem 1.
2. A copy of your Python script named **p2_vaccine.ipynb** (done in Jupyter Notebook). Your scripts must be sufficient to reproduce your answers to Problems 2.
3. A copy of your Python script named **p3_warmup.ipynb** (done in Jupyter Notebook). Your scripts must be sufficient to reproduce your answers to Problems 3.
4. A copy of your Python script named **p3_2_Colab.ipynb** (done in Google Colab or Jupyter Notebook). Your scripts must be sufficient to reproduce your answers to Problems 3_2. Alternatively, you can combine the code from this file with the previous one.
5. A copy of your data fille inaug_speeches_LLM.csv which contains the scores of the US president speeches that you generated using RoBERTa
6. A copy of your Python script named **p4_mini_project.ipynb** (done in Jupyter Notebook) that includes code and textual description and read as a blog post.
7. PDF copies of 2, 3, 4, 6: **p2_ vaccine.pdf**, **p3_warmup.pdf**, **p3_2_Colab.pdf**, and **p4_mini_project.ipynb** [1]
8. A text file named **Declaration.txt** in which you provide a declaration of collaboration and usage of ChatGPT or any other Large Language Model (LLM) based software (more below). If you did not collaborate with anyone and did not use any LLM, say that explicitly.

The deadline is **Monday 24th 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 consider the following: 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**

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

not be tolerated, and 5) do not post your solutions online (even after the release of your marks). For those who want to 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 only exception is Problem 1.)**

<u>Use of ChatGPT and other LLM-based software:</u> In this project, you are allowed to consult with ChatGPT as well as any other Large Language Models or software built on top of it (e.g., Bard, LLaMA). If you do that, then you HAVE TO DO THE FOLLOWING: 1) write your own prompts, answers, and code (no answer or code copying from any LLM-based software to your answers and no copying of prompt from the brief to these software) and 2) report which LLM-based software you used (e.g., ChatGPT) and how you did that in the declaration text file (see above). I already tested out prompts with these models. If your submitted results are like the answers I got, your paper will be flagged for misconduct. Moreover, failure to declare your usage of these models will likely work against your case.

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[2] and this link[3].

Problem 1 [10 marks]: Git/GitHub

In this problem, you will demonstrate your ability to use simple commands in Git and GitHub. Do the following:

A. Create an empty directory p1_git.
B. Initialise a local .git repository inside p1_git
C. Create a new file *file1.txt* with one row content: “Hello World! First line.”
D. Commit a .git change with a message “Add a first file”
E. Create a second file *file2.txt* with one row content: “This is a second file.”
F. Add a second row to *file1.txt*: “This is a second line in file1”.
G. Commit a .git change with a message “Add a second file” (this includes adding file2 only)
H. Commit a .git change with a message “Add a second row to file1” (second line to file1)
I. Create a *public* GitHub repository, and name it: *bee2041_p1_g*
J. Push your work to *bee2041_p1_g* repository.
K. Add a third file, file3.txt with a message: “Add a third file”.
L. Commit a .git change with a message “Add a third file”
M. Push your most recent changes to *bee2041_p1_g* repository.
N. Copy the link to your GitHub repository *bee2041_p1_g* and paste it in the first Jupyter Notebook cell of Problem 2
O. Create a .zip file that contains directory p1_git and its content. Add it to your submission package.

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

[3] 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

Problem 2 [20 marks]: Contagion in networks

In this problem, you will show your understanding of epidemic models and contagion in networks, a topic we covered in Weeks 10-11. While you are not asked to write a simulation from scratch, you will be adding some code and modifying existing code.

For this problem, we will be creating network models (Erdos-Renyi and Barabasi-Albert), and we will be using an SIR (Susceptible–Infected–Recovered) epidemic model with vaccines. In this model, at every time step, each node can be in one of the following four states:

a. Susceptible: cannot infect others, is not infected, and can be infected
b. Infected: can infect others, is infected, and cannot be double-infected (whatever that means). Infected nodes may either stay Infected, turn into Recovered, or turn into Immunised.
c. Recovered: cannot infect others, is not infected, but can be infected (again).
d. Immunised: cannot infect others, is not infected, and cannot be infected for a few time steps (immunisation period). At the end of immunisation period, nodes become Susceptible again.

In the initial state of the network, most nodes start as *Susceptible*, a few (with probability *"probInfectionInit" e.g., 10%*) start as *Infected*, and other few nodes (with probability "probImmunityInit") start as *Immunised*. Every iteration (time step), susceptible nodes may become infected if one of their neighbours is infected. Basically, an Infected node can infect each of its Susceptible neighbours with a probability "*probInfection*". Moreover, every iteration, each Infected node may recover (thus moves to "Recovered" state, with a probability "probRecovery"). In this setup, once a node becomes Infected and then Recovered, it can return to being Infected again. Please follow the instructions below.

A. Create a copy of the file *P3_Contagion_Vaccine.ipynb* from Week 11 in your directory.[4] You will be making changes on this new copy.

B. The setup described above is different from the one in *P3_Contagion_Vaccine.ipynb*. Therefore, we need to make some changes. Change the following lines to:

```
max_iterations = 10
probInfectionResidual = 0.0
randomInitialInfection=True
```

This means, we are changing the maximum number of iterations to 10. We are not allowing agents to be infected randomly other than through their infected neighbours. We are also specifying that nodes will be chosen randomly to start in the state Infected at the start of the simulation.

However, we want vaccination to be allowed in two different ways: 1) random vaccination and 2) targeted vaccination (highest degree nodes are initially vaccinated). For this, we introduce a new parameter randomInitialVaccination. When this variable is true, we will do random vaccination, and when it is false, we will do targeted vaccination.

[4] Find it here: https://vle.exeter.ac.uk/mod/folder/view.php?id=1904985

To make this less confusing, we will start first by assigning nodes into Susceptible or Infected (using random infection). Then, we will do a round of potential re-assignment of vaccination (random or targeted) on a proportion of nodes. Nodes that are initialised as Infected cannot be initialised as Immunised. You will need to make sure that you only reassign a node as vaccinated if it is Susceptible. In the case of random vaccination, it means you are initially vaccinating a proportion (probImmunityInit) of Susceptible nodes (rather than a proportion of all nodes). In the case of targeted vaccination, you are vaccinating only the Susceptible nodes that are in the top (nb_agents_to_immunise) in terms of degree (i.e., you will end up vaccinating fewer than nb_agents_to_immunise nodes).

The effect of these changes will require you to make some changes inside the init() function. First, since you need to write a separate if/else statement (random vs. targeted) for vaccination, you need to move the elif block to a new if/else statement.

elif rd.random() < probImmunityInit:

After you set the state of the node to 'immunised' (whether targeted or random), you need to add and initialise a feature called 'rem_immunity_period' and set it up to immunityPeriod (which captures the immunity period). For example, for any node *a* this would be like:

systemState.nodes[a]['rem_immunity_period'] = immunityPeriod

When updating the step() function, you need to remember two things about this: 1) each time step, you decrease this number by one and 2) once this period is zero, you should change its state to *'Susceptible'*.

Second, you will need to comment out lines of code inside init() function that will do targeted infection. For example,

*nb_agents_to_infect = int(probInfectionInit * float(nb_agents))*

You may also comment out block of code inside step() function starting with:

# residual infection

But make sure to keep this line uncommented: *systemState = nextSystemState*
You don't need to make these commented out if you make the change in the first 3 rows above. Go ahead and re-run the simulation to make sure it is still working fine.

C. The current simulation creates a 2-d grid network. We want to be able to use other types of networks, and we want to pass these networks as parameters. We also want to pass the number of nodes as a parameter to the init() function, and we want to pass another parameter (call it netParam). This is a network specific parameter. For Erdos-Renyi, it is the probability of connection p, for Barabasi-Albert it is the number of edges a new node has m. Finally, we want to pass the probability of initial infection, initial vaccination, whether vaccination is random, and immunity period. The init() function line should look something like:

def init(network, nb_agents, netParam, probInfectionInit, probImmunityInit, randomInitialVaccination ,immunityPeriod):

You need to make necessary changes in the code inside init() function. Test your simulation again (passing an ER network with p=0.1). For the sake of this testing alone, you can add default values for your parameters in init() like:

def init(network =nx.erdos_renyi_graph, nb_agents, netParam=.1, probInfectionInit, probImmunityInit, randomInitialVaccination , immunityPeriod=3):

D. We want to start to minimise the output of this simulation. We don't care about drawing a network or even to plot of number of Susceptible, Infected, and Recovered node at each time step. Instead, we only care about two things: the total number of nodes that are either Susceptible or Immunised at the end of the simulation, and the number of Infected nodes at each time step. We only need the latter in order to make an early stop of the simulation when the network has no Infected nodes (note the *break* command inside run_simulation() function). To do this, you can change collect_statistics() function so that it only returns two parameters: statS and nbI. You may comment out the draw() function (we don't need it). Now make the function run_simulation() return the last element in statS (number of susceptible agents at the end of the simulation). That's the only output we care about from this simulation. Try your simulation again to make sure it is working as expected.

E. It is now time to comment out all printouts in the code. We don't want anything printed to the screen (other than the last element of statS). This includes all code about starting of simulation, which agent got infected, etc. You can also comment out these lines:

```
# if __name__ == "__main__":
#     run_simulation()
```

An important thing to note about this simulation is that the outcome is not always the same each time. Obviously, we use probabilities of infection and recovery. But mainly, because the group of agents we choose to infect at the start are chosen randomly (for example, we set randomInitialInfection=True in B). To avoid having varying results, we will repeat each simulation multiple times. We will calculate the output of each repetition (i.e., the last element of statS), and calculate their means. To make this change, add a loop inside run_simulation() function. Use a parameter called 'rep' to specify the number of repetitions that we want to repeat our simulation for.

F. Almost there! We now need to pass all these parameters inside the run_simulation() with default values. The first line of run_simulation() should now look like:

def run_simulation(nba = 100, maxIter = 10, probI = 0.2, probR = 0.2, probV = 0.0, probI_init = 0.1, probV_init = 0.1, vRandom = True, immPeriod = 3, network = nx.erdos_renyi_graph, netParam=0.1, rep=10):

where nba: number of agents, maxIter: maximum number of iterations, probI is the probability of infection, probR: probability of recovery, probV: probability of immunisation, probI_init: probability of initially infected agents, probV_init: probability of initially vaccinated agents, vRandom: whether vaccination is random or not (Boolean), immPeriod: is the immunisation period, network: network model, netParam: network parameter, and rep: number of repetitions.

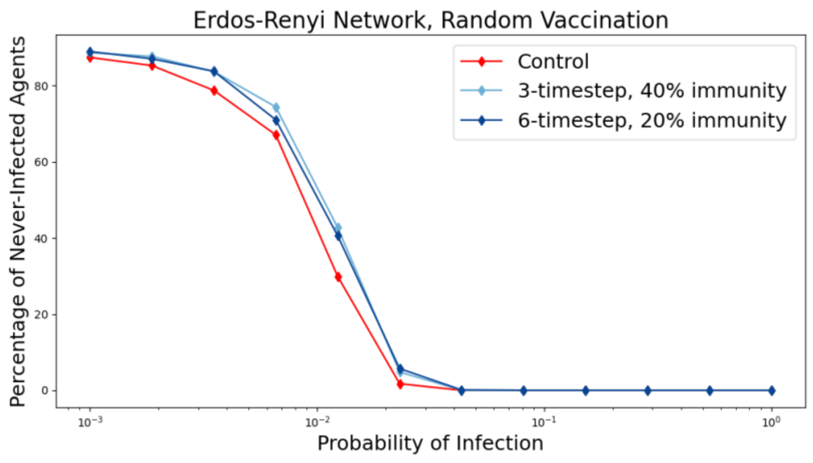
Note here that by setting *probV = 0.0* (by default) we are limiting the 'Immunised' status to those agents who are vaccinated at the start. I did this for the purpose of this exercise and the plots below, but of course you can change that back by setting this variable to a non-zero value, if you decide to use this code for Problem 4.

G. Calculate Percentage of "Never-Infected" Agents as:

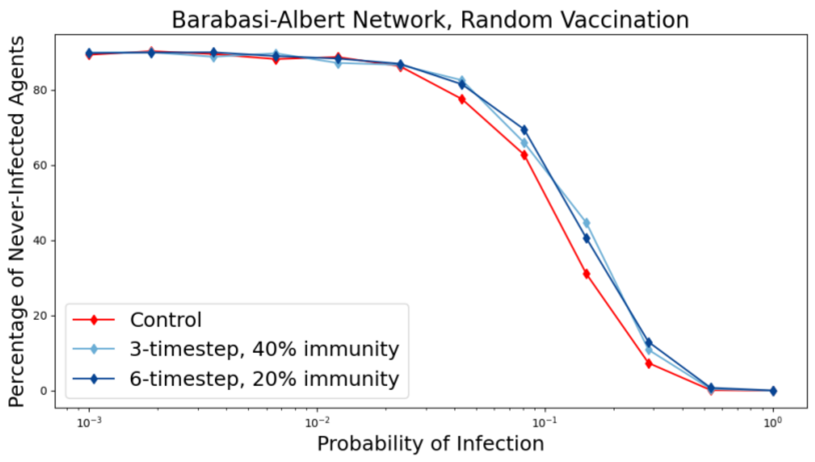
$$\frac{number\ of\ Susceptible\ or\ Immunised\ agents\ at\ the\ end}{number\ of\ agents} * 100$$

H. Now you're ready to produce the following three plots! We will basically test out the impact of two types of vaccinations: 1) expensive but with longer lasting effect (probV_init = 0.2; immPeriod = 6) and 2) cheap but with a shorter lasting effect (probV_init = 0.4; immPeriod = 3). The idea is that we have a fixed budget which would allow us to invest in either buying the more expensive one which last twice the number of days or buying the double amount of the cheaper one (thus vaccinating twice as many people). The impact will be studied on the number of never-Infected (that is, neither Infected, nor Recovered) agents at the end of 10 rounds in two types of networks: Erdos-Renyi and Barabasi-Albert. We will also study random vs. targeted vaccination for Barabasi-Albert network.

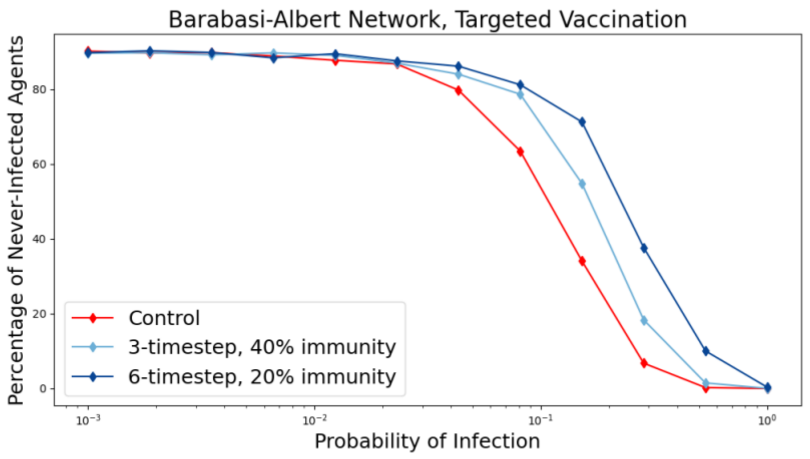
a. Erdos-Renyi Network, Random Vaccination:
   - Use: 500 nodes, default network (Erdos-Renyi)
   - For control: probV_init = 0
   - For 3-timestep, 40% immunity: use probV_init = 0.4
   - For 6-timestep, 20% immunity: use probV_init = 0.2, immPeriod = 6
   - For x-axis, use: x = np.logspace(-3,0,12)
   - Leave all other values as set by default



b. Barabasi-Albert Network, Random Vaccination:
   - Use: 500 nodes, network = nx.barabasi_albert_graph
   - For control: probV_init = 0
   - For 3-timestep, 40% immunity: use probV_init = 0.4
   - For 6-timestep, 20% immunity: use probV_init = 0.2, immPeriod = 6
   - For x-axis, use: x = np.logspace(-3,0,12)
   - Leave all other values as set by default



a. Barabasi-Albert Network, Targeted Vaccination:
   - Use: 500 nodes, network = nx.barabasi_albert_graph, vRandom=False
   - For control: probV_init = 0
   - For 3-timestep, 40% immunity: use probV_init = 0.4
   - For 6-timestep, 20% immunity: use probV_init = 0.2, immPeriod = 6
   - For x-axis, use: x = np.logspace(-3,0,12)
   - Leave all other values as set by default



I. What do you conclude from the three plots above. Why do you think the third plot is different? Explain in one paragraph.

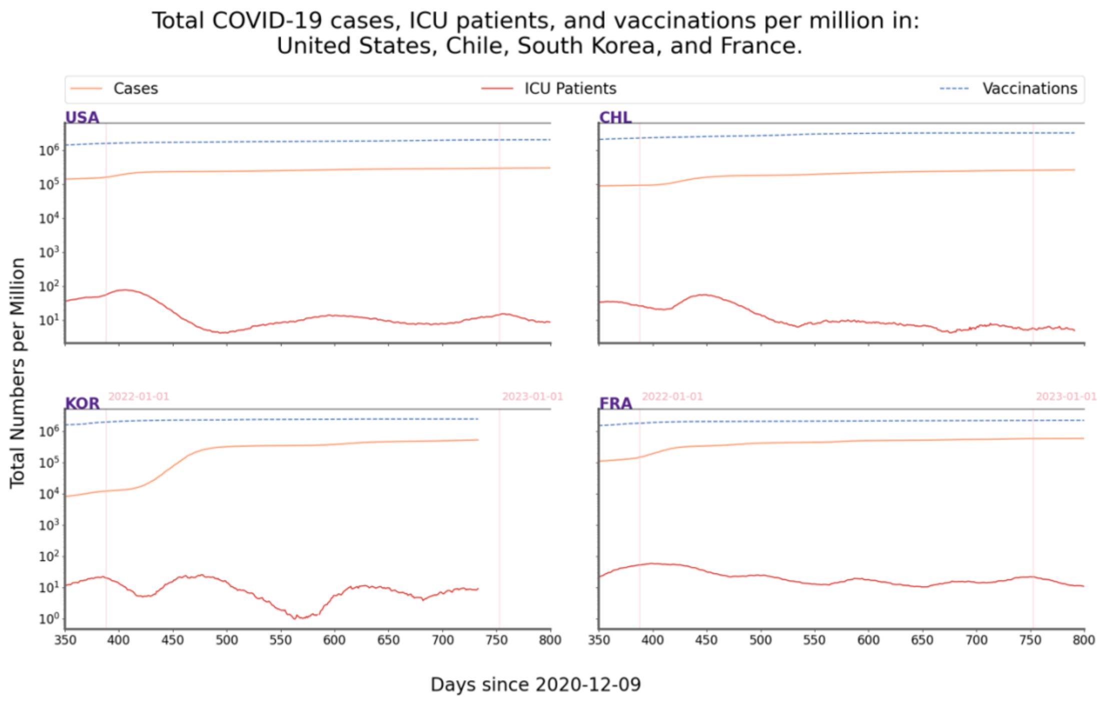
Problem 3 [30 marks]: Mini Project – Warmup and Exploration

In this part, you will get a chance to warmup and explore three different data sets before you choose one of them for the mini project in Problem 4. In each of the following subproblems (3.1, 3.2, and 3.3) you will have two parts: A) reproduce a plot, B) create a new compelling plot that you choose. The goal of these two parts is to get a good chance to explore the data and see if you can come up with interesting ideas with it. This will be helpful for you to make a choice for your mini project in Problem 4. You can re-use the plot you make in part B in the mini project (if you end up choosing that data set). Remember that you need to invest more time in the mini project. You will receive a mark for any progress you make (even if you don't end up reproducing the plot in part A).

Problem 3.1 [14 marks]: COVID-19 – Our World In Data

In this problem, you will use a data set, named *owid-covid-data.csv* (see folder *data*), from "Our World" in Data website.[5] The data is about COVID-19 cases in all countries on every day since 1 January 2020 until 16 March 2023. The website provides many visualisations related to COVID in different countries and over days.[6] The data you will find in data folder was downloaded from this link which has more information about the data.[7]

A. Use the *owid-covid-data.csv* file to 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.



[5] https://ourworldindata.org

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

[7] 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

B. Use the data set to perform a 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. Please provide 1-2 sentences to explain your interesting analysis. Write it in a separate cell inside Jupyter (using Markdown). You are allowed to re-use this analysis in the mini project if you end up choosing this data set.

Problem 3.2 [8 marks]: Formal Inaugural Speeches of the United States Presidents
In this problem, you will use a data set about the formal inaugural speeches of the US Presidents across history (see *inaug_speeches.csv* in *data* folder).

A. Use the *inaug_speeches.csv* file to produce the figure below. The data set contains speeches that you will need to analyse one by one to see how much each one is related to the words "Optimistic" and "Quantitative Reasoning Skills". For this, you will need to write code that utilises *RoBERTa*, a Large Language Models (which is based on BERT). Read more about it here.[8] Working with RoBERTA would require downloading the python package *Transformers*. Downloading it on your machine would require setting up a virtual environment. The process is painful. To make your life easier, I recommend that you use Google Colab (instead of Jupyter Notebook) for this part. If you have a Gmail account, then you have a free access to a Google Colab account (like Google Docs, etc.). If you don't have a Gmail account, create one for the purpose of this project. Once you are in Google Colab, you can install the package *Transformers* and start setting it up.[9]

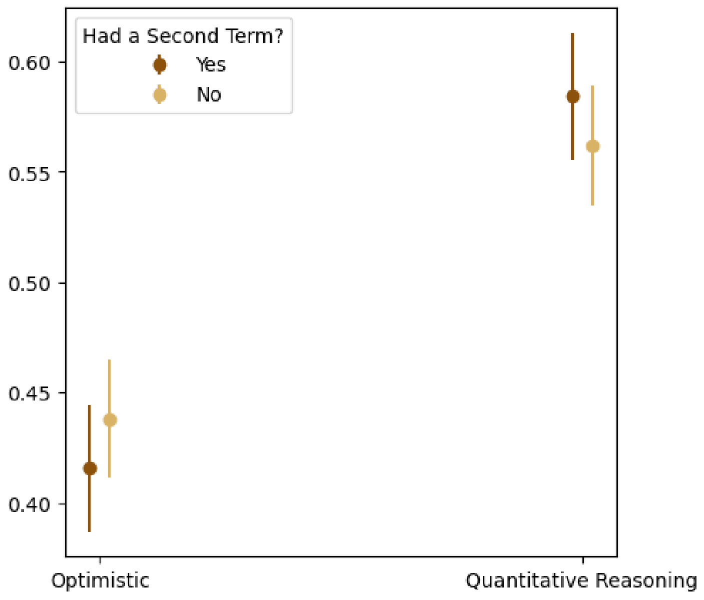
Once you setup, load the data set, for each speech, you will evaluate its relevant to the two following phrases: 'quantitative reasoning skills' and 'optimistic'. Save these values in two columns: QuantLLMQuant, and Optimistic. Save this data frame to a new .csv file named *inaug_speeches_LLM.csv.* Load this file in Jupyter notebook to do the rest. Make sure you enclose the Colab file and the file *inaug_speeches_LLM.csv* in your zip file.

The figure below shows that the relationship between the relevance of the first formal inaugural speech by each US president to these two terms and whether they managed to get elected for a second term is rather weak (but also interesting).

The point estimates are the means, and the error bars are the 95% confidence intervals (mean ± standard error * 1.96).

[8] https://huggingface.co/docs/transformers/model_doc/roberta

[9] See here for more info: https://huggingface.co/roberta-large-mnli



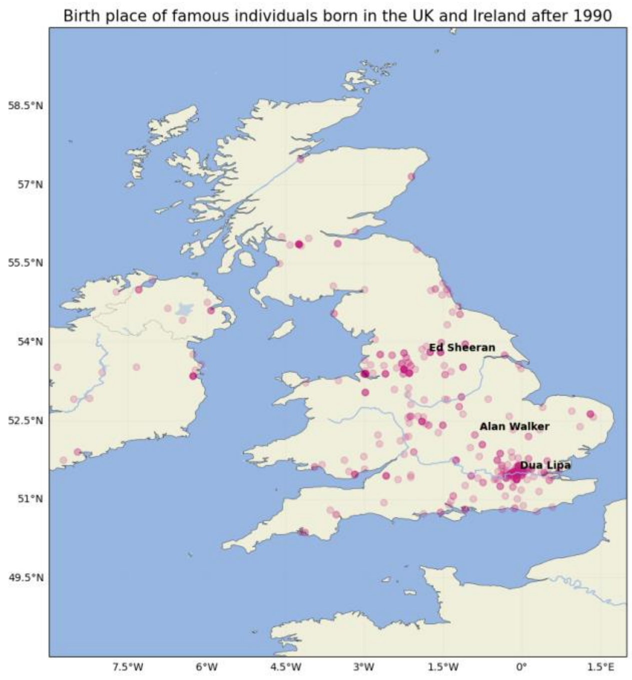
B. Use the data set *inaug_speeches.csv* (or *inaug_speeches_LLM*.csv) to perform a compelling extra analysis (not necessarily using a Large language Model). You will get marks if you create a compelling and interesting visualisation and/or analysis. One plot (or one piece of analysis) is enough. Please provide 1-2 sentences to explain your interesting analysis. Write it in a separate cell inside Jupyter (using Markdown). You are allowed to re-use this analysis in the mini project if you end up choosing this data set.

Problem 3.3 [8 marks]: Famous People – Pantheon

In this problem, you will use a data set about "famous people" (see *person_2020_update.csv* file in data folder). The data contains famous individuals with information about time and location of their birth and death as well as other information like occupation and how "memorable" they are. The data is downloaded from the Pantheon project.[10]

A. Use the *person_2020_update.csv* file to reproduce the plot shown below, as accurately as possible. You are free to use any geographic map library. I used the Python library *Cartopy*. *Cartopy* is the successor for *Basemap* which is covered in a section in the Data Science book we studied from.[11] Since *Basemap* is no longer supported, setting it up would be very painful (but if you already have it on your PC, feel free to use it for this project). You can read more about Cartopy here.[12] Before you plot, make sure you only consider individuals who were born strictly after 1990, and have non-null values for the longitude and latitude coordinates of their birth locations, and are born in the United Kingdom or Ireland.

[10] You can explore data here (using different filters): https://pantheon.world/explore/rankings?show=people&years=-3501,2020
[11] https://jakevdp.github.io/PythonDataScienceHandbook/04.13-geographic-data-with-basemap.html
[12] https://scitools.org.uk/cartopy/docs/latest/getting_started/index.html



B. Use the data set to perform a compelling extra analysis (not necessarily related to geographic maps). You will get marks if you create a compelling and interesting visualisation and/or analysis. One plot (or one piece of analysis) is enough. Please provide 1-2 sentences to explain your interesting analysis. Write it in a separate cell inside Jupyter (using Markdown). You are allowed to re-use this analysis in the mini project if you end up choosing this data set.

Note: *Cartopy* is not usually installed by default with Anaconda. The recommended way to install it is using the following in bash:

```
$ conda install -c conda-forge cartopy
```

If this does not work for you, you can read more about other options here:
https://scitools.org.uk/cartopy/docs/latest/installing.html#installing

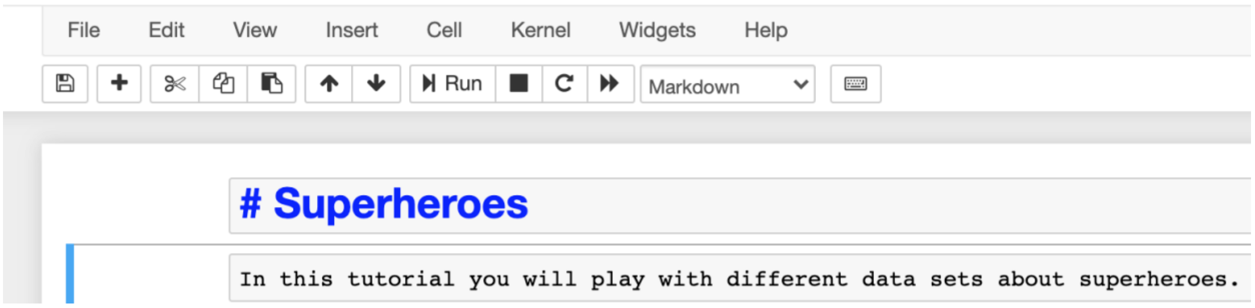
Problem 4 [40 marks]: Mini Project – A Data-driven Blog Post

For this mini project, you will need to write a data-driven blog post.

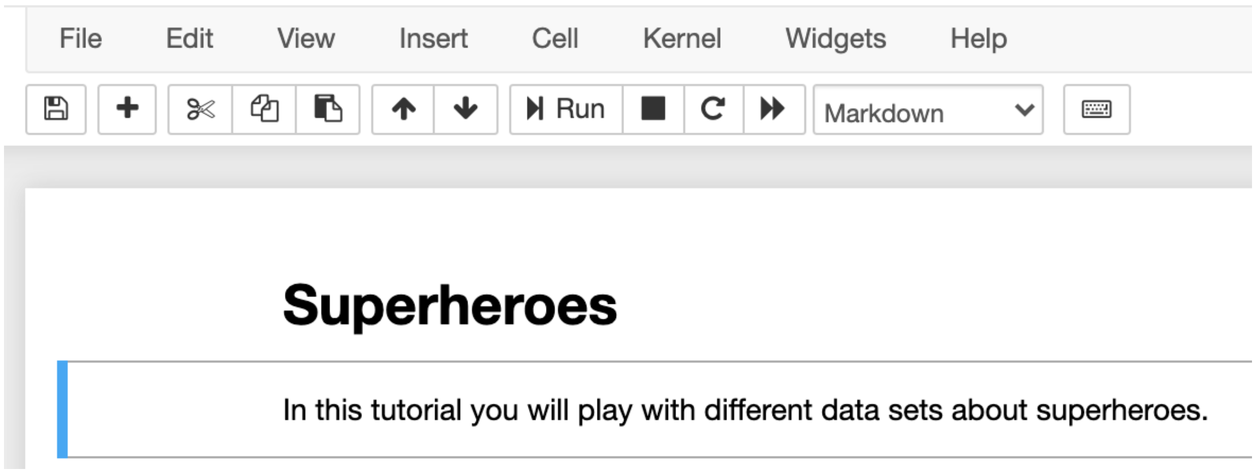
Structure: There are tons of tutorials and examples online. Here's a good guide (you can ignore some parts that are not relevant for this project).
https://playbook.datopian.com/dojo/writing-a-data-oriented-blog-post/#_11-simple-steps-to-create-data-driven-blog-posts
Here are some good examples of data-driven blog posts. Let's start with some fancy ones by The Pudding:
https://pudding.cool/2017/08/the-office/
https://pudding.cool/2017/08/screen-direction/
Check their website, they have some interactive posts as well: https://pudding.cool
You are not expected to write a post to this level of quality (certainly not an interactive one), but you may consider these as the upper limit. Nevertheless, you may find their tutorial useful: https://pudding.cool/process/how-to-make-dope-shit-part-1
To be fair, you may aim to write a post that's more like these ones:
https://buffer.com/resources/social-media-language/
https://datahub.io/blog/automated-kpis-collection-and-visualization-of-the-funnels

Size: Your post should have between 500 – 2000 words, and it should contain 4-7 pieces of output (e.g., plots, tables, summary statistics).

Code/Platform: You will need to write your post in Jupyter Notebook (using Python), where you include Python code, output (plots, tables, statistics, analysis), and text (change cell type to "Markdown" for text). Markdown cells uses markdown language (very simple; look it up). See how the cell looks when it is being edited vs. when it is run below.



```
# Superheroes
In this tutorial you will play with different data sets about superheroes.
```



Superheroes

In this tutorial you will play with different data sets about superheroes.

Data: You will have to choose one topic to write your post about. You will choose one of the three data sets you played with in Problem 3 (hence the exploration task). While you don't have to, you can add new data sets, as long as they're still about the same topic. Alternatively, you may choose to do analysis using simulations that extend the work in Problem 2 (contagion in networks). You can take inspiration from *Our World in Data* COVID-19 figures.[13] If you would like to use data from outside these, that's also acceptable, but you will need to consult with me first. Be default, I will accept any projects done using data from *Our World in Data* website (even if they are not COVID related).[14] The most important thing is that your post should have one topic/theme. For example, don't include plots from Famous/memorable people and COVID-19 together (unless you can come up with a compelling connection).

Submission: In addition to submitting your post as a Jupyter and PDF files (as specified above), you should also consider using Git to track your progress. You will lose few points for not having your project developed with Git version control (Git, not GitHub). If you do this, **please do NOT upload your Git changes to a public repository on GitHub (you will get zero and be reported for misconduct if you do so).** If you decide to use Git/GitHub, you should create a private repository on GitHub, and use https://gitfront.io to share a read-only link to your repository. See this video that explains how to do this.[15]

You may also use HackMD (https://hackmd.io) to create an online (but indiscoverable) link to present your post. To do this, you should make sure you **do NOT "Publish" your post (even after the end of the module)**. You don't need to create an account on HackMD; you can login using your GitHub account. To upload your Jupyter notebook file, you need to download it as a Markdown file. This will create a .zip file which contains one .md file, and a set of your plots. Here's an example of how it will look like on HackMD (this is a tutorial that I gave for BEE1038 last year and this year):
https://hackmd.io/@ZAIixDaKTDe2glG9TRGiwA/rkEwwmyrd

[13] https://ourworldindata.org/coronavirus
[14] https://ourworldindata.org
[15] https://www.youtube.com/watch?v=IejuI40oZ5E

Here' another example by someone else: https://hackmd.io/@linnil1/Sy0p1s9ZX

Audience: You should assume that the audience of your post has some knowledge and interest about the topic, but they have no idea about the data set. Your audience is well educated and have basic knowledge that is comparable to your university's 2nd-year students who never coded before.

Assessment: Your blog post will receive a wholistic mark, based on the following rubrics:
Primary:

- Insightfulness: is the analysis insightful and compelling? Are there some interesting, thought-provoking, and/or surprising findings?
- Soundness: Is the analysis sound? Are all comparisons fair (not comparing apples to oranges)? Was proper filtering done? Are plots used appropriately based on data types (e.g., using bar plots for categorical data points, scatter plots for independent data points, lines for time series)?
- Presentation: Is the narrative coherent? does it all read as a one story? Is it easy to understand the post? Is it easy to follow the ideas and the main story? Are there appropriate (and readable) labels, legends, and captions provided for plots?

Secondary:

- Visual appeal: are the plots visually appealing (beautiful, appropriate colours, non-trivial yet simple plots)? You may use this page to choose nice colours.[16]
- Pre-processing: Has there been an appropriate level of pre-processing of data? Is the code easy to follow? Is the code efficient? Did the student put some effort in cleaning or re-shaping data?
- Practice: Did the student use Git/GitHub or any other version control while preparing this post? Did the student upload their post to HackMD (with indiscoverable link)?

[16] https://colorbrewer2.org