CSCI 140 Project 4: Analyzing Immunization Data

Due 1700 October 30 (5:00 PM EST)

UNICEF maintains a database which houses data sets related to health, development, and other information related to maternal and child health. For this project, we will use immunization data maintained by UNICEF, which contains information on yearly vaccinations administered to children around the world. The vaccine data you have been given is sourced from: https://data.unicef.org/topic/child-health/immunization/.

This data considers vaccines for the following infectious diseases and agents (abbreviation for vaccine shown in all caps): tuberculosis, BCG; diphteria, pertussis, and tetanus, DTP1 and DTP3; meningococcal disease, MCV1 and MCV2; hepatitis B, HEPBB and HEPB3; Haeomphilus influenza, HIB1; polio, IPV3 and POL3; pneumococcal disease, PCV3; rubella, RCV1; rotavirus, ROTAC; and Yellow Fever Virus, YFV. Data is categorized as the percentage of children vaccinated, and is provided both globally and regionally (e.g. East Asia and Pacific, Middle East and North Africa, etc.).

You will create functions to process and visualize this data, and will write a main program that performs data QC and makes use of your functions. You have been given four files:

- **vaccine_data.csv** is a comma-delimited text file which contains all of the data
- **Project_4.py** is a skeleton file where you will write your functions, import lines have been provided for you, but you must write the def lines according to the specifications below
- **Project_4_Main.py** is a skeleton file which will contain your main program

In order to complete this assignment you must have functional versions of the following packages installed: pandas, numpy, matplotlib, seaborn.

BE AWARE:

In order to receive full credit, you must use pandas objects and the pandas/seaborn/matplotlib libraries to make plots and edit data when possible. Implementations which write code to take the place of pandas functions/methods and/or that use other imported objects may not receive credit. Plots made using other softwares will not be accepted for credit. Manipulations to data performed without corresponding code (e.g. opening data in Excel and editing it) will receive no credit.

Keep these instructions open on your Desktop while working and refer to them often. They will tell you exactly how to complete the assignment.

Part One: Data QC

The first part of this project requires reading in the file **vaccine_data.csv** and reformatting some of the data. It will be helpful for you to look at the data frame after each step. Ask if you don't know what this means.

You have been provided with code in the main program to correct. You must edit these lines in place. **You may not add any new lines of code or alter these lines dramatically - this means you must use the pandas functionality and should not add other functions, loops, or list comprehension. All of the changes you need to make are relatively minor.**

The code you have been given to debug will do the following:

-Read the data in from the file to a pandas data frame called **vaccine**, consider that there are no column names in the raw data

-Name the columns: 'Region', 'Vaccine', 'Year', and 'Percentage' (the quotes indicate that these are strings, there should not be quotes in the actual text of your column names)

-Update region names to remove spaces and ampersands, for example, 'Eastern & Southern Africa' should be changed to 'Eastern_and_Southern_Africa'

-Change the type of the Year column to a string

-Create a new column named Description that contains the full name of the pathogen or disease that the vaccine is administered for; you MUST use the dictionary provided to accomplish this task. This column will end up as the last column in the data frame – that is fine, you do not and should not move it.

If you are doing this in a notebook, we strongly suggest putting each line of code in its own cell. That way you can look after each step to see if things worked the way they should have. If you put the code all in one block, it can be very hard to figure out where the errors are originating.

Part Two: Functions

For this section of the project you will create 2 functions to use with your processed data frame or other data frames in a similar format.

Function 1: *make_subset(df, region = None, vaccine = None, year = None)*

The subset function returns a data frame that is a subset (or a copy) of the data frame passed in by the user as the required argument **df**. The optional arguments **region**, **vaccine** and **year** allow the user to specify which subset of the data they are looking for. These arguments all have a default value of None. The user may specify values for all three arguments, for only two of the arguments, or for a single argument. If the user specifies nothing for these arguments, you should return a COPY of the original data frame. Do not return the original data frame. If you don't understand the difference, please ask us and clarify.

Here are a few examples. So that you can clearly see what is happening, these examples make use of a small set of data. The data frame passed in for **df** in all of these examples consists of the following data:

	Region	Vaccine	Year	Percentage	Description
0	Western_Europe	PCV3	2009	27	pneumococcal disease
1	Latin_America_and_Caribbean	HIB3	2001	72	Haemophilus influenza
2	North_America	MCV2	2005	79	meningococcal_disease
3	Western_Europe	DTP1	2008	98	diphteria_pertussis_tetanus
4	Middle_East_and_North_Africa	DTP3	1997	87	diptheria_pertussis_tetanus
5	East_Asia_and_Pacific	BCG	1992	90	tuberculosis
6	North_America	MCV1	1995	89	meningococcal_disease
7	Global	RCV1	1982	4	rubella
8	West_and_Central_Africa	HIB3	1991	0	Haemophilus influenza
9	North_America	POL3	2009	93	polio
10	West_and_Central_Africa	RCV1	1981	0	rubella
11	East_Asia_and_Pacific	MCV1	1998	83	meningococcal_disease

Notice that the columns are not sorted in any particular way. You should not assume the data is sorted when writing your subsetting code.

Please note that these examples are not exhaustive, i.e. they don't show every possible case. The returned data frames shown are shown in the view from Jupyter notebook. When you run your code from the command line, if you explicitly print the results, they will not be formatted neatly like the examples shown here.

Example 1:
df is the data frame from the introduction, **vaccine** is ['PCV3', 'RCV1', 'HEPB3']; function returns a data frame with the following rows:

	Region	Vaccine	Year	Percentage	Description
0	Western_Europe	PCV3	2009	27	pneumococcal disease
7	Global	RCV1	1982	4	rubella
10	West_and_Central_Africa	RCV1	1981	0	rubella

Example 2:
df is the data frame from the introduction, **region** is ['West_and_Central_Africa'], **year** is ['1981', '1987']; function returns a data frame with the following rows:

	Region	Vaccine	Year	Percentage	Description
10	West_and_Central_Africa	RCV1	1981	0	rubella

Example 3:
df is the data frame from the introduction, **vaccine** is ['PCV3', 'HEPB3']; **region** is ['West_and_Central_Africa'], **year** is ['1981', '1987']; function returns an empty data frame:

Region	Vaccine	Year	Percentage	Description

NOTE: if you have written your subsetting properly, you do not check for the case where the inputs don't match any of the rows separately – this should happen automatically without you writing any additional code.

Key points:

- You can assume that the user will pass in inputs of the correct types and formats. df will always be a Data Frame, region (if passed in), vaccine(if passed in), and/or year (if passed in) will always be lists of one or more strings
- If you are reading in from a file anywhere in this function, you are doing it wrong.
- The basis of this function is subsetting a data frame. We have discussed this. If you are looking up pandas methods to append data frames to each other, pivot the data frame, or complicated code we didn't talk about, you are probably doing it wrong.
- It is fine to hard code in the column names, like Region for example - that might feel funny to you, but you will need to do it
- If the user passes in a combination of arguments for which there are no rows that meet all of the criteria, you should return an empty data frame.
- Remember that if the user calls the function with no arguments for **vaccine, year,** or **region**, your function should return a COPY of the data frame.
- Don't overcomplicate this. Our reference implementation is 8 lines long (not including the def line). We expect your code to be simplified and efficient. If you are creating separate cases for each possible combination of arguments, your code is overly complicated.
- Curious how to make a copy of a data frame? Use the copy method.

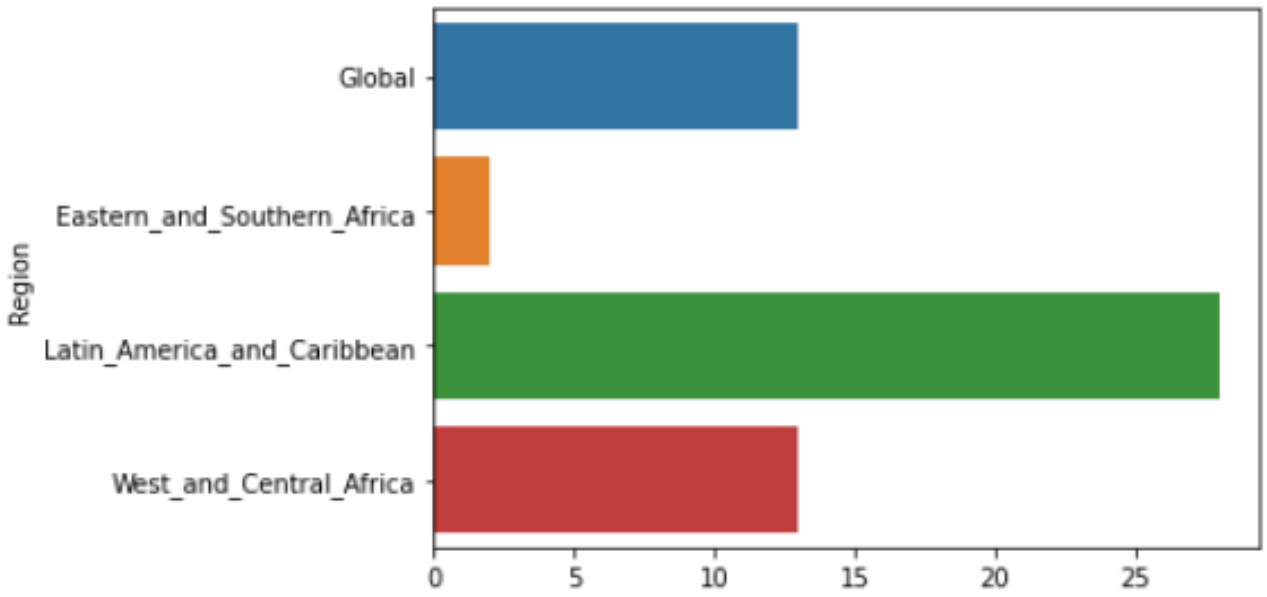
Function 2: *make_plot (series_object, title ='', bar = True)*

Write a function called **make_plot** that takes one required argument: **series_object**, a pandas Series that contains numeric (float or integer) values. The optional argument **title** is a string that is a title for the plot, and the optional argument **bar** determines what type of plot will be drawn as explained below. The **make_plot** function returns a plot that visualizes the data in **series_object**. The index of the Series should be reflected on the x or y axis as appropriate (see below). The type of plot differs by the value of the argument **bar**. When **bar** is True, the function produces a barplot, when **bar** is False, it produces a line graph.

Suppose you have a pandas Series where the Region is the index and the percentage is the data:

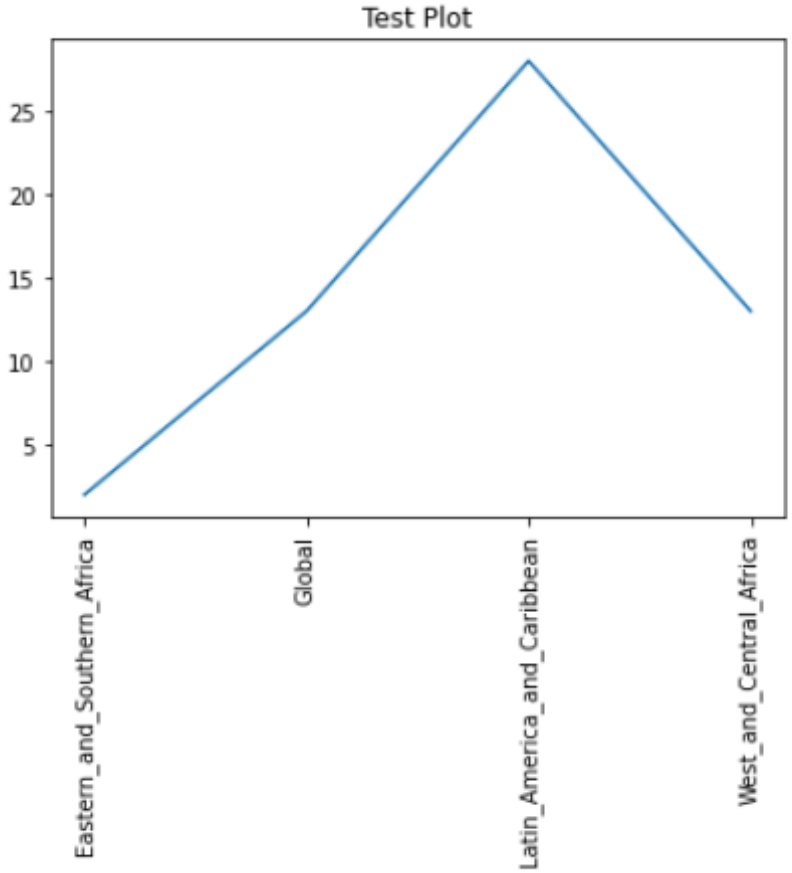
```
Region
Global                          13
Eastern_and_Southern_Africa      2
Latin_America_and_Caribbean     28
West_and_Central_Africa         13
Name: Percentage, dtype: int64
```

We get the following plot when call make_plot function with **series_object** as the above series and **bar** is True:



Notice that the barplot is horizontal. You are not responsible for changing or moving the y-axis title (Region in this case).

We get the following plot when call **make_plot** with **series_object** as the above series and **bar** is False and **title** is `'Test Plot'`:



Notice that the labels on the x-axis are rotated 90 degrees. This should always happen for the line plot regardless of what the input is.

Key points:

- **series_object** is a Series, not a data frame. You can assume the data in **series_object** is numeric.
- You can assume the inputs are of the correct types (e.g. **title** will be a string etc.)
- All of the code for making the different plots are is in your notes/readings verbatim. Don't spend hours searching the internet for something else. Use the course materials.
- When **bar** is True, make a barplot, when **bar** is False, make a lineplot.
- Don't forget to have your function return the plot!
- Plots will show up in the notebook but not on the command line and that is OK. You can make them show up on the command line but we don't want you to do that otherwise we can't test your code remotely. If you add in code to make them show up, please comment it out before you turn your project in.

Part Three: Putting it all together

For this part of the assignment you will add lines to the main program to use the functions you wrote in Part 2 on the re-formatted data frame you created in Part 1. The lines of code you write for Part 3 will follow (i.e. go under) the lines of code from Part 1. ASK IF YOU DO NOT UNDERSTAND THIS – YOUR CODE WILL NOT WORK PROPERLY OTHERWISE.

Use your functions. You should not repeat code from the functions in the main program. Use good style – don't pass in default arguments, and don't provide unnecessary arguments. Some tasks will require additional code. This is indicated in the instructions. Read carefully.

- Create a data frame called BCG_2019 that contains the rows from the **vaccine** data frame that correspond to BCG vaccine for the year 2019. This will include all available regions.

- From the data frame you made above, create a pandas Series called BCG2019_Series that has the data in the Region column as the index and the data from the Percentage column as the values. The easiest way to do this is to create a new data frame with Region as the index and then select the Percentage column.

- Create a barplot for the percentage outreach (Percentage) of BCG vaccine by region in 2019.

- Create a data frame called DTP1_Years that contains the rows from the **vaccine** data frame that correspond to DTP1 vaccinations in the East Asia and Pacific region. This will include all available years.

- From the data frame you made above, create a pandas Series called DTP1_series that has the data in the Year column as the index and the data from the Percentage column as the values. The easiest way to do this is to create a new data frame with Year as the index and then select the Percentage column.

- Create a line plot of the data store in DTP1_series with the title: DTP1 Vaccinations by Year in East Asia and Pacific Region

SUBMISSION EXPECTATIONS

Project_4.py: Your function code goes in this file. You have been given a skeleton file that contains the appropriate import lines. You must fill in the def lines and they must match the specifications exactly (HINT: look at the title for each section). Changing default arguments, the order of arguments, the number of arguments etc. is not permitted.

Project_4_Main.py : Your correction of data QC of the .csv file in Part 1 and the additional main program in Part 3 go in this file.

Project_4.pdf: A PDF document containing your reflections on the project. You must also cite any sources you use. Please be aware that you can consult sources, but all code written must be your own. Programs copied in part or wholesale from the web or other sources or individuals will result in reporting of an Honor Code violation. If your code contains structures not used in class, be prepared to explain the structure in a meeting with one of the instructors to receive credit for your work.

POINT VALUES AND GRADING RUBRIC

Part 1: Data QC (5 pts)

Part 2: make_subset (8.5 pts), make_plot(5 pts)

Part 3: Main program (6 pts)

Writeup (0.5 pts)

Please note that you will be graded on style in terms of using pandas methods where appropriate and writing compact code as needed and specified in the instructions. This doesn't mean you can't use multiple lines or include conditionals, but rather that if something can be done with a pandas method in one line, you shouldn't be doing it with a for loop in many lines.

Grade Level	Execution	Correctness and Algorithms	Formatting	Style and References
A	Code executes without errors, requires no manual intervention	Code correctly implements all algorithms, passes all test cases (with possible exception of extremely rare case as applicable), and meets all required functionality	Inputs and outputs formatted as per the specification both in terms of content and type, may be minimal formatting issues (spacing, punctuation, etc.); correct file formats and names	Code uses current structures and modules discussed in class; meaningful variable names; commented as appropriate; any references cited in write-up
B	Code executes without errors, may require minimal manual intervention such as change of an import line	Meets all required functionality with exception of minor/ rare test cases (as applicable)	Inputs and outputs formatted as per the specification both in terms of content and type, some minor formatting issues may be present (e.g. incorrect string text), correct file formats and names	Code uses current structures and modules discussed in class; meaningful variable names; commented as appropriate; any references cited in write-up
C	Code does not execute without minor manual intervention such as correcting indentation or terminating parentheses or a single syntax error	Code contains logical errors and fails subset of both rare and common test cases, overall algorithmic structure reflects required functionality but implementation does not meet standards	Inputs and outputs have major formatting issues, files incorrectly named but file formats correct	Code uses current structures and modules discussed in class; meaningful variable names; commented as appropriate; references cited in write-up
D	Code does not execute without major manual intervention such as changing variable names, correcting multiple syntax errors, algorithmic changes	Code contains major logical errors and fails majority of test cases, lack of cohesive algorithmic structure, minimal functionality	Inputs and outputs have major formatting issues, files incorrectly named but file formats correct	Code uses some structures and modules from class, but otherwise largely taken from outside sources with citations; variable names and code structure hard to decipher
F	Code requires extensive manual intervention to execute	Code fails all or nearly all test cases and has no functionality to solve the proposed problem	Inputs and outputs have major formatting issues, files in incorrect formats which cannot be graded	Code uses structures and modules from outside sources only with no citations; variable names and code structure hard to decipher; code written in other programming languages or Python 2.7