CSCI 141 Project 4: Analyzing Immunization Data

Due December 2nd, 2022 by 2200 hrs(10:00 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 the data and will write a main program that performs data QC and makes use of your functions. You have been given three 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.

BE AWARE: Your project submission (Project_4.py and Project_4_Main.py) will be graded on style in terms of using pandas methods where appropriate and writing compact code as needed and specified in the instructions. In order to receive full credit, you must use pandas objects and the pandas/numpy libraries to edit data when possible. This doesn't mean you can't use multiple lines or include conditionals, but rather that if something can be done with pandas function or method, you shouldn't write loops to iterate over data frames and series even if you can get the expected output. Implementations which write code to take the place of pandas functions/methods and/or that use other imported objects may not receive credit. Manipulations to data performed without corresponding code (e.g. opening data in Excel and editing it) will receive no credit.

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 should do the following:

(1) 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

(2) 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)

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

(4) Change the type of the Year column to a string

(5) 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.

(6) Drop any rows with missing data (ANY missing data)

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: Function

For this section of the project you will create 1 function to use with your processed data frame or other data frames in a similar format.

make_subset(df, region = None, vaccine = None, year = None, additive = True)

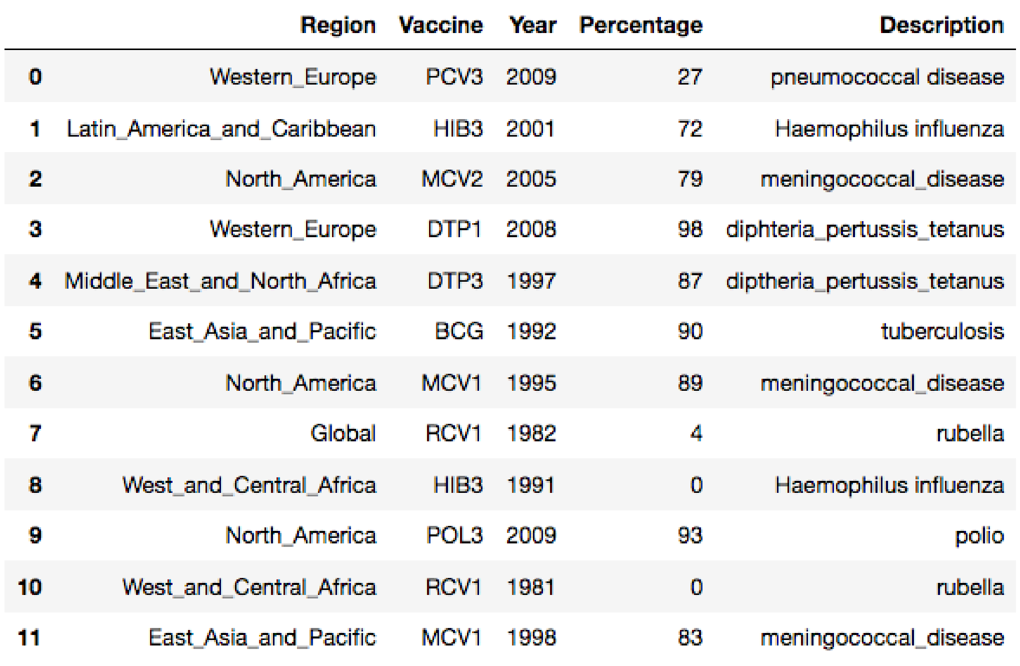
This function returns a data frame that is a subset (or a copy) of the data frame passed in by the user as the required argument **df**. This data frame has at least three columns representing the region, vaccine, and year. The data types of the Year column are strings.

The optional arguments **region**, **vaccine,** and **year,** which will be lists of one or more strings if passed in, allow the user to specify which subset of the data they are looking for. These arguments all have a default value of None. The optional argument **additive** is a Boolean.

When **additive** is True, for the optional arguments **region**, **vaccine** and **year**, 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 all these three 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.

When **additive** is False, the user must specify values for all three arguments **region**, **vaccine,** and **year**.

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:

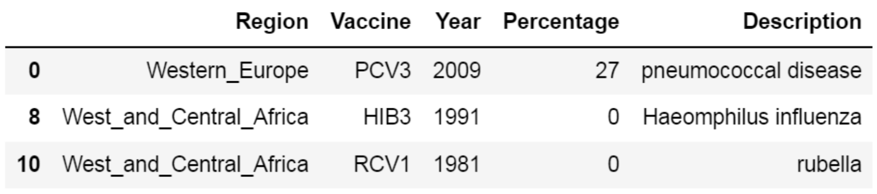


	Region	Vaccine	Year	Percentage	Description
0	Western_Europe	PCV3	2009	27	pneumococcal disease
1	Latin_America_and_Caribbean	HIB3	2001	72	Haemophilus influenza
2	North_America	MCV2	2005	79	meningococcal_disease
3	Western_Europe	DTP1	2008	98	diphteria_pertussis_tetanus
4	Middle_East_and_North_Africa	DTP3	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 the 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.

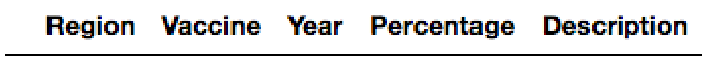
Example 1: **df** is the data frame from the introduction, **additive** is False, **vaccine** is ['PCV3', 'HEPB3']; **region** is ['West_and_Central_Africa'], **year** is ['1981', '1987']; function returns a data frame with the following rows:



	Region	Vaccine	Year	Percentage	Description
0	Western_Europe	PCV3	2009	27	pneumococcal disease
8	West_and_Central_Africa	HIB3	1991	0	Haeomphilus influenza
10	West_and_Central_Africa	RCV1	1981	0	rubella

When **additive** is False, we treat **region**, **vaccine** and **year** as OR requirements. Rows which meet any of the criteria will be part of the output data frame, i.e., the output data frame will include any rows where the vaccine is PCV3 or HEPB3 **OR** where the region is West_and_Central_Africa **OR** where the year is 1981 or 1987.

Example 2: **df** is the data frame from the introduction, **additive** is True, **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

When **additive** is True, we treated **region**, **vaccine** and **year** as AND conditions, i.e., the output data frame will only include rows where all these three - the region, the vaccine, and the year - meet the conditions. In this example, the user passes in a combination of arguments for which there are no rows that meet all of the criteria; the function returns an empty data frame.

Example 3: **df** is the data frame from the introduction, **additive** is True, **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 4: **df** is the data frame from the introduction, **additive** is True, **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

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, **additive** will always be a Boolean. You can also assume that the user will pass in values for all three arguments - **region**, **vaccine**, and **year** - when **additive** is False.
- If you are reading in from a file anywhere in this function, you are doing it wrong.
- You can hardcode the column names, like Region, for example.
- 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.
- 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 7~13 lines long (not including the def line). We expect your code to be simplified and efficient. If you create separate cases for each possible combination of arguments, your code is overly complicated.
- **You will be graded to some degree on programming style – specifically: parsimony in terms of not repeating the same EXACT line(s) of code several times and using pandas methods and structures whenever possible.**
- Curious about how to make a copy of a data frame? Use the copy method.

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.

You must use your function to accomplish the following tasks. You should not repeat code from the function. 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!

1. When **additive** is True, create a data frame called BCG_2019 that contains the rows from the **vaccine** data frame that correspond to BCG vaccinations for the year 2019. This will include all available regions.
2. From the data frame you made above, create and print 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.
3. When **additive** is False, 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 or for the year 1980.
4. From the data frame you made above, create and print 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.

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. 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 mentioned in class or readings, please include the following in your write-up:
If it's a method:
(1) What does this method do? What should be its input and output?
(2) Why do you use this new method instead of the way we learned in class or readings or labs? What is the advantage?
If it's not a method, but a concept, e.g., recursive data structures,
(1) Apply it to one of the examples in lecture notes.
(2) For recursive data structures, please specify base case(s) and recursive case(s).
(3) Why do you use it instead of the way we learned in class or readings or labs? What is the advantage?

You can expect significant grade penalties if you get worse time and/or space complexity by introducing anything that is NOT mentioned in class or readings or if the program can be written in a more concise way using anything we learned.

POINT VALUES AND GRADING RUBRIC

Part1: Data QC (30 pts)
Part2: make_subset function (27 pts)
Part3: Main program(30 pts)
Writeup (2.5 pts)
Autograder (10.5 pts)

RUBRIC FOR MANUAL GRADING

Grade Level	**Appropriateness**	**Style**	**Code Efficiency**
A	Code uses current structures and modules discussed in class; meaningful variable names; commented as appropriate; any references used cited in write-up	Follows Python style guidelines we have discussed to include appropriate naming of variables and spacing; Proper use and naming of constants	Only contains structures necessary to accomplish the task at hand; Uses the most concise implementation possible
B	Code uses structures and modules discussed in class; variable names meaningful; references cited in write-up	Largely follows Python style guidelines but may be a few minor deviations	Only contains structures necessary to accomplish the task at hand; Implementation is generally concise
C	Code uses structures and modules discussed in class with minor additions from outside sources (e.g. string formatting) appropriately cited; most variable names meaningful	Several deviations from Python style guidelines; constants not used properly	Contains a small number of unnecessary structures such as casts that are not needed; Implementation somewhat concise
D	Code uses some structures and modules taken from outside sources with citations; variable names and code structure hard to decipher	Only loosely follows Python style guidelines	Contains a large number of unnecessary structures such as unused variables and casts; Implementation somewhat concise but could be improved, code may be hard to follow
F	Code uses structures and modules largely from outside sources only; variable names hard to decipher; code written in other programming languages or Python 2.7	Code appears to have been written with no consideration for style	Contains many unnecessary structures; Code is difficult to follow and overly verbose