UPLUS EDUCATION
优加教育
Address: Lv1/868 Shepperton Rd, East Vic Park

Year 8

Science

Term 3 Week 7

Name: _____

Date: __/__/_____

Data Science Across the Disciplines: Review

Now that you have (supposidly) completed your tasks from T3W7, your data should be cleaned and ready to be analysed. Your cleaned data should present unique unit patterns as well as return the number of these patterns. Here are some additional tasks for you to do to find out some metrics within your data.

Examining the range of disciplines

For this part, it is left to you to think about the best way to break the problems down. You won't want to write new code for each different discipline, so think about how you can use structured programming to efficiently answer the questions.

Task 1 Question 1:

What proportion of study patterns contain:

- a maths unit?
- a stats unit?
- a medicine unit?
- a psychology unit?
- a philosophy unit?
- a computer science unit?
- a business unit?
- a law unit?
- a music unit?
- etc

Task 1 Question 2: How many different disciplines are studied altogether (for the purposes of this question assume the first four characters of the unit code represent the discipline.)

Task 1 Question 3:

Print a list of each unit that is being studied, and how many students are studying it, in order from the unit with the most students to the unit with the least.

Covid Tests

President Trump has repeatedly said that the US tests more than any other country in the world by far, and sometimes more than all the other countries put together.

In this week's work, we'll investigate the accuracy of one aspect of these claims, the per capita test rates.

Reading in the data

For this week's data, we'll use COVID testing data from the 29th July.

Task 2 Question 1:

Read the COVID data and visually examine it.

Data Conversion and Cleaning

Task 2 Question 2:

Read the first 5 lines again. This time use the split method to turn each line into a list before printing it out. Remember to remove the new line character at the end of each row. This can be accomplished using replace or strip.

Your output should start like this:

```
['Entity', 'Code', 'Date', 'Daily tests per thousand people (7-day s
moothed) (tests per thousand)\n']

['Argentina', 'ARG', '"Feb 18', ' 2020"', '0']

['Argentina', 'ARG', '"Feb 19', ' 2020"', '0’]
```

You may have noticed another problem, caused by the fact that the dates include commas. In fact, you may conclude that with this date format, the choice of a comma as the delimiter (separator) was not a particularly good one, and an alternative such as tab separated (tsv) would have been better choice for these data.

Nevertheless, it is not ambiguous, because commas that are not intended as delimiters only appear within double quotes. The 'user' (our code in this case) is expected to take the quotes into account when splitting the lines.

There are many ways to deal with dates, but for now, we can just remove the comma and the quotes, since neither provide us with any information. The fields within the quotes (month, day and year) can be distinguished by their order (the comma is just for human consumption) and the quotes are redundant since it is a text file and all the fields will be read as strings.

Change your code so that it has a preprocessing step before it splits the lines. For each line after the header line your preprocessing step should:

- find the positions of the two double quotes
- replace the comma between the quotes with a space
- replace the old date with the new date without a comma
- print out an error message if the line doesn't have double quotes

We will break this down into smaller steps.

Task 2 Question 3:

Print each line (of the first 5, other than the header, and with the whitespace removed) followed by the indices of the two double quotes:

```
Argentina,ARG,"Feb 18, 2020",0
14 27
Argentina,ARG,"Feb 19, 2020",0
14 27
```

```
Argentina,ARG,"Feb 20, 2020",0

14 27

...
```

Task 2 Question 4:

Next, print the line followed by the date string with the comma removed from the date string:

```
Argentina,ARG,"Feb 18, 2020",0

"Feb 18 2020"

Argentina,ARG,"Feb 19, 2020",0

"Feb 19 2020"

Argentina,ARG,"Feb 20, 2020",0

"Feb 20 2020"
```

Task 2 Question 5:

Next, print the original lines with the old date field replaced by the cleaned date field:

```
Argentina,ARG,Feb 18 2020,0

Argentina,ARG,Feb 19 2020,0

Argentina,ARG,Feb 20 2020,0
```

Task 2 Question 6:

Now that we have this working, we don't want this preprocessing step 'muddying' up our code, so let's put it in a separate function.

- Write a function clean(data_row) that takes an argument, a string

- Function clean(data_row) strips any unnecessary whitespace characters from the ends
- If it contains a string in double quotes, strips the quotes and the comma between them
- If it doesn't contain any quotes (this will be the header line), it strips everything in the line from the space before the first parenthesis

If called with the following code:

```
with open(DATA,'r') as file:
    for i in range(5):
        print(clean(file.readline()))
```

the output should start like this:

```
Entity,Code,Date,Daily tests per thousand people
Argentina,ARG,Feb 18 2020,0
Argentina,ARG,Feb 19 2020,0
...
```

Homework

Continue on with the work to further clean and investigate this data.

Task 3 Question 1:

Next, rather than printing the lines, store them all in a list (of lists).

Define a variable as an empty list (). Write code that cleans and splits the entire input into lists, and appends them to data_lists.

For example, the following should print the first five rows as a list of lists:

```
print(data_lists[:5])
```

```
[['Entity', 'Code', 'Date', 'Daily tests per thousand people'], ['Argentina', 'ARG
', 'Feb 18 2020', '0'], ['Argentina', 'ARG', 'Feb 19 2020', '0'], ['Argentina', 'A
RG', 'Feb 20 2020', '0'], ['Argentina', 'ARG', 'Feb 21 2020', '0']]
```

How many entries (lines of data) are there in the file?

Checking our cleaning so far

We now have a tidy list of lists, each with the four fields. Or do we?

With big data it may not be possible to manually look at every entry to see if we've accounted for every possibility. We should try to make our cleaning or preprocessing as general as possible so that we catch unexpected variations or bad data.

In practice, we often have to make some assumptions about the data. However we should endeavour to test these. In this case, we've assumed that the patterns we see at the start of the file continue through the file. So let's check that assumption.

Task 3 Question 2:

Write code that checks whether all your entries have 4 fields.

If not, why not? What have we missed?

Hint: Print out the first row where there are not 4 fields. Print out the number of that row, open the data file in Execel, and have a look at the data in that row. What do you find?

General vs Specific

The variations you see in the data are not unusual - remember that this is the collated official government data, its not an 'exercise'. It is *exactly the kind of thing you would deal with as a working Data Scientist*.

We'll need to come back to some differences in the content of the data, but for now let's focus on the formatting. Or, more precisely, *transforming* the data from the format in which it is provided to a format that is suitable for our use.

Task 3 Question 3:

Alter your code to do more cleaning as necessary so that it is transformed into a list of lists, each with four fields.

You should try to make your code as general as possible. This means that, rather than just adjust for the specific case, think about *patterns*.

For example, we have seen that the data format uses double quotes around fields containing the delimiter (a comma in this case). If the data is not corrupt (which is an assumption) therefore, we would expect the double quotes to always occur in pairs. By focussing only on dates, we have been more *specific* than we need to be, so we may miss other cases. A more *general* solution will assume that the same pattern may occur in other fields.

Ensure you re-test your code after any changes you make to ensure it satisfies the requirements. (You might find it useful to write them all down.)

Task 3 Question 4:

Complete the function get_cleaned_lists (filename) so that it returns a list of lists, each containing the fields from the data file, with quotes, commas, and leading/trailing whitespace characters removed, any parenthesised text removed, and the daily tests ratio (the last column of the data) as a float.

If you've made here, congratulations, you have reached the end of materials for this week. Here you would have learned about the troubles and inconsistencies of real world data as well as gained skills with data cleaning. Next week we will examine how to read and find insights within the data