assignment1-answers

October 2, 2019

1 Homework 1 - SOLUTION

Please complete this notebook by filling in the cells provided.

```
In [9]: # Don't change this cell; just run it.
        import numpy as np
        from datascience import *
```

Reading: - Textbook chapters 1, 2, 3, 4 and 5.

For all problems that you must write our explanations and sentences for, you **must** provide your answer in the designated space. Moreover, throughout this homework and all future ones, please be sure to not re-assign variables throughout the notebook! For example, if you use `max_temperature` in your answer to one question, do not reassign it later on.

Directly sharing answers is not okay, but discussing problems with the course staff or with other students is encouraged. Refer to the syllabus page to learn more about how to learn cooperatively.

You should start early so that you have time to get help if you're stuck.

1.1 1. Scary Arithmetic

An ad for ADT Security Systems says,

> "When you go on vacation, burglars go to work [...] According to FBI statistics, over 25% of home burglaries occur between Memorial Day and Labor Day."

Do the data in the ad support the claim that burglars are more likely to go to work during the time between Memorial Day and Labor Day than at other times? Please explain your answer.

From Memorial Day to Labor Day is a bit over 25% of the year. This means that over 25% of home burglaries during this time span would not necessarily be abnormal. Therefore the data do not support the claim that burglars are more liklely to go to work during this time span than other time spans.

1.2 2. Characters in Little Women

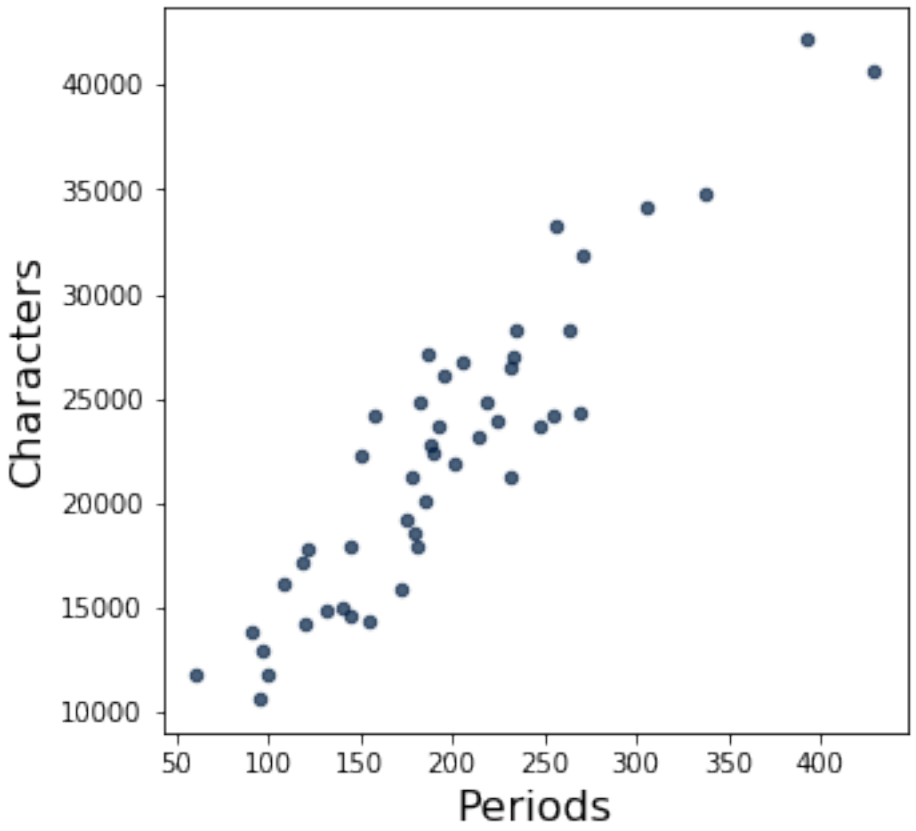
In lecture, we counted the number of times that the literary characters were named in each chapter of the classic book, Little Women. In computer science, the word "character" also refers to a letter, digit, space, or punctuation mark; any single element of a text. The following code generates a scatter plot in which each dot corresponds to a chapter of Little Women. The horizontal position

of a dot measures the number of periods in the chapter. The vertical position measures the total number of characters.

```
In [3]: # This cell contains code that hasn't yet been covered in the course,
        # but you should be able to interpret the scatter plot it generates.

        from datascience import *
        from urllib.request import urlopen
        import numpy as np
        %matplotlib inline

        little_women_url = 'https://abby621.github.io/cs1070_textbook/data/little_women.txt'
        chapters = urlopen(little_women_url).read().decode().split('CHAPTER ')[1:]
        text = Table().with_column('Chapters', chapters)
        Table().with_columns(
            'Periods',    np.char.count(chapters, '.'),
            'Characters', text.apply(len, 0)
            ).scatter(0)
```



Question 1. About how many periods are in the chapter with the most characters? Assign either 1, 2, 3, 4, or 5 to the name `characters_q1` below.

1. 250
2. 390
3. 440
4. 32,000
5. 40,000

```
In [2]: characters_q1 = 2
```

Question 2. Which of the following chapters has the most characters per period? Assign either 1, 2, or 3 to the name `characters_q2` below. 1. The chapter with about 60 periods 2. The chapter with about 350 periods 3. The chapter with about 440 periods

```
In [3]: characters_q2 = 1
```

To discover more interesting facts from this plot, read Section 1.3.2 of the textbook.

1.3 3. Names and Assignment Statements

Question 1. When you run the following cell, Python produces a cryptic error message.

```
In [4]: 4 = 2 + 2
```

```
          File "<ipython-input-4-4c8b769209ad>", line 1
        4 = 2 + 2
                ^
    SyntaxError: can't assign to literal
```

Choose the best explanation of what's wrong with the code, and then assign 1, 2, 3, or 4 to `names_q1` below to indicate your answer.

1. Python is smart and already knows `4 = 2 + 2`.
2. `4` is a number, and it doesn't make sense to make a number be a name for something else. In Python, "`x = 2 + 2`" means "assign `x` as the name for the value of `2 + 2`."
3. It should be `2 + 2 = 4`.
4. I don't get an error message. This is a trick question.

```
In [4]: names_q1 = 2
```

Question 2. When you run the following cell, Python will produce another cryptic error message.

```
In [5]: two = 3
        six = two plus two
```

```
  File "<ipython-input-5-820d4d61e3dd>", line 2
    six = two plus two
                ^
SyntaxError: invalid syntax
```

Choose the best explanation of what's wrong with the code and assign 1, 2, 3, or 4 to `names_q2` below to indicate your answer.

1. The `plus` operation only applies to numbers, not the word "two".
2. The name "two" cannot be assigned to the number 3.
3. Two plus two is four, not six.
4. The name `two` cannot be followed directly by another name.

```
In [5]: names_q2 = 4
```

1.4 4. Differences between Universities

Question 1. Suppose you'd like to *quantify* how *dissimilar* two universities are, using three quantitative characteristics. The US Department of Education data on UW and Cal describes the following three traits (among many others):

Trait	UW	Cal
Average annual cost to attend ($)	13,566	13,707
Graduation rate (percentage)	83	91
Socioeconomic Diversity (percentage)	25	31

You decide to define the dissimilarity between two universities as the maximum of the absolute values of the 3 differences in their respective trait values.

Using this method, compute the dissimilarity between UW and CAL. Name the result `dissimilarity`. Use a single expression (a single line of code) to compute the answer. Let Python perform all the arithmetic (like subtracting 91 from 83) rather than simplifying the expression yourself. The built-in `abs` function takes absolute values.

```
In [6]: dissimilarity = max(abs(13566-13707),abs(83-91),abs(25-31))
        dissimilarity

Out[6]: 141
```

1.5 5. Studying the Survivors

The Reverend Henry Whitehead was skeptical of John Snow's conclusion about the Broad Street pump. After the Broad Street cholera epidemic ended, Whitehead set about trying to prove Snow wrong. (The history of the event is detailed here.)

He realized that Snow had focused his analysis almost entirely on those who had died. Whitehead, therefore, investigated the drinking habits of people in the Broad Street area who had not died in the outbreak.

What is the main reason it was important to study this group?

1) If Whitehead had found that many people had drunk water from the Broad Street pump and not caught cholera, that would have been evidence against Snow's hypothesis.

2) Survivors could provide additional information about what else could have caused the cholera, potentially unearthing another cause.

3) Through considering the survivors, Whitehead could have identified a cure for cholera.

```
In [7]: # Assign survivor_answer to 1, 2, or 3

        # I will accept either:
        survivor_answer = 1

        # or

        survivor_answer = 2
```

Note: Whitehead ended up finding further proof that the Broad Street pump played the central role in spreading the disease to the people who lived near it. Eventually, he became one of Snow's greatest defenders.

1.6 6. Creating Arrays

Question 1. Make an array called `weird_numbers` containing the following numbers (in the given order):

1. -2
2. the sine of 1.2
3. 3
4. 5 to the power of the cosine of 1.2

Hint: `sin` and `cos` are functions in the `math` module.

```
In [12]: # Our solution involved one extra line of code before creating
         # weird_numbers.
         import math
         weird_numbers = make_array(-2,math.sin(1.2),3,5**math.cos(1.2))
         weird_numbers

         # I also accepted these in list form:
         # weird_numbers = [-2,math.sin(1.2),3,5**math.cos(1.2)]
```

```
Out[12]: array([-2.        ,  0.93203909,  3.        ,  1.79174913])
```

Question 2. Make an array called `book_title_words` containing the following three strings: "Eats", "Shoots", and "and Leaves".

```
In [11]: book_title_words = make_array('Eats','Shoots','and Leaves')
         book_title_words

         # I also accepted these in list form, as with question 6.1

Out[11]: array(['Eats', 'Shoots', 'and Leaves'], dtype='<U10')
```

Strings have a method called `join`. `join` takes one argument, an array of strings. It returns a single string. Specifically, the value of `a_string.join(an_array)` is a single string that's the concatenation ("putting together") of all the strings in `an_array`, **except** `a_string` is inserted in between each string

Question 3. Use the array `book_title_words` and the method `join` to make two strings:

1. "Eats, Shoots, and Leaves" (call this one `with_commas`)
2. "Eats Shoots and Leaves" (call this one `without_commas`)

Hint: If you're not sure what `join` does, first try just calling, for example, `"foo".join(book_title_words)`.

```
In [13]: with_commas = ", ".join(book_title_words)
         without_commas = " ".join(book_title_words)

         # These lines are provided just to print out your answers.
         print('with_commas:', with_commas)
         print('without_commas:', without_commas)

with_commas: Eats, Shoots, and Leaves
without_commas: Eats Shoots and Leaves
```

1.7 7. Indexing Arrays

These exercises give you practice accessing individual elements of arrays. In Python (and in many programming languages), elements are accessed by *index*, so the first element is the element at index 0.

Question 1. The cell below creates an array of some numbers. Set `third_element` to the third element of `some_numbers`.

```
In [21]: some_numbers = make_array(-1, -3, -6, -10, -15)

         third_element = some_numbers[2]
         third_element

Out[21]: -6
```

Question 2. The next cell creates a table that displays some information about the elements of `some_numbers` and their order. Run the cell to see the partially-completed table, then fill in the missing information in the cell (the strings that are currently "???") to complete the table.

```
In [22]: # elements_of_some_numbers = Table().with_columns(
         #     "English name for position", make_array("first", "second", "???", "???", "fifth
         #     "Index",                     make_array("???", "1", "2", "???", "4"),
         #     "Element",                   some_numbers)
         elements_of_some_numbers = Table().with_columns(
             "English name for position", make_array("first", "second", "third", "fourth", "fi
             "Index",                     make_array("0", "1", "2", "3", "4"),
             "Element",                   some_numbers)
         elements_of_some_numbers
```

Out[22]:

English name for position	Index	Element
first	0	-1
second	1	-3
third	2	-6
fourth	3	-10
fifth	4	-15

Question 3. You'll sometimes want to find the *last* element of an array. Suppose an array has 142 elements. What is the index of its last element?

```
In [ ]: # I will accept any of the following:
        index_of_last_element = -1
        index_of_last_element = 141
        index_of_last_element = len(array)-1
```

More often, you don't know the number of elements in an array, its *length*. (For example, it might be a large dataset you found on the Internet.) The function `len` takes a single argument, an array, and returns the length of that array (an integer).

Question 4. The cell below loads an array called `president_birth_years`. The last element in that array is the most recent birth year of any deceased president. Assign that year to `most_recent_birth_year`.

```
In [24]: president_birth_years = Table.read_table("president_births.csv").column('Birth Year')
         president_birth_years
```

```
Out[24]: array([1732, 1735, 1743, 1751, 1758, 1767, 1767, 1773, 1782, 1784, 1790,
                1791, 1795, 1800, 1804, 1808, 1809, 1822, 1822, 1829, 1831, 1833,
                1837, 1843, 1856, 1857, 1858, 1865, 1872, 1874, 1882, 1884, 1890,
                1908, 1911, 1913, 1913, 1917])
```

```
In [28]: # I will accept either of the following
         most_recent_birth_year = president_birth_years[-1]
         most_recent_birth_year = president_birth_years[len(president_birth_years)-1]
         most_recent_birth_year = president_birth_years.item(-1)

         # And will give half credit for the following non-programmatic versions:
         most_recent_birth_year = 1917
         most_recent_birth_year = president_birth_years[37]

         most_recent_birth_year
```

```
Out[28]: 1917
```

1.8 8. Basic Array Arithmetic

Question 1. Multiply the numbers 42, 4224, 42422424, and -250 by 157. For this question, **don't** use arrays.

```
In [29]: first_product = 42*157
         second_product = 4224*157
         third_product = 42422424*157
         fourth_product = -250*157
         print(first_product, second_product, third_product, fourth_product)
```

```
6594 663168 6660320568 -39250
```

Question 2. Now, do the same calculation, but using an array called `numbers` and only a single multiplication (`*`) operator. Store the 4 results in an array named `products`.

```
In [30]: numbers = make_array(42,4224,42422424,-250)
         products = numbers * 157
         products
```

```
Out[30]: array([      6594,     663168, 6660320568,     -39250])
```

Question 3. Oops, we made a typo! Instead of 157, we wanted to multiply each number by 1577. Compute the fixed products in the cell below using array arithmetic. Notice that your job is really easy if you previously defined an array containing the 4 numbers.

```
In [31]: fixed_products = numbers * 1577
         fixed_products
```

```
Out[31]: array([      66234,     6661248, 66900162648,     -394250])
```

Question 4. We've loaded an array of temperatures in the next cell. Each number is the highest temperature observed on a day at a climate observation station, mostly from the US. Since they're from the US government agency NOAA, all the temperatures are in Fahrenheit. Convert them all to Celsius by first subtracting 32 from them, then multiplying the results by $\frac{5}{9}$. Round each result to the nearest integer using the `np.round` function.

```
In [32]: max_temperatures = Table.read_table("temperatures.csv").column("Daily Max Temperature"
         max_temperatures
```

```
Out[32]: array([25, 87, 89, ..., 62, 73, 61])
```

```
In [38]: celsius_max_temperatures = np.round((max_temperatures-32)*5/9)
         celsius_max_temperatures
```

```
Out[38]: array([-4., 31., 32., ..., 17., 23., 16.])
```

Question 5. The cell below loads all the *lowest* temperatures from each day (in Fahrenheit). Compute the size of the daily temperature range for each day. That is, compute the difference between each daily maximum temperature and the corresponding daily minimum temperature. **Give your answer in Celsius!**

```
In [40]: min_temperatures = Table.read_table("temperatures.csv").column("Daily Min Temperature"
         min_temperatures

Out[40]: array([13, 69, 67, ..., 31, 52, 41])

In [43]:

Out[43]: array([25, 87, 89, ..., 62, 73, 61])

In [44]: celsius_temperature_ranges = np.round(abs(celsius_max_temperatures - ((min_temperature
         celsius_temperature_ranges

Out[44]: array([ 7., 10., 13., ..., 18., 12., 11.])
```

1.9 9. World Population

The cell below loads a table of estimates of the world population for different years, starting in 1950. The estimates come from the US Census Bureau website.

```
In [45]: world = Table.read_table("world_population.csv").select('Year', 'Population')
         world.show(4)
```

```
<IPython.core.display.HTML object>
```

The name `population` is assigned to an array of population estimates.

```
In [46]: population = world.column(1)
         population

Out[46]: array([2557628654, 2594939877, 2636772306, 2682053389, 2730228104,
                2782098943, 2835299673, 2891349717, 2948137248, 3000716593,
                3043001508, 3083966929, 3140093217, 3209827882, 3281201306,
                3350425793, 3420677923, 3490333715, 3562313822, 3637159050,
                3712697742, 3790326948, 3866568653, 3942096442, 4016608813,
                4089083233, 4160185010, 4232084578, 4304105753, 4379013942,
                4451362735, 4534410125, 4614566561, 4695736743, 4774569391,
                4856462699, 4940571232, 5027200492, 5114557167, 5201440110,
                5288955934, 5371585922, 5456136278, 5538268316, 5618682132,
                5699202985, 5779440593, 5857972543, 5935213248, 6012074922,
                6088571383, 6165219247, 6242016348, 6318590956, 6395699509,
                6473044732, 6551263534, 6629913759, 6709049780, 6788214394,
                6866332358, 6944055583, 7022349283, 7101027895, 7178722893,
                7256490011])
```

In this question, you will apply some built-in Numpy functions to this array.

The difference function `np.diff` subtracts each element in an array by the element that precedes it. As a result, the length of the array `np.diff` returns will always be one less than the length of the input array.

The cumulative sum function `np.cumsum` outputs an array of partial sums. For example, the third element in the output array corresponds to the sum of the first, second, and third elements.

Question 1. Very often in data science, we are interested understanding how values change with time. Use `np.diff` and `np.max` (or just `max`) to calculate the largest annual change in population between any two consecutive years.

```
In [47]: largest_population_change = max(np.diff(population))
         largest_population_change
```

```
Out[47]: 87515824
```

Question 2. Describe in words the result of the following expression. What do the values in the resulting array represent (choose one)?

```
In [48]: np.cumsum(np.diff(population))
```

```
Out[48]: array([  37311223,   79143652,  124424735,  172599450,  224470289,
                 277671019,  333721063,  390508594,  443087939,  485372854,
                 526338275,  582464563,  652199228,  723572652,  792797139,
                 863049269,  932705061, 1004685168, 1079530396, 1155069088,
                1232698294, 1308939999, 1384467788, 1458980159, 1531454579,
                1602556356, 1674455924, 1746477099, 1821385288, 1893734081,
                1976781471, 2056937907, 2138108089, 2216940737, 2298834045,
                2382942578, 2469571838, 2556928513, 2643811456, 2731327280,
                2813957268, 2898507624, 2980639662, 3061053478, 3141574331,
                3221811939, 3300343889, 3377584594, 3454446268, 3530942729,
                3607590593, 3684387694, 3760962302, 3838070855, 3915416078,
                3993634880, 4072285105, 4151421126, 4230585740, 4308703704,
                4386426929, 4464720629, 4543399241, 4621094239, 4698861357])
```

1) The total population change between consecutive years, starting at 1951.

2) The total population change between 1950 and each later year, starting at 1951.

3) The total population change between consecutive years and 1950, starting at 1951.

```
In [52]: # Assign cumulative_sum_answer to 1, 2, or 3
         cumulative_sum_answer = 2
```

```
Out[52]: True
```

```
In [56]: # The following code proves that the above answer is correct. See if you can understa
         [np.cumsum(np.diff(population))[-idx] == population[-idx] - population[0] for idx in
```

```
Out[56]: [True,
          True,
          True,
          True,
          True,
          True,
```

```
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
    True,
```

```
True,
True,
True,
True,
True,
True,
True,
True,
True,
True]
```

1.10 10. Submission

Download this IPython notebook and upload it to your git repository. Instructions for this can be found here.