QUIZ 4

Topics: Data visualization and statistic.
Instructor: Dr. Selim Yılmaz (selimyilmaz@mu.edu.tr)
Out Date: 12/09/2022 23:59:59
Due Date: 12/23/2022 23:59:59

DECLARATION OF HONOR CODE[1]

In this course, I take academic integrity very seriously and ask you to do as well. That's why, this page is dedicated to some clear statements that define the policies of this assignment, and hence, will be in force. Before reading this assignment booklet, please first read the following rules to avoid any possible violation on academic integrity.

- This assignment must be done **individually** unless stated otherwise.
- You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an **abstract** way. That is, you **cannot copy code** (in whole or in part) of someone else, **cannot share your code** (in whole or in part) with someone else either.
- The previous rule also holds for the **material found on the web** as everything on the web has been written by someone else.
- You **must not look** at solution sets or program code from other years.
- You **cannot share or leave** your code (in whole or in part) in publicly accessible areas.
- You have to **be prepared to explain** the idea behind the solution of this assignment you submit.
- Finally, you must **make a copy of your solution** of this assignment and keep it **until the end of this semester**.

I have carefully read every of the statements regarding this assignment and also the related part of the official disciplinary regulations of Muğla Sıtkı Koçman University and the Council of Higher Education. By submitting my assignment, I hereby declare that I shall abide by the rules of this assignment to prevent any violation on academic integrity.

[1] This page should be read carefully before starting implementation.

Fraud Detection in Presidential Election of USA in 2012

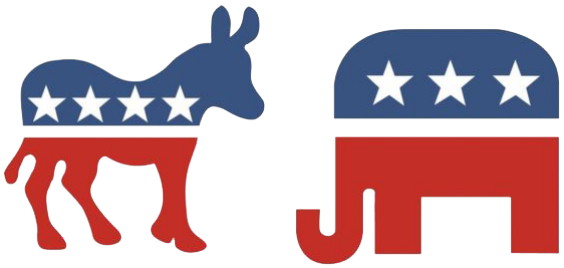

image credit:*http://www.bbc.co.uk/*

Introduction

In a society where freedom and democracy reign, people have a right to return to their ruling class. Election is a formal way of accepting or rejecting a political proposition by voting. People, particularly those living in poor countries, worry about their votes and question whether the election is trustworthy. However, thanks to the statistical tools, we can easily find out if any fraud attempts were made.

In this assignment, you will implement a *Python* program that analyzes the results of the 2012 presidential election in the United States of America (USA) and interprets whether it is fraudulent or not. It was an election with four major parties (*Democratic*, *Republican*, *Libertarian*, and *Green*) and 30 nominees, most of whom were write-in candidates. Democrat nominee *Obama B.*, republican nominee *Romney M.*, libertarian nominee *Jonhson G.*, and the green nominee *Stein J.* participated in the election that resulted in the victory of the Democrats. You are given the election results in a file named `ElectionUSA2012.csv`[2]. This file records the number of votes separately for each state. Each row represents a state in the USA, and there are eight different information in a single row: State name, total votes, electoral votes, total vote, # of votes for Obama, Romney, Johnson, Stein, and others. To sum up, there are 204 election results (you can exclude the votes for 'others' in this assignment) you care about to reveal fraudulence, if any.

You will look closely at the least significant digits of votes (the **ones** and **tens** of numbers in this assignment), which are essentially noise and do not affect who wins. The idea is that in any real election, we expect the ones and tens to be uniformly distributed; that is, 10% of the digits is 0, 10% of the digits is 1, and so forth. If the distribution is not uniform, then it is likely made up by someone rather than collected from the ballot boxes. To accomplish this assignment, there are some steps you need to take.

Step 1: Read election data and write them in another format

The content of `ElectionUSA2012.csv` file is explained in the previous section. Each information in a row is separated by a comma (,). To read the file, write a function called `retrieveData` that takes two inputs, one of which represents the **filename**, and the other is a **list** consisting of the names of the nominees. It returns a one-dimensional list that contains the vote counts from every row in a successive manner. Save the output to a file named `retrievedData.txt`

[2]https://www.fec.gov/introduction-campaign-finance/election-and-voting-information/federal-elections-2012/

>>> retrieveData("ElectionUSA2012.csv", ["Obama","Romney","Johnson","Stein"])
[795696, 122640, 1025232, 394409, 7854285, ..., 20928, 4406, 7665, 0]

Note 1: The arguments of the defined function (filename and a list of nominees' names) should be dynamic and take their values from command-line arguments. It is necessary to call your program as shown below to invoke `retrieveData` with the parameter values of those illustrated above:

$ python VoteAnalyzer.py ElectionUSA2012.csv Obama,Romney,Johnson,Stein

Note 2: A change in the order of the nominees' names (2nd command-line argument) should affect the output.

Note 3: Once you have written the data, you are done with it. Feel free to read the input file from scratch to perform the following tasks.

Step 2: The bar graph of the vote counts

Here you are to plot a bar figure to visualize the vote distribution of two nominees who dominated the election (Obama and Romney for the 2012 election of USA) as a function of state name. To do that, write a function `DispBarPlot` that takes no input and returns none. The figure should look exactly the same as the one given in Fig. 1. In the figure, the x-axis represents the states, whereas the y-axis represents the vote counts each nominee took. Vote counts should be represented with blue and red bars. Do not forget to create a legend box (shown in the top right of Figure 1) without which nothing would be interpreted. Save your first figure in a file named `ComparativeVotes.pdf`.

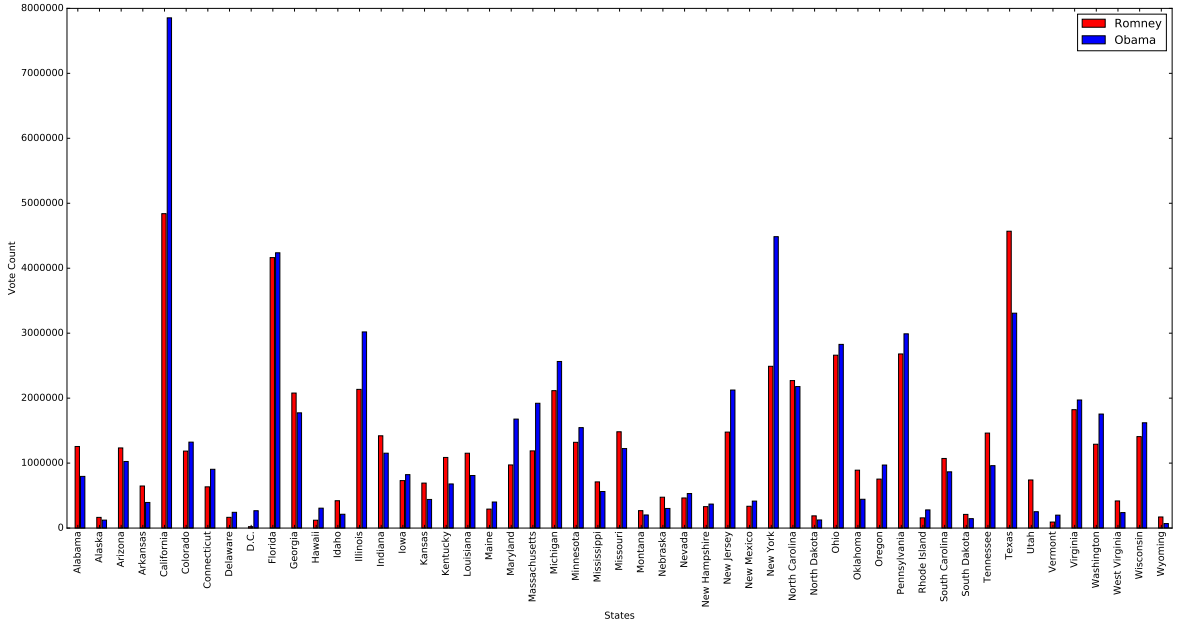

Figure 1: Comparative demonstration with bar plot

Note 1: Your implementation should consider only two nominees having the highest votes.

Note 2: Your implementation should output the same plot every time you run in varying order in nominees' names. Likewise, your plot should not be affected by a change in the order of the header of the provided file.

Step 3: Bar plot for vote comparison

To reveal the margins between the votes given to each nominee, you are expected to visualize the comparative vote percentages of all nominees. To do that, write a function named `compareVoteonBar` that creates a figure window containing a bar plot that should look exactly the same as provided in Fig. 2. In this plot, vote percentages should be given as x-axis and nominees' names should be provided in a legend box. As you may realize, there is no vote percentage present in the election data file. Therefore, you first need to obtain the percentages of votes and visualize them in **descending order** (as in Fig. 2). Save your plot in a file named `CompVotePercs.pdf`. Consider again the notes given in the previous step.

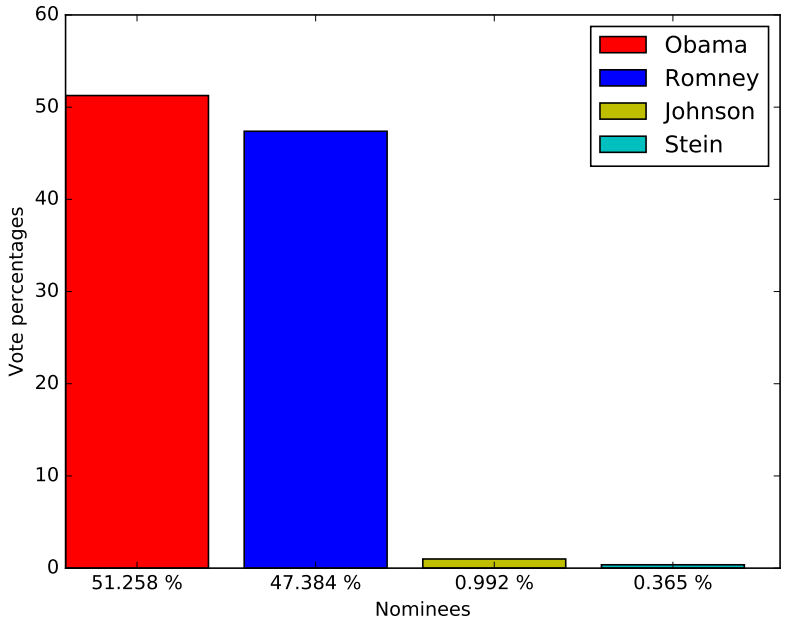

Figure 2: Comparative vote percentages of nominees

Step 4: Obtain histogram

As opposed to the previous step, now you are concerned about the ones and tens of the digits of votes to get their frequencies. To do that, write a function named `obtainHistogram` that takes a list as input and produces an output as a list of 10 numbers. Each element of the output list represents the frequency of digits appearing in the one- and ten-places of the numbers in the input list. Note that the tens place is 0 while ones place is the number itself for the numbers less than 10.

```
>>> obtainHistogram([7, 24, 25, 180, 249, 326, 446, 446, 512, 552, 612, 618, 618, 714, 780, 839, 846, 890, 949, 951])
[0.1, 0.15, 0.15, 0.025, 0.175, 0.075, 0.1, 0.025, 0.1, 0.1]
```

Step 5: Histogram plot

To complete this step, you need to obtain a frequency list calculated in `obtainHistogram`. Create a function named `plotHistogram` that takes a frequency list and plots the frequencies of one and ten places of numbers for the 2012 election data of USA. Your histogram plot should look exactly the same as provided in Fig. 3. In this figure, you see two plot lines with different colors. The straight red line is the frequency distribution of the numbers ranging from 0 to 9 whereas the green dashed line is the *ideal line* for the uniform distribution. The x- and y-axes of the plot represent the digit value and the corresponding frequency, respectively. Do not forget to add the legend box here and save your figure as `Histogram.pdf`.

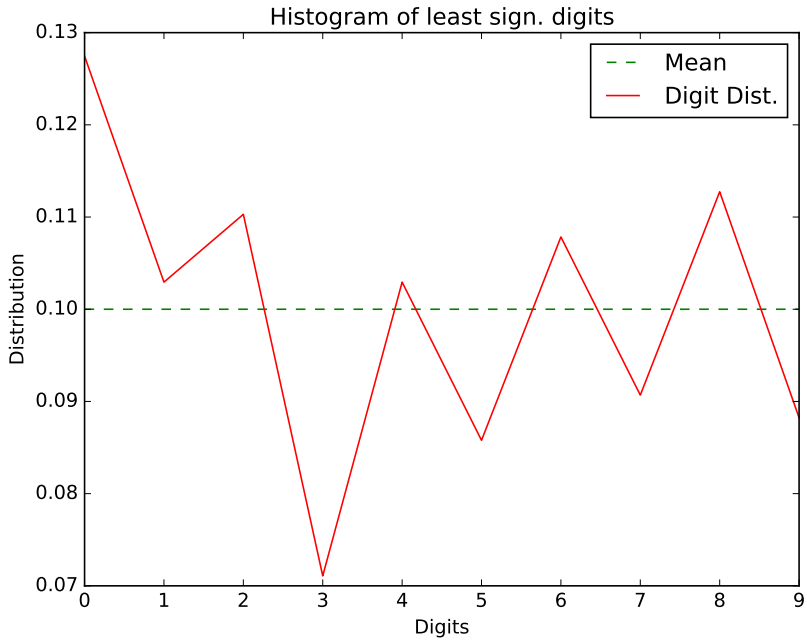

Figure 3: Histogram plot

As seen, the USA election data is rather different from expected ideal line. However, looking only at Fig. 3, we cannot deduce if it is a fraudulent election. We will appeal more to the principles of statistical ways.

Step 6: Histogram plot of smaller samples

This is a repetition of the previous step, but with smaller samples randomly generated. Write a function `plotHistogramWithSample` without input. Create five different-sized (10, 50, 100, 1000, and 10000) lists of random numbers ranging from 0 to 100. For each size, perform the previous step and create a histogram plot of the generated random numbers.

These plots should be in different colors to distinguish them (as shown in Fig. 4). Once you create, you realize that the more samples you use, the closer to the ideal line it is. Save each of the figures named `HistogramofSample1.pdf, HistogramofSample2.pdf, ..., HistogramofSample5.pdf`.

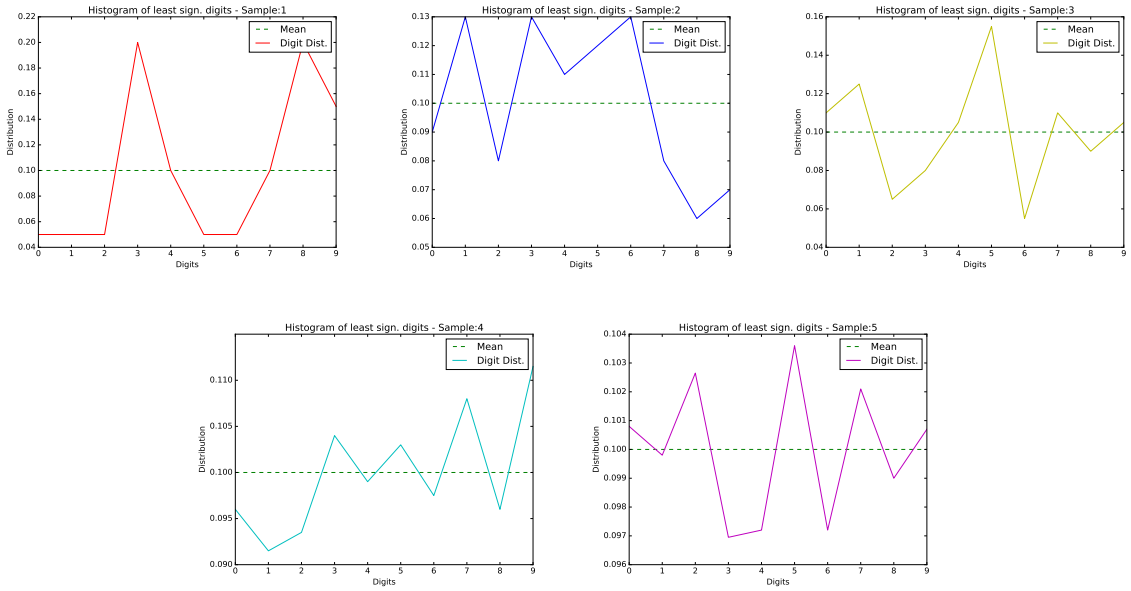

Figure 4: Histogram plot of random samples

It is not surprising that you obtain histogram plots that show different frequency distributions, since they are created with random numbers. Also, you get a different histogram plot every time you run.

Step 7: Similarity calculation of vectors

As you all know, the similarity of two plot lines increases with an increase in the number of samples used. But we need a computational way to also verify such closeness. As plot lines are created by list, here we need to calculate the similarities of two given lists. One common way to calculate this is **Mean Squared Error** (MSE). Write a function named `calculateMSE` taking two lists to calculate their similarity. An illustration of the MSE calculation for two lists is given below:

```
>>> calculateMSE([4, 7, 2, 3], [5, 2, 9, 6])
```

$21 = \frac{(4-5)^2+(7-2)^2+(2-9)^2+(3-6)^2}{4}$

Step 8: Calculation of the MSE for the USA election

Once you have completed the previous step, you can now calculate the MSE values of the USA election data. To do that, invoke `calculateMSE` by passing the frequency list (recall that it is obtained from `obtainHistogram`) to obtain the MSE of that histogram with the uniform distribution represented by the green dashed line (*ideal line*) in Fig. 3. When you

invoke the `calculateMSE` function with frequency distribution of the 2012 election of USA data, it should output the MSE value of 0.00023644752018454436, or approximately 0.0002, if it works correctly.

Step 9: Comparison of MSEs (optional)

This step is closely related to the next step; to achieve the next step, you need to compare the MSE values of the USA with those of 10,000 groups of random numbers of the same size as the election data of the USA (i.e., 204 numbers). Write a function named `compareMSEs` that takes an argument of the MSE value of the USA election histogram calculated in the previous step, and then move on to the next step to find the details of the implementation of this function.

Step 10: Interpretation of results (optional)

Once the MSEs have been calculated, it is time to interpret the results in this final step. Here, **null hypothesis** means that our observed sample (the 2012 election data in the USA) is **not fraudulent**. To prove that the election data are fraudulent, we must **reject** the null hypothesis[3].

In order to test the null hypothesis, we first need a p-value of the election data from the USA that represents *level of rejection*. Here, you should pay attention to the cases where the MSE results of the USA election are greater than those of a 10,000 groups of random numbers. To calculate the p value of the USA election data, divide the number of times the MSE of the USA election data is greater than that of 10,000 groups of random numbers, denoted T_s by a value of 10,000:

$$p - value = \frac{T_s}{10,000} \tag{1}$$

If the MSE value of the USA election is smaller than or equal to % 5 of the MSEs calculated from random numbers (a common value of significance level -α), you can conclude that the null hypothesis is rejected and confidently claim that the election results are fraudulent, and *vice versa*.

You should display the results on the console and also in a file named `myAnswer.txt`. The content of your program's output should exactly match the following formatting, including capitalization and spacing (except where ___ is replaced by your findings).

Output content and format:

[3]to get an insight on hypothesis testing, read *https://onlinecourses.science.psu.edu/statprogram/node/138*

MSE value of 2012 USA election is ___
The number of MSE of random samples that are larger than or equal to USA election MSE is ___
The number of MSE of random samples that are smaller than USA election MSE is ___
2012 USA election rejection level p is ___
Finding: We reject the null hypothesis at the p= ___ level ***or***
Finding: There is no statistical evidence to reject null hypothesis

General Note: You are encouraged to use `DataFrame` of the `Pandas` module as an alternative to `Python list` in your implementation.

Notes

- **Do not miss** the deadline and **save your work** until the beginning of next semester.
- Your source code should be designed as **easy-to-follow**. Decompose your source code as much as possible (i.e., adopt functions).
- **Place comment** in it as much as possible.
- The assignment must be **original**, **individual work**. **Duplicate or very similar assignments are both going to be considered as cheating.**
- You will submit your work on SE1003 course page at https://dys.mu.edu.tr with the file hierarchy as below[4]:

→ <student id>.zip
→ VoteAnalyzer.py

[4] do not put any of these files into any folder, **compress them together directly** and submit the compressed file!