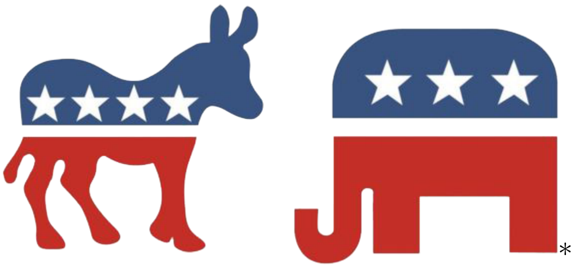
PROGRAMMING ASSIGNMENT 4

Subject : Data visualization and statistic
Advisor : Res. Assist. (Necva BOLUCU, Selma DILEK, Burcu YALCINER, Selim YILMAZ)

Due Date : 21.12.2016

Fraud Detection in 2012 USA Presidential Election



*taken from *http://www.bbc.co.uk/*

Introduction

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

In this assignment, you will implement a *Python* program that analyzes the results of the USA presidential election held in 2012 and interprets whether it is fraudulent or not. It was election with 4 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 green nominee *Stein J.* participated in the election resulting with the victory of the democrats. You are provided with election results in a file named `ElectionUSA2012.csv`[1]. This file records the number of votes state by state. There are eight different information in one row: State name, total votes, electoral votes, total vote, # of votes for Obama, Romney, Johnson, Stein, and others. Each row represents a state in the USA. To summarize, there are 204 election results (exclude the votes for "others") you care about to reveal fraudulence, if any.

You will look closely at the least significant digits of votes (ones and tens place 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; namely 10% of the digits is 0, 10% of the digits is 1, and so forth. If distribution is not uniform, then it is likely

[1] obtained from http://www.fec.gov/pubrec/fe2012/federalelections2012.shtml

made-up by someone rather than collected from ballot-boxes. To accomplish this assignment, there are some steps you need to carry out.

Step 1: Read election data

The content of `ElectionUSA2012.csv` file is explained in previous section. Each information in a row is separated with a delimiter (,). 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 nominees' names. It returns a one-dimensional list that contains the vote counts from every row in a successive manner. Save the output under a file named retrievedData.txt

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

Note 1: The arguments of the function defined (filename and a list of nominees' names) should be dynamic and take their values from command-line arguments. To invoke retrieveData with the parameter values of those illustrated above, it is necessary to call your program as shown below[2]:

```
$ python Assignment4.py ElectionUSA2012.csv Obama,Romney,Johnson,Stein
```

Note 2: A change in the order of nominees' names (2nd system argument) should change the output.

Step 2: Bar plot of vote counts

Once you obtain all vote counts, plot a bar figure in order to visualize the vote distribution of two nominees who dominated the election (Obama and Romney for 2012 USA election) 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 same as given in Fig. 1. In the figure, x-axis represents the states whereas y-axis represents vote counts each nominee took. Vote counts should be represented with blue and red bars. Do not forget to create legend box without which nothing would be interpreted. Save your first plot in a file named ComparativeVotes.pdf.

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

Step 3: Bar plot for vote comparison

In order to reveal the margins between the votes given to each nominee, you are expected to visualize the comparative vote percentages of all nominees. Write a function compareVoteonBar that creates a figure window containing bar plot that should look exactly same as provided in Fig 2. In this plot, vote percentages should be given as x-labels and nominees' names should be provided in a legend box. As all you realize, there is no vote percentage present in election data file. Therefore, you need to obtain vote percentages first and visualize them in

[2] check your interpreter version by typing python −−version and be sure it is 3.x

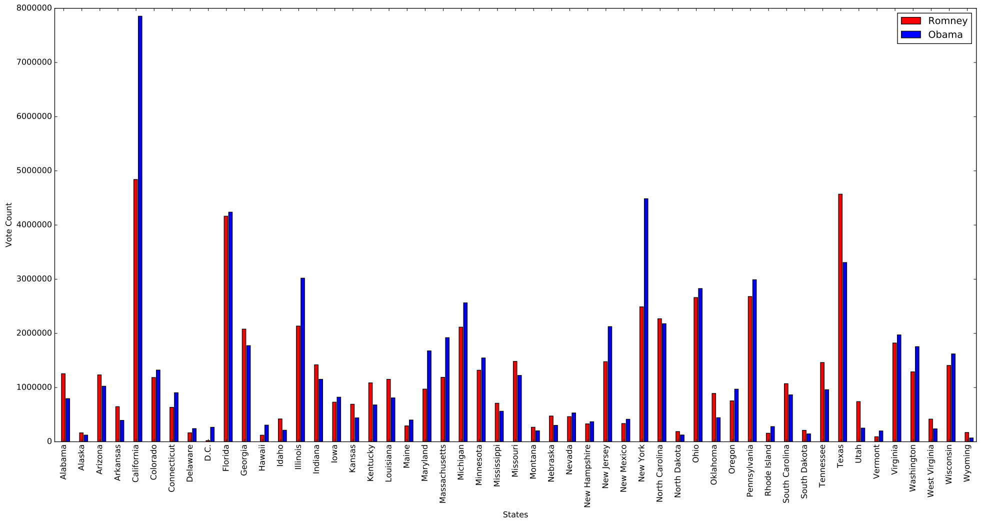


Figure 1: Comparative demonstration with bar plot

a descending order (as in Fig 2). Save your plot in a file named CompVotePercs.pdf. Consider the note given in the previous step.

Step 4: Obtain histogram

As opposed to the previous step, now you care about ones and tens digits of votes to get the frequencies of them. 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 output list represents the frequency of digit appeared in ones and tens place in input. Note that it is 0 for numbers less than 9.

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

As seen, the USA election data is rather different from expected ideal line. However, looking only in Fig. 3, we cannot deduce if it is fraudulent election. We will appeal more principle

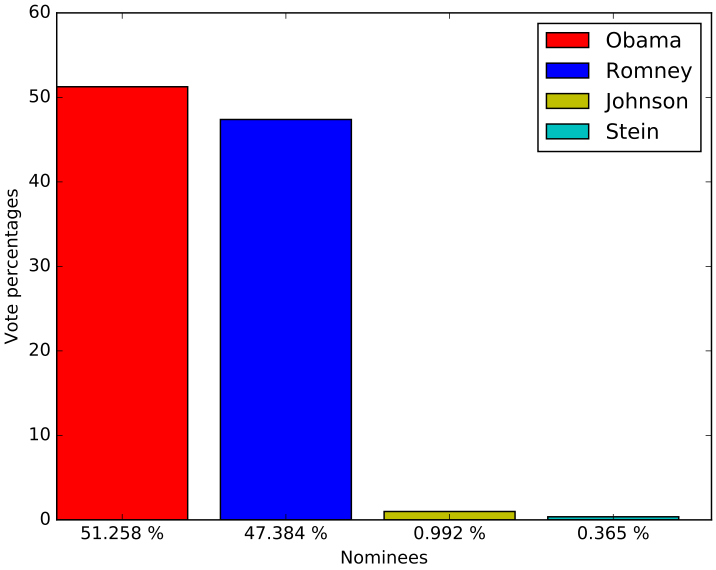


Figure 2: Comparative vote percentages of nominees

statistical ways.

Step 6: Histogram plot of smaller size samples

This is the repetition of the previous step but with smaller samples randomly generated. Write a function plotHistogramWithSample taking no input. Create 5 different-sized (10, 50, 100, 1000, and 10000) lists of random numbers ranging from 0 to 100. For each size, perform previous step and create histogram plot of the generated random numbers. These plots should be in different colors to distinguish them (as shown in Fig. 4). Once you created, you realize the more sample you use the closer to the ideal line it is. Save each of the figure named HistogramofSample1.pdf, HistogramofSample2.pdf, ..., HistogramofSample5.pdf.

It is not surprising that you obtain histogram plots that show different frequency, as it is created with random numbers. Besides, you obtain different histogram plot every time you run.

Step 7: Uniformity calculation

As you all realize the closeness of two plot lines increases with an increase in the number of samples used. But we need computational way to also verify such closeness. As plot lines are created by list, here we need to calculate the difference/closeness of given two lists. One common way for calculation is *mean squared error* (MSE). Write a function calculateMSE taking two lists to calculate the closeness of them. An illustration of MSE calculation is given below:

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

84 → $(4\text{-}5)^2+(7\text{-}2)^2+(2\text{-}9)^2+(3\text{-}6)^2$

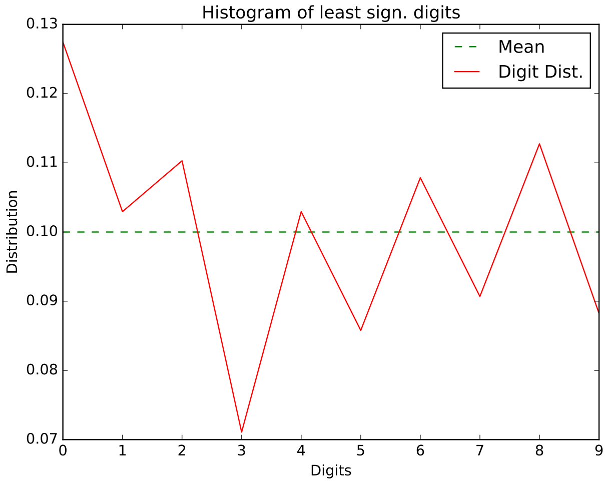


Figure 3: Histogram plot

Step 8: MSE calculation of USA election

Once you completed the previous step, now you can calculate MSE values of USA election data. To do that, write a function that takes a histogram (remember, it is obtained from obtainHistogram) and returns the mean squared error of that histogram with the uniform distribution represented by green dashed line (*ideal line*) in Fig. 3. When you invoke calculateMSE function with an input of histogram data, it should output the MSE value of 0.0023644752018454436, or approximately 0.002, if it works correctly.

Step 9: Comparison of MSEs

This step is closely related to the following step, to accomplish the next step you need to compare MSE values of USA to those of 10000 groups of random numbers with same size as USA election data (204 numbers). Write a function named compareMSEs that takes an argument of MSE value of USA election histogram calculated in previous step then go to next step.

Step 10: Interpreting results

Once calculated MSEs, it is the turn to interpret the results in this final step. Here, **null-hypothesis** is our observed sample (the USA election data) is not fraudulent. To prove election data is fraudulent, we must reject the null hypothesis[3]. Here, we need p-value of USA election data which represents the rejection level. Here, you should pay attention to the %of MSEs that USA election result is greater than those obtained in previous step. To calculate p-value of USA election data you should divide the number of times that MSE of USA election data is greater than those obtained in previous step to 10000 which is the number of groups. If MSE value of USA election is smaller that % 5 of random MSEs (a

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

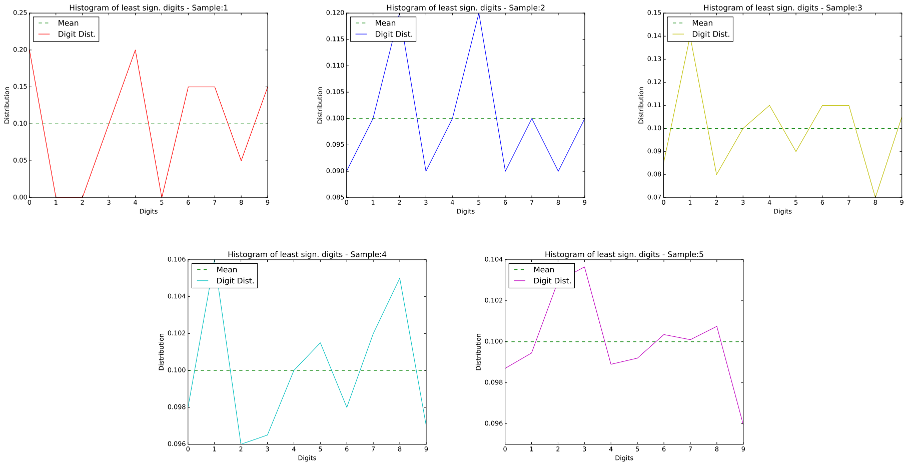


Figure 4: Histogram plot of random samples

common value of significance level -α) (which is 500 random MSEs), you can conclude that null-hypothesis is rejected and confidently claim that the election results are fraudulent, and *vice versa*.

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

Formatting:

MSE value of 2012 USA election is ___

The number of MSE of random samples which are larger than or equal to USA election MSE is ___

The number of MSE of random samples which 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

Notes specified to this assignment

- Avoid redundancy, never repeat yourself!
- The structure of your implementation should be dynamic, do not define static expressions. As your grades will be evaluated on a completely different dataset, I advise you to test your implementation over a different dataset but having same structure. For this reason, 2008 presidential election results of USA —`ElectionUSA2008.csv`—are also provided to you.
- Do not define static path in order that it runs properly on any PC.
- As you cannot display any plot, do not run your work on your own 'DEV space'.
- Feel free to employ any built-in function.
- Do not attempt to avoid extreme cases (null value i.e.) possibly written in the command line.
- Be sure your submitted work exactly matches the hierarchy detailed below, as the submission with the score of 0 will not be considered for evaluation.
- Should you have a question do not hesitate to ask, but first consider office hours of TA in charge of this assignment(Selim YILMAZ).

Notes

- Do not miss the deadline.
- Save all your work until the assignment is graded.
- The assignment must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.
- You can ask your questions via Piazza (https://piazza.com/hacettepe.edu.tr/fall2016/bbm101) and you are supposed to be aware of everything discussed in Piazza.
- You will submit your work from https://submit.cs.hacettepe.edu.tr/index.php with the file hierarchy as below:

This file hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system)

```
→ <student id>
    → assignment4.py
    → retrievedData.txt
    → ComparativeVotes.pdf
    → CompVotePercs.pdf
    → Histogram.pdf
    → HistogramofSample1.pdf
    → HistogramofSample2.pdf
    → HistogramofSample3.pdf
    → HistogramofSample4.pdf
    → HistogramofSample5.pdf
    → myAnswer.txt
```