Lab 4

Group members:

- Abebe Amare (ama100)
- Nathan Kim (njk24)
- Eugene Kim (yk171)
- Revo Tesha (rat20)
- Sibora Seranaj (ss801)

CS 216, Everything Data, Spring 2020

In this lab exercise, you will apply hypothesis testing and Bayesian inference for data analysis. You should work in a group of 4-5 students of your choice; you will only need to submit once as a group (see submission instructions below). Remember that labs are graded for sincere effort: if you do not finish the entire lab assignment before the end of class, turn in what you have so far.

In [1]:
```
# First we need to import some libraries that we will
# need or that might otherwise be useful for you. We give
# a short description, but feel free to look up the
# documentation on using additional features of these
# libraries. The name abbreviations used here are standard
# conventions.

import csv # For pulling data from csv files in standard Python
import pandas as pd # For handling structured data in Python
import numpy as np # For numerical computation in Python
import matplotlib.pyplot as plt # for plotting histograms
import seaborn as sns # for easier to use visualization
from scipy import stats # For computing chi-squared test
```

Part 1: Hypothesis Testing Benford's Law

Suppose that you measure some naturally-occurring phenomenon, such as the land area of cities or lakes, or the price of stocks on the stock market. You can plot a histogram of of the first digits of your measurements, showing how frequent each first digit (1-9) is. For example, maybe you measure stock prices and get \$10, \$85, \$132, and \$29. The resulting distribution of first digits would be [1, 8, 1, 2], with 1 being the most common.

Your measurements were made in some arbitrary units, like dollars. What if you made the measurements in some other units, like pesos, euros, or yuan? Would you expect the shape of the histogram of first digits to change much? Data are called scale-invariant if measuring them in any scale (say, meters vs. feet vs. inches) yields similar statistical results, such as the same histogram of first digits. Many natural processes produce scale-invariant data. Benford's Law states the remarkable fact there is only one possible histogram that results from all of these processes! Let P(d) be the probability of the given digit $d \in \{1, 2, \ldots, 9\}$ being the first one in a measurement. Then, we have the probability distribution

$$P(d) = \log_{10}(d+1) - \log_{10}(d)$$

If we can demonstrate that a given distribution that should follow Benford's law does not, that can sometimes be an indication of incorrect or fraudulent data. In this part, we will look at the first digit distributions of population and vote counts and compare with the counts we would expect according to Benford's law. Note that Benford's law only holds when the data has no natural limit nor cutoff. For example, it would not hold for people's height in meters (where almost every value would start with 1 or 2). If you are interested, Wikipedia has more information (https://en.wikipedia.org/wiki/Benford%27s_law) about Benford's Law. (You don't need to read the Wikipedia article in order to complete this problem, however.)

Problem A

For this problem, we will look at the data in SUB-EST2009_ALL.csv which is the population numbers from the US census. We have included some Python code below that pulls the pupulation data from the 2000 census, plots the histogram of their first digits side-by-side with the "theoretical" histogram we would expect according to Benford's law, and performs hypothesis testing (using a $\chi 2$ test (https://en.wikipedia.org/wiki/Chi-squared_test)) to know how well the observed first-digit frequencies conform to Benford's law.

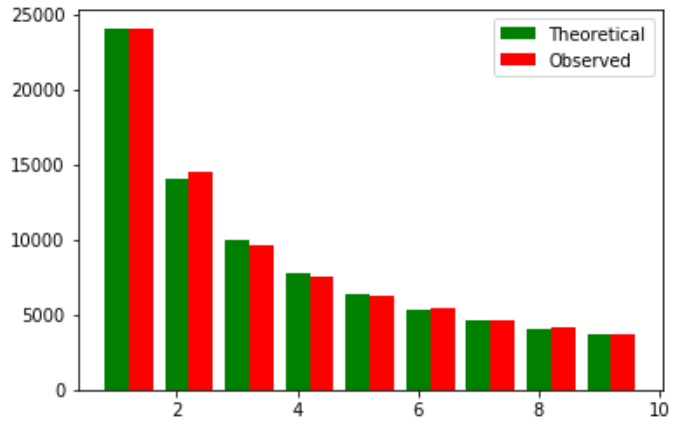
Your task is to run the below code to compare the distributions and compute p-value. Is it close to 1 or 0? What does that allow you to conclude about whether the data follow Benford's law?

In [2]:
```
# We pull the frequencies of first digits from
# the data file. We show a Pandas/Numpy approach
# here, you are welcome to use a similar approach
# or to use standard Python for parsing the data
# from the csv
df_census = pd.read_csv('SUB-EST2009_ALL.csv')
pop_counts_2000 = df_census[df_census['POPCENSUS_2000'] != 'X']['POPCENSUS_2000'].valu
es
first_digits = np.array([int(pop_counts_2000[i][0]) for i in range(len(pop_counts_2000
))])
freqs = np.array([np.sum(first_digits==i) for i in range(1,10)])
print(freqs)
```

```
[23976 14543  9656  7587  6253  5425  4604  4161  3741]
```

In [3]:
```
def benford():
    return [np.log10(d+1)-np.log10(d) for d in range(1,10)]

theoretical_freqs = np.array([f * np.sum(freqs) for f in benford()])
```

In [4]:
```
# Plot two distros side-by-side:

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
# The third parameter below is the width of the bar, set
# such that we can squeeze another set of bars in:
rects1 = ax.bar(range(1,10), theoretical_freqs, 0.4, color='g')
# In the following, adding the bar width to the x coordinates
# position this set of bars side-by-side with the last:
rects2 = ax.bar([i+0.4 for i in range(1,10)], freqs, 0.4, color='r')
ax.legend((rects1[0], rects2[0]), ('Theoretical', 'Observed'))
plt.show()
```



In [5]:
```
# Computing p-value
chisq, pvalue = stats.chisquare(freqs, theoretical_freqs)
print('chi-squared test p-value: ', pvalue)
```

```
chi-squared test p-value:  2.2747922135321912e-05
```

What can you conclude about whether the data follow Benford's law?

The null hypothesis is that the distribution of first digits was generated by Benford's law, that is, it is scale invariant. Although the histograms look extremely similar, the p-value is extremely small, allowing us to reject the null hypothesis: the data do not follow Benford's law and are therefore not scale invariant.

Problem B

In this problem, you will use hypothesis testing to analyze data from the 2009 Iranian election; the data file for this problem is election-iran-2009.csv. The numbers of interest in the .csv file are the vote counts for candidates Ahmadinejad, Rezai, Karrubi, and Mousavi. There were multiple allegations of fraud in this election; your goal is to examine whether that can be substantiated by comparison with Benford's law.

As in problem A, consider the first-digit distribution of all candidates' vote counts (that is, check separately the distribution of digits for each of the four candidates). For each, draw the histogram side by side with the theoretical distribution and use hypothesis testing (with a p-value of 0.05) to test if the distributions follow Benford's law. You can use the code above as a model, or write your own. What do your results allow you to conclude?

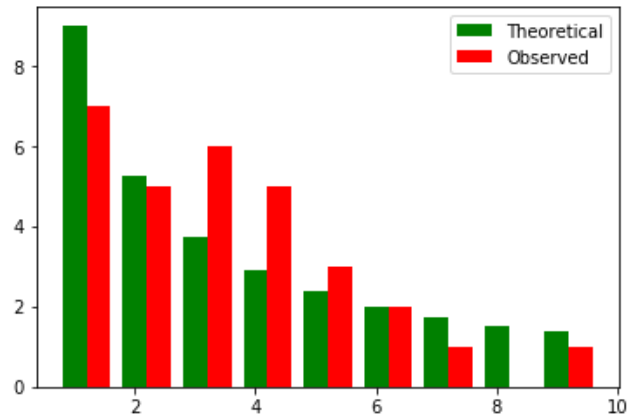
In [9]:
```
# We pull the data into a Pandas data frame here
# Feel free to import it from the CSV using standard
# Python if you prefer, but note that the vote counts
# have commas in them.
df_iran = pd.read_csv('election-iran-2009.csv', thousands=',')
df_iran.head()
```

Out[9]:

	Region	Ahmadinejad	%	Rezai	%	Karrubi	%.1	Mousavi	%.2	Total votes	Invalid votes	Valid votes	
0	East Azerbaijan	1131111	56.75	16920	0.85	7246	0.36	837858	42.04	2010340	17205	1993135	2
1	West Azerbaijan	623946	47.48	12199	0.93	21609	1.64	656508	49.95	1334356	20094	1314262	1
2	Ardabil	325911	51.11	6578	1.03	2319	0.36	302825	47.49	642005	4372	637633	8
3	Isfahan	1799255	68.88	51788	1.98	14579	0.56	746697	28.58	2637482	25163	2612319	2
4	Ilam	199654	64.58	5221	1.69	7471	2.42	96826	31.32	312667	3495	309172	3

In [42]:
```
## code for Ahmadinejad
Ahmadinejad=df_iran["Ahmadinejad"]
#Rezai=df_iran["Rezai"]
#Karrubi=df_iran["Karrubi"]
#Mousavi=df_iran["Mousavi"]

first_digits = np.array([int(str((Ahmadinejad.iloc[i]))[0]) for i in range(len(Ahmadin
ejad))])
print(first_digits[0])

freqs = np.array([np.sum(first_digits==i) for i in range(1,10)])


def benford():
    return [np.log10(d+1)-np.log10(d) for d in range(1,10)]

theoretical_freqs = np.array([f * np.sum(freqs) for f in benford()])

# Plot two distros side-by-side:

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
# The third parameter below is the width of the bar, set
# such that we can squeeze another set of bars in:
rects1 = ax.bar(range(1,10), theoretical_freqs, 0.4, color='g')
# In the following, adding the bar width to the x coordinates
# position this set of bars side-by-side with the last:
rects2 = ax.bar([i+0.4 for i in range(1,10)], freqs, 0.4, color='r')
ax.legend((rects1[0], rects2[0]), ('Theoretical', 'Observed'))
plt.show()
#print(freqs)
# Computing p-value
chisq, pvalue = stats.chisquare(freqs, theoretical_freqs)
print('chi-squared test p-value: ', pvalue)
```

```
1
```

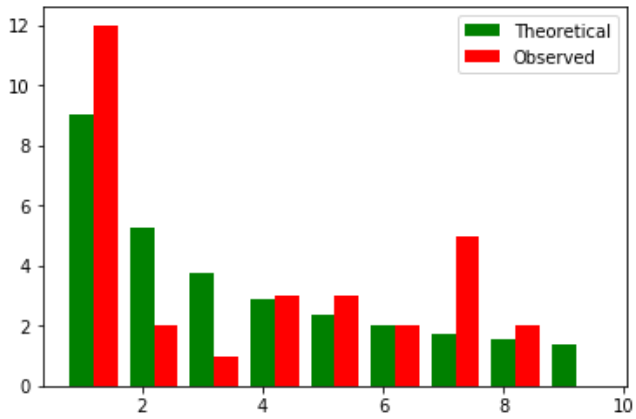



```
chi-squared test p-value:  0.7090550582450321
```

In [ ]:
```
## pvalue
```

In [38]:
```
## code for Rezai
Rezai=df_iran["Rezai"]
#Rezai=df_iran["Rezai"]
#Karrubi=df_iran["Karrubi"]
#Mousavi=df_iran["Mousavi"]

first_digits = np.array([int(str((Rezai.iloc[i]))[0]) for i in range(len(Rezai))])
print(first_digits[0])

freqs = np.array([np.sum(first_digits==i) for i in range(1,10)])


def benford():
    return [np.log10(d+1)-np.log10(d) for d in range(1,10)]

theoretical_freqs = np.array([f * np.sum(freqs) for f in benford()])

# Plot two distros side-by-side:

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
# The third parameter below is the width of the bar, set
# such that we can squeeze another set of bars in:
rects1 = ax.bar(range(1,10), theoretical_freqs, 0.4, color='g')
# In the following, adding the bar width to the x coordinates
# position this set of bars side-by-side with the last:
rects2 = ax.bar([i+0.4 for i in range(1,10)], freqs, 0.4, color='r')
ax.legend((rects1[0], rects2[0]), ('Theoretical', 'Observed'))
plt.show()
#print(freqs)

# Computing p-value
chisq, pvalue = stats.chisquare(freqs, theoretical_freqs)
print('chi-squared test p-value: ', pvalue)
```

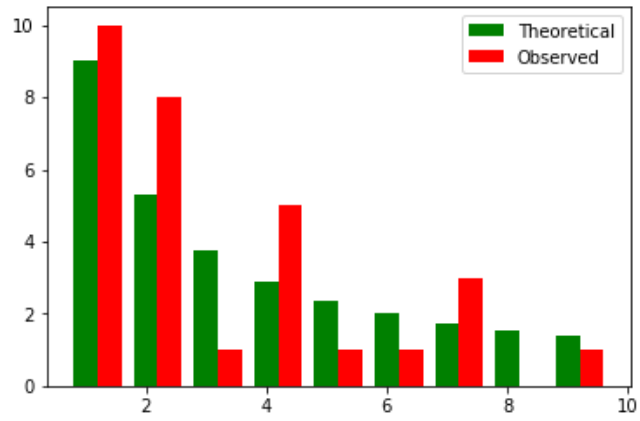
```
1
```




```
chi-squared test p-value:  0.11813362307099959
```

In [39]:
```
## code for Karrubi

Karrubi=df_iran["Karrubi"]
#Rezai=df_iran["Rezai"]
#Karrubi=df_iran["Karrubi"]
#Mousavi=df_iran["Mousavi"]

first_digits = np.array([int(str((Karrubi.iloc[i]))[0]) for i in range(len(Karrubi))])
print(first_digits[0])

freqs = np.array([np.sum(first_digits==i) for i in range(1,10)])


def benford():
    return [np.log10(d+1)-np.log10(d) for d in range(1,10)]

theoretical_freqs = np.array([f * np.sum(freqs) for f in benford()])

# Plot two distros side-by-side:

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
# The third parameter below is the width of the bar, set
# such that we can squeeze another set of bars in:
rects1 = ax.bar(range(1,10), theoretical_freqs, 0.4, color='g')
# In the following, adding the bar width to the x coordinates
# position this set of bars side-by-side with the last:
rects2 = ax.bar([i+0.4 for i in range(1,10)], freqs, 0.4, color='r')
ax.legend((rects1[0], rects2[0]), ('Theoretical', 'Observed'))
plt.show()
#print(freqs)

# Computing p-value
chisq, pvalue = stats.chisquare(freqs, theoretical_freqs)
print('chi-squared test p-value: ', pvalue)
```

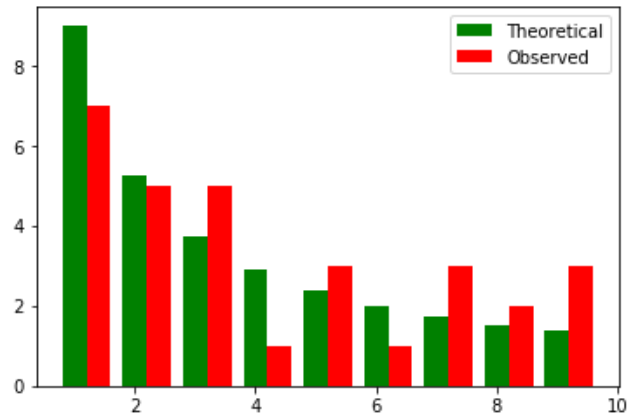
```
7
```




```
chi-squared test p-value:  0.35300574369451
```

In [40]:
```
## code for Karrubi

Mousavi=df_iran["Mousavi"]
#Rezai=df_iran["Rezai"]
#Karrubi=df_iran["Karrubi"]
#Mousavi=df_iran["Mousavi"]

first_digits = np.array([int(str((Mousavi.iloc[i]))[0]) for i in range(len(Mousavi))])
print(first_digits[0])

freqs = np.array([np.sum(first_digits==i) for i in range(1,10)])


def benford():
    return [np.log10(d+1)-np.log10(d) for d in range(1,10)]

theoretical_freqs = np.array([f * np.sum(freqs) for f in benford()])

# Plot two distros side-by-side:

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
# The third parameter below is the width of the bar, set
# such that we can squeeze another set of bars in:
rects1 = ax.bar(range(1,10), theoretical_freqs, 0.4, color='g')
# In the following, adding the bar width to the x coordinates
# position this set of bars side-by-side with the last:
rects2 = ax.bar([i+0.4 for i in range(1,10)], freqs, 0.4, color='r')
ax.legend((rects1[0], rects2[0]), ('Theoretical', 'Observed'))
plt.show()
#print(freqs)


# Computing p-value
chisq, pvalue = stats.chisquare(freqs, theoretical_freqs)
print('chi-squared test p-value: ', pvalue)
```

```
8
```




```
chi-squared test p-value:  0.6702036659602462
```

In [ ]:
```
# Write your code to solve Problem B here
# Feel free to insert additional code blocks as necessary
```

We found the p-values to be relatively high, being larger than .01. This means that we can accept the null hypothesis and our model is correct for the given data. The p-value of Ahmadinejad was .709, Rezai was .228, Karrubi was .353, and Mousavi was .670.

Part 2: Exploring Movie Lens Data

The is the same movie lens dataset from HW 4. As a reminder, it contains about 100,000 ratings from 1000 users on 1700 movies, made available by grouplen.org. This information is in three files: `u.user`, `u.data`, and `u.item`. Below, we import the relevant information into three dataframes (about the users, the ratings, and the movies). You are welcome to work directly with these dataframes, or to transform the data however you like for solving the problems in this part. Most of the data is self explanatory, but be sure to note that a given movie can be categorized as multiple genres (see the movies dataframe / `u.item` file below). To denote this, each row of the table corresponds to a given movie, and there is a column for every genre; there is a 1 in the column if the movie is of that genre, and a 0 otherwise.

In [43]:
```
df_users = pd.read_table('u.user', sep='|', names = ['user_id', 'age', 'sex', 'occupat
ion', 'zip'])
df_users.head()
```

Out[43]:

	user_id	age	sex	occupation	zip
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

In [44]:
```
df_ratings = pd.read_table('u.data', sep=',', names = ['user_id', 'movie_id', 'rating'
, 'timestamp'])
df_ratings.head()
```

Out[44]:

	user_id	movie_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

In [45]:
```
df_movies = pd.read_table('u.item', sep='|', encoding='latin',\
                          names = ['movie_id', 'movie_title', 'release_date', 'video_r
elease_date',\
                                   'imdb_url', 'unknown', 'action', 'adventure', 'ani
mation',\
                                   'children', 'comedy', 'crime', 'documentary', 'dra
ma', 'fantasy',\
                                   'film_noir', 'horror', 'musical', 'mystery', 'roma
nce', 'sci_fi',\
                                   'thriller', 'war', 'western'])
df_movies.head()
```

Out[45]:

	movie_id	movie_title	release_date	video_release_date	imdb_url	unknown	action	ac
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	1
2	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	0	0
3	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	0	1	0
4	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	0	0

5 rows × 24 columns

In this part, our goal will be to answer the following question. **Suppose a user rates the two movies To Be or Not to Be (1942) and Bad Taste (1987). What are the odds that this user is female versus male, based on the data given?** In other words, find the ratio of the probability that this user is female to the probability that this user is male, given that they rated both movies.

Problem A

The most straightforward method would just look at all users who have rated these two movies, compute the breakdown of the number of reviewers by gender, and use that ratio to predict the gender of the mysterious reviewer. Why won't this work?

In [ ]:
```
# Write your code to investigate the problem here
```

Write your answer to problem A here

Problem B

Now, assume that given a user's gender, the chances that the user rates any two given movies are independent. In other words, assume

$$\text{P(U rates movies A and B | U is female)} = \text{P(U rates A | U is female) P(U rates B | U is female)},$$

and the same goes for male. Given this assumption, use Bayes' Theorem to solve for the following ratio of probabilities in terms of quantities you can measure from the data.

$$\frac{\text{P(U is female | U rates movies A and B)}}{\text{P(U is male | U rates movies A and B)}}$$

Write your answer to problem B here

Problem C

Using the expression you derived in problem B, find the relative odds that a user who rates the two movies is female versus male.

In [ ]:
```
# Write your code to answer problem C here
```

In other words, under our assumption of independence, we find that the user is only about 11% as likely to be female as male, given that they rated the two movies.

Submitting Lab 4

1. Double check that you have written all of your answers along with your supporting work in this notebook. Make sure you save the complete notebook.
2. Double check that your entire notebook runs correctly and generates the expected output. To do so, you can simply select Kernel -> Restart and Run All.
3. You will download two versions of your notebook to submit, a .pdf and a .py. To create a PDF, we reccomend that you select File --> Download as --> HTML (.html). Open the downloaded .html file; it should open in your web broser. Double check that it looks like your notebook, then print a .pdf using your web browser (you should be able to select to print to a pdf on most major web browsers and operating systems). Check your .pdf for readability: If some long cells are being cut off, go back to your notebook and split them into multiple smaller cells. Also, make sure that it is a reasonable length; print statements which are truncated inside of the notebook may come to many pages in the pdf. To get the .py file from your notebook, simply select File -> Download as -> Python (.py).
4. Upload the .pdf to gradescope under lab 4 report and the .py to gradescope under lab 4 code. Only submit once per group, but be sure to add your partner using the group feature on gradescope (https://www.gradescope.com/help#help-center-item-student-group-members).