The Distribution of First Digits

In this lab, you will explore the distribution of first digits in real data. For example,

- the first digit of 52 is 5,
- the first digit of 30.8 is 3, and
- the first digit of 0.07 is 7.

In this lab, you will investigate the question: how frequently does each digit 1-9 appear as the first digit of the number?

Question 0

Make a prediction.

1. Approximately what percentage of the values do you think will have a *first* digit of 1? What percentage of the values do you think will have a first digit of 9?
2. Approximately what percentage of the values do you think will have a *last* digit of 1? What percentage of the values do you think will have a last digit of 9?

(Don't worry about being wrong. You will earn full credit for any justified answer.)

ENTER YOUR WRITTEN EXPLANATION HERE.

1. I believe that approximately 11% of the values in the data set will have a first digit of 1. This is because there are 9 possible outcomes for the first digit in each value (1-9), giving each exactly a 1/9 or about 11% probability of appearing as the first digit in the value. Because of this, I also believe that approximately 11% of values will have a first digit of 9, becauase these should have the same chance of appearing as the first digit.

2. Similar to question 1, I believe that approximately 11% of the values in the data set will have a last digit of 1 and therefore 11% will have a last digit of 9. This is because these same 9 digits compose the possible pool of outcomes, resulting in about an 11% chance of each outcome occurring as a last digit.

Question 1

The S&P 500 is a stock index based on the market capitalizations of large companies that are publicly traded on the NYSE or NASDAQ. The CSV file (https://dlsun.github.io/pods/data/sp500.csv) contains data from February 1, 2018 about the stocks that comprise the S&P 500. We will investigate the first digit distributions of the variables in this data set.

Read in the S&P 500 data. What is the observational unit in this data set (in other words, what does each row represent)? Is there a variable that is natural to use as the index? If so, set that variable to be the index. (You can do this using `DataFrame.set_index()` or by specifying `index_col=` when you read in the CSV.) Once you are done, display the `DataFrame`.

In [14]:
```
import pandas as pd
from plotnine import *
```

In [15]:
```
df = pd.read_csv("https://dlsun.github.io/pods/data/sp500.csv")
df
```

Out[15]:

	date	Name	open	close	volume
0	2018-02-01	AAL	$54.00	$53.88	3623078
1	2018-02-01	AAPL	$167.16	$167.78	47230787
2	2018-02-01	AAP	$116.24	$117.29	760629
3	2018-02-01	ABBV	$112.24	$116.34	9943452
4	2018-02-01	ABC	$97.74	$99.29	2786798
...	...	...	...	...	...
500	2018-02-01	XYL	$72.50	$74.84	1817612
501	2018-02-01	YUM	$84.24	$83.98	1685275
502	2018-02-01	ZBH	$126.35	$128.19	1756300
503	2018-02-01	ZION	$53.79	$54.98	3542047
504	2018-02-01	ZTS	$76.84	$77.82	2982259

505 rows × 5 columns

ENTER YOUR WRITTEN EXPLANATION HERE.

The observational unit in the dataset is each unique company. This is because collectively, the data set is a complete stock market index displaying the market performance of each of the companies that have traded on either the NYSE or the NASDAQ. Therefore, each individual row in this data set represents the information for a specific company's stocks.

The variable that is most natural to use as the index in this dataset is the name of the company. This is because this is a unique value different for each company that can constitute a row in a table and hence be located in Python.

In [16]:
```
df.set_index("Name") #Setting the Name variable as an index
```

Out[16]:

Name	date	open	close	volume
AAL	2018-02-01	$54.00	$53.88	3623078
AAPL	2018-02-01	$167.16	$167.78	47230787
AAP	2018-02-01	$116.24	$117.29	760629
ABBV	2018-02-01	$112.24	$116.34	9943452
ABC	2018-02-01	$97.74	$99.29	2786798
...	...	...	...	...
XYL	2018-02-01	$72.50	$74.84	1817612
YUM	2018-02-01	$84.24	$83.98	1685275
ZBH	2018-02-01	$126.35	$128.19	1756300
ZION	2018-02-01	$53.79	$54.98	3542047
ZTS	2018-02-01	$76.84	$77.82	2982259

505 rows × 4 columns

Question 2

We will start by looking at the `volume` column. This variable tells us how many shares were traded on that date.

Extract the first digit of every value in this column. (*Hint:* First, turn the numbers into strings. Then, use the text processing functionalities of `pandas` to extract the first character of each string.) Make an appropriate visualization to display the distribution of the first digits. (*Hint:* Think carefully about whether the variable you are plotting is quantitative or categorical.)

How does this compare with what you predicted in Question 0?

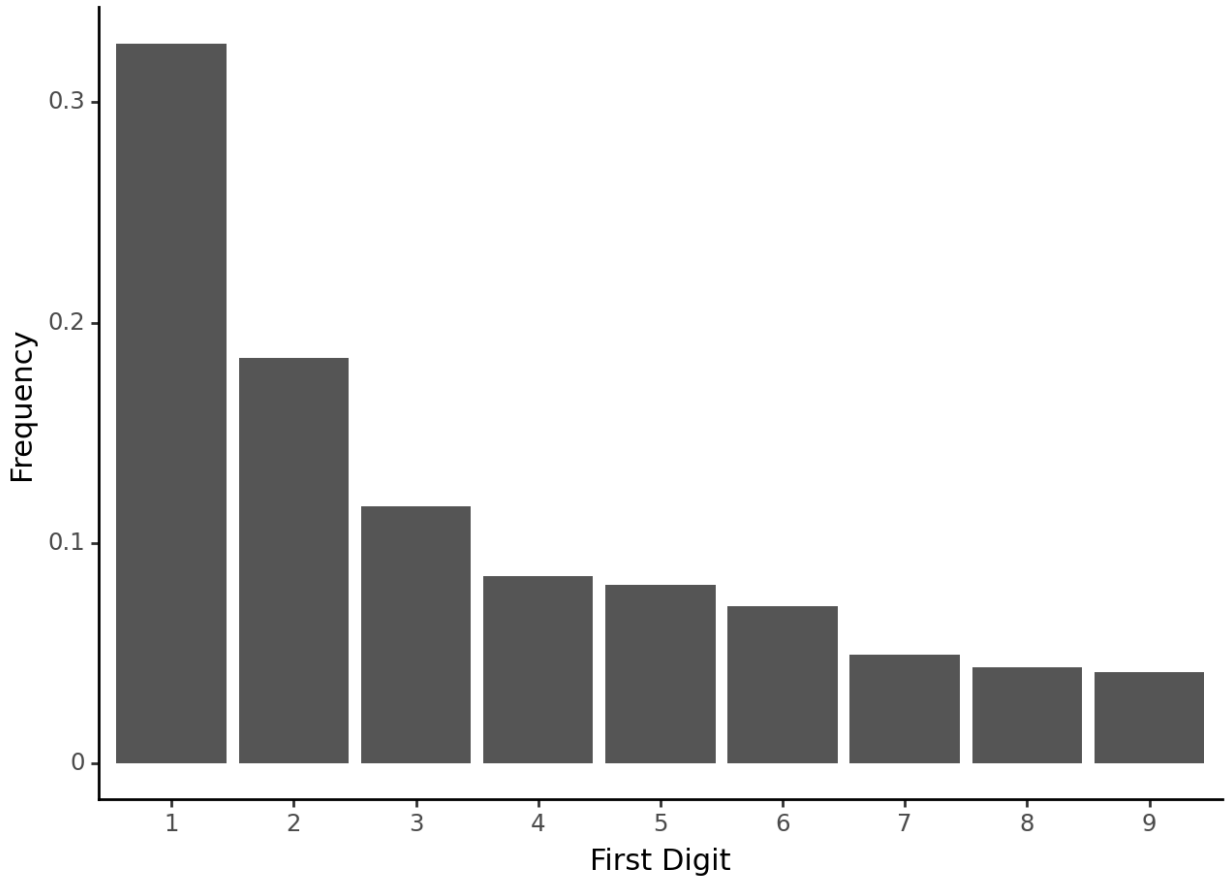
In [17]:
```
df["volume"] = df["volume"].astype("str") #Turning volume variable/column into
df["first_digit"] = df["volume"].str[0] #Finding the first digit using indexing
first_digit_dist = df["first_digit"].value_counts(normalize=True).to_frame().r
first_digit_dist
```

Out[17]:

	first_digit	proportion
0	1	0.326733
1	2	0.184158
2	3	0.116832
3	4	0.085149
4	5	0.081188
5	6	0.071287
6	7	0.049505
7	8	0.043564
8	9	0.041584

In [18]:
```
first_digit_dist["proportion"] = first_digit_dist["proportion"].astype("float"
```

In [19]:
```
(ggplot(first_digit_dist, aes(x = "first_digit", y = "proportion"))
+ geom_col()
+ xlab("First Digit")
+ ylab("Frequency")
+ theme_classic()
)
```



Out[19]:
```
<Figure Size: (640 x 480)>
```

ENTER YOUR WRITTEN EXPLANATION HERE.

The graph visually displaying the proportion and frequency with which each digit occurs as the first digit in the volume value reveals stark differences from my original prediction. Intially, I predicted that the distribution of first digits would have an approximately even spread and that each of the digits would have about an 11% chance and frequency of occurrence to be the first digit. However, the data establishes that in fact, the number 1 occurred as a first digit nearly 33% of the time, which is signiticantly greater than the succeeding digits. The frequency with which each digit appears as the first digit in the volume also decreases as the digit gets greater in value, establishing a linear and proportional relationship between a digit's value and the frequency with which it appears as the first digit.

Question 3

Now, repeat Question 2, but for the distribution of *last* digits. Again, make an appropriate visualization and compare with your prediction in Question 0.

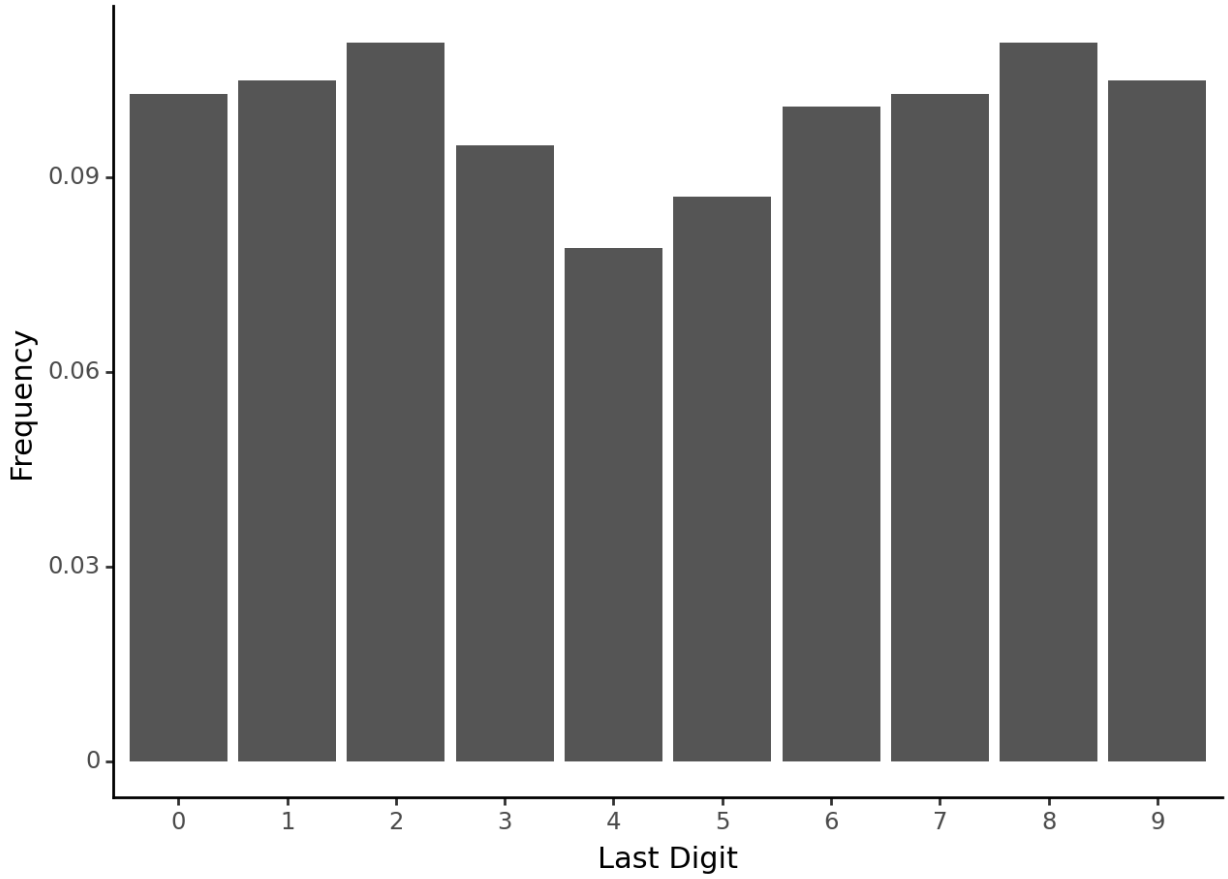
In [20]:
```
df["volume"] = df["volume"].astype("str") #Turning volume column into a string
df["last_digit"] = df["volume"].str[-1] #Indexing last digit
last_digit_dist = df["last_digit"].value_counts(normalize=True).to_frame().res
last_digit_dist
```

Out[20]:

	last_digit	proportion
0	8	0.110891
1	2	0.110891
2	9	0.104950
3	1	0.104950
4	7	0.102970
5	0	0.102970
6	6	0.100990
7	3	0.095050
8	5	0.087129
9	4	0.079208

In [21]:
```
last_digit_dist["proportion"] = last_digit_dist["proportion"].astype("float") #
```

In [22]:
```
(ggplot(last_digit_dist, aes(x = "last_digit", y = "proportion"))
+ geom_col()
+ xlab("Last Digit")
+ ylab("Frequency")
+ theme_classic()
)
```



Out[22]: <Figure Size: (640 x 480)>

ENTER YOUR WRITTEN EXPLANATION HERE.

This visual distribution of the last digits in each of the volume values in the S&P 500 dataset reveals that while my initial prediction was not entirely accurate, there true frequencies with which each digit appears as the last are significantly more evenly distributed than those of the first digits. Originally, I predicted that each of the digits would be the last approximately 11% of the time because there were 9 digits and each would have the same chance of occurring.

While this was not correct, the frequencies with which digits appear as the last range from about 7% of the time (the digit 4) to 11% of the time (the digits 2 and 8). This establishes a relatively even spread of frequencies, as opposed to the distribution of first digits.

Question 4

Maybe the `volume` column was just a fluke. Let's see if the first digit distribution holds up when we look at a very different variable: the closing price of the stock. Make a visualization of the first digit distribution of the closing price (the `close` column of the `DataFrame`). Comment on what you see.

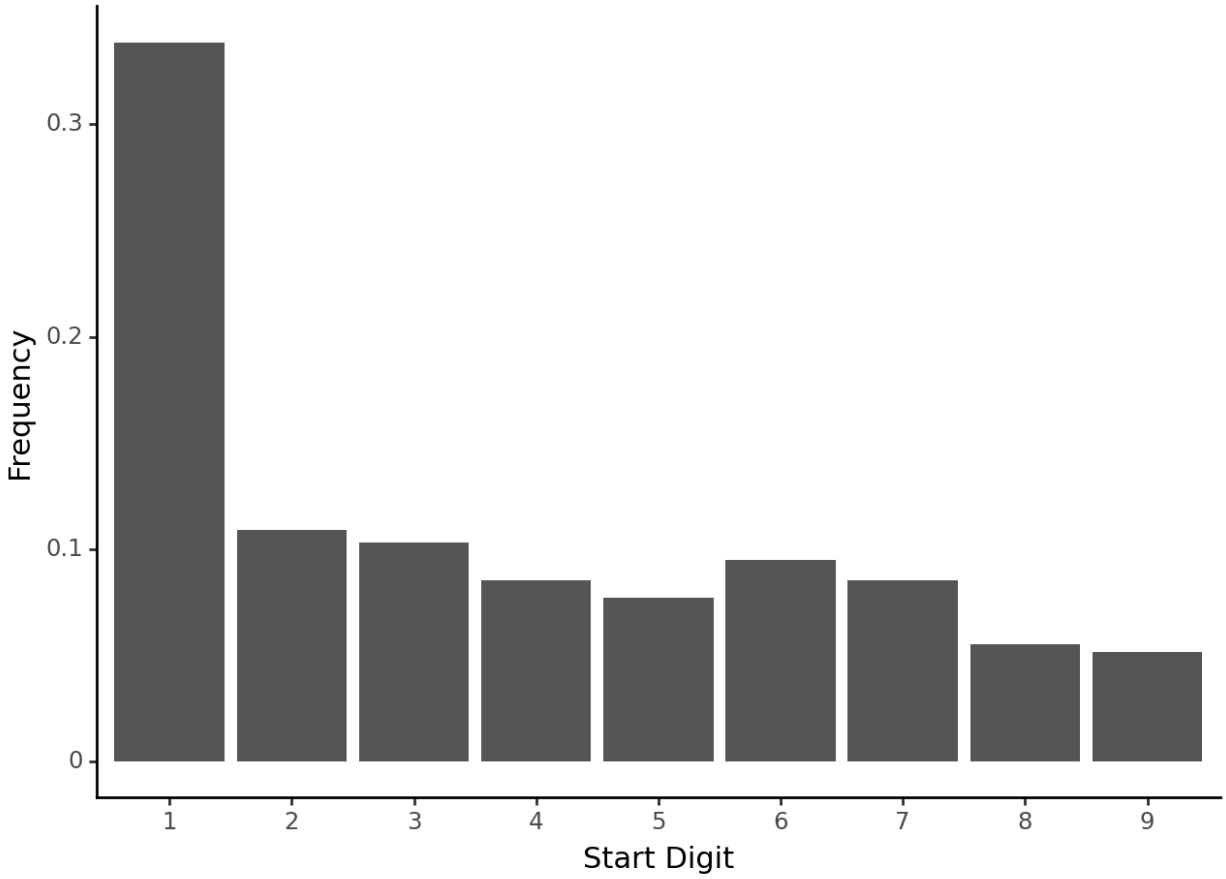
(*Hint:* What type did `pandas` infer this variable as and why? You will have to first clean the values using the text processing functionalities of `pandas` and then convert this variable to a quantitative variable.)

In [23]:
```
df["start_digit"] = df["close"].astype("str").str[1] #Indexing by 1 to remove 
start_digit_dist = df["start_digit"].value_counts(normalize=True).to_frame().r
start_digit_dist
```

Out[23]:

	start_digit	proportion
0	1	0.338614
1	2	0.108911
2	3	0.102970
3	6	0.095050
4	4	0.085149
5	7	0.085149
6	5	0.077228
7	8	0.055446
8	9	0.051485

In [24]:
```
(ggplot(start_digit_dist, aes(x = "start_digit", y = "proportion"))
+ geom_col()
+ xlab("Start Digit")
+ ylab("Frequency")
+ theme_classic()
)
```



Out[24]: <Figure Size: (640 x 480)>

ENTER YOUR WRITTEN EXPLANATION HERE.

The distribution of first digits is maintained in the values for the closing price variable. This is seen in that the distributions for both the volume and closing price variables, the digit 1 appeared as the first digit approximately 33% of the time with the other digits having a relatively even spread of frequency. In fact, the distribution of frequencies with which the digits 2-9 appear as the first digit in the closing price is even more similarly (evenly) distributed than that of the first digits in the volume values.

Pandas inferred the first digit as a categorical variable most likely because the closing price variables were turned into strings. This means that Pandas understood we were indexing and extracting a value from a string for this variable and we already had the quantitative variable of the proportion (or "Frequency", as it is labelled) on the y-axis.

Submission Instructions

- Restart this notebook and run the cells from beginning to end.
  - Go to Runtime > Restart and Run All.

In [ ]:
```
# @markdown Run this cell to download this notebook as a webpage, `_NOTEBOOK.h

import google, json, nbformat
```

```
# Get the current notebook and write it to _NOTEBOOK.ipynb
raw_notebook = google.colab._message.blocking_request("get_ipynb",
                                                      timeout_sec=30)["ipynb"]
with open("_NOTEBOOK.ipynb", "w", encoding="utf-8") as ipynb_file:
  ipynb_file.write(json.dumps(raw_notebook))

# Use nbconvert to convert .ipynb to .html.
!jupyter nbconvert --to html --log-level WARN _NOTEBOOK.ipynb

# Download the .html file.
google.colab.files.download("_NOTEBOOK.html")
```

- Open `_NOTEBOOK.html` in your browser, and save it as a PDF.
  - Go to File > Print > Save as PDF.
- Double check that all of your code and output is visible in the saved PDF.
- Upload the PDF to Gradescope.
  - Please be sure to select the correct pages corresponding to each question.