FA691-HW1

January 22, 2023

1 FA691 Homework 1

2 Due: Wednesday, January 25 @ 11:59PM

Name: Ryan Shea

Date: 2023-01-22

[1]:
```
import numpy as np
import matplotlib.pyplot as plt

# Set seed of random number generator
CWID = 10445281 #Place here your Campus wide ID number, this will personalize
#your results, but still maintain the reproduceable nature of using seeds.
#If you ever need to reset the seed in this assignment, use this as your seed
#Papers that use -1 as this CWID variable will earn 0's so make sure you change
#this value before you submit your work.
personal = CWID % 10000
np.random.seed(personal)
```

2.1 Question 1 (20pt)

2.1.1 Question 1.1

An urn contains four type A coins and two type B coins. When a type A coin is flipped, it comes up heads with probability 1/3, whereas when a type B coin is flipped, it comes up heads with probability 1/2. A coin is randomly chosen (uniformly) from the urn and flipped. Given that the flip landed on heads, what is the probability that it was a type B coin?

Hint: Recall Bayes' theorem.

(Note that the following fields can be added wherever you desire to show a solution. You can use the Markdown blocks for a written response, and the Code blocks for showing python code and its output. Some questions will require just one, and some both. I will not always provide you with these, but you can add them at your discretion wherever necessary. If it makes sense to do the python code first then that's fine. If you want to include multiple of each, that's ok too. Do what you feel is necessary to answer the question fully.)

Bayes Theorem:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

In this case, A is the probablity that it was a type B coin and B is the fact that it landed on heads.

$$P(B|A)$$

B is the fact that it landed on heads. We know that the probability of a type B coin landing on heads is 1/2. Therefore,

$$P(B|A) = 1/2$$

$$P(A)$$

A is the probability that it was a type B coin. We know that there are 2 type B coins in the urn (out of 6). Therefore,

$$P(A) = 2/6 = 1/3$$

$$P(B)$$

B is the fact that it landed on heads. We know that the probability of a type A coin landing on heads is 1/3 and the probability of a type B coin landing on heads is 1/2. Therefore,

$$P(B) = (1/3 * 2/3) + (1/2 * 1/3) = 7/18$$

Putting it all together, we get:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} = \frac{1/2 * 1/3}{7/18} = \frac{9}{21}$$

2.1.2 Question 1.2

Simulate this system by sequentially sampling a coin and a flip. From 10,000 repeated simulations, what percentage of heads results came from a type B coin? Comment on your answer in comparison with the result you found in Question 1.1.

[2]:
```
def system(trials):
    coins = np.array(["A", "A", "A", "A", "B", "B"])

    total_heads = 0
    total_b = 0
    prob = np.zeros(trials)

    for i in range(trials):
        pick = np.random.choice(coins)

        if pick == "B":
            res = np.random.choice([0, 1]) # 1 in 2 chance of being 1, or heads
            if res == 1: # if heads
                total_heads += 1
                total_b += 1
```

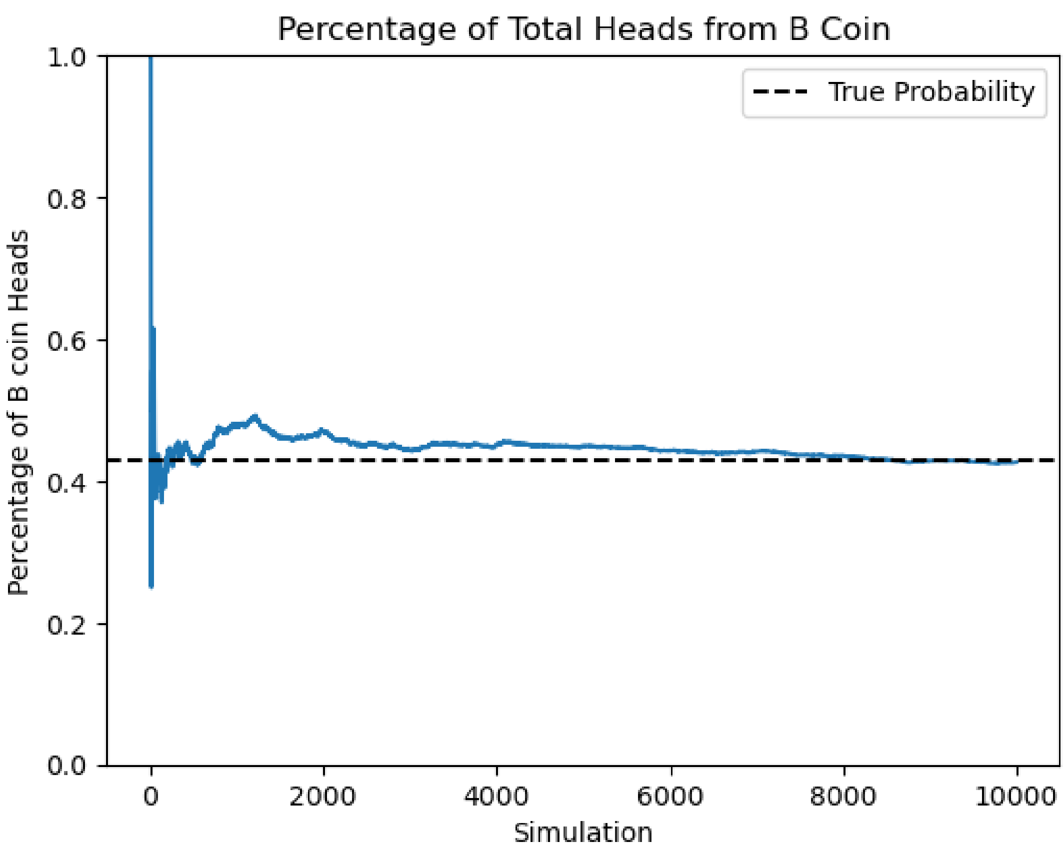
```
        else: # if the pick == "A"
            res = np.random.choice([0, 0, 1]) # 1 in 3 now
            if res == 1:
                total_heads += 1
        prob[i] = 0 if total_heads == 0 else total_b / total_heads

    return prob[-1], prob # last probability, entire probability array

simulated_prob, prob_array = system(10000)
print(f"Simulated Probabilty: {simulated_prob}")

plt.plot(prob_array)
plt.title("Percentage of Total Heads from B Coin")
plt.xlabel("Simulation")
plt.ylabel("Percentage of B coin Heads")
plt.axhline(9/21, xmin=0, linestyle="--", color="black", label="True␣
 ↪Probability")
plt.ylim(top=1, bottom=0)
plt.legend()
plt.show()
```

```
Simulated Probabilty: 0.42646685293370584
```



The Law of Large Numbers holds up here: as the number of trials increases, the percentage of B coin heads approaches the true probability of 9/21.

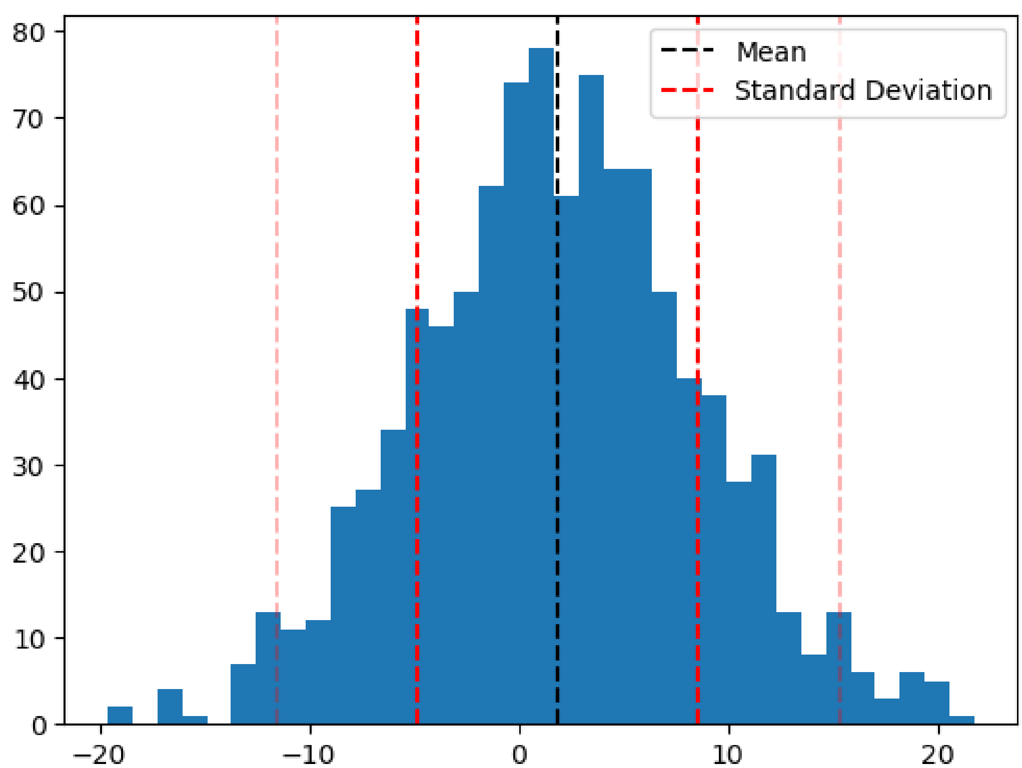
2.2 Question 2 (10pt)

2.2.1 Question 2.1

Generate a vector `x` containing 1,000 realizations of a random normal variable with mean 2 and variance 7. Plot a histogram of `x` using 35 bins.

[3]:
```
x = np.random.normal(loc=2, scale=7, size=1000)

plt.hist(x, bins=35)
plt.axvline(x.mean(), ymin=0, color='black', linestyle='--', label="Mean")
plt.axvline(x.mean() + x.std(), ymin=0, color='red', linestyle='--',␣
 ↪label="Standard Deviation")
plt.axvline(x.mean() - x.std(), ymin=0, color='red', linestyle='--')
plt.axvline(x.mean() + 2*x.std(), ymin=0, color='red', linestyle='--', alpha=0.
 ↪3)
plt.axvline(x.mean() - 2*x.std(), ymin=0, color='red', linestyle='--', alpha=0.
 ↪3)
plt.legend()
plt.show()
```



2.2.2 Question 2.2

Calculate the mean and standard deviation of these 1,000 values. Do your answers make sense?

[4]:
```
print(f"Mean: {x.mean()}\nStandard Deviation: {x.std()}")
```

```
Mean: 1.8690932188568559
Standard Deviation: 6.715481092094306
```

These make sense as the mean is close to 2 and the standard deviation is close to 7. If I were to increase the observations they would come even closer.

2.2.3 Question 2.3

Take out 10 random samples of 500 observations each (with replacement). Create a vector of the means of each sample. Calculate the mean of the sample means and the standard deviation of the sample means. What do you observe about these results?

[5]:
```
mu = np.zeros(10)
for i in range(len(mu)):
    mu[i] = np.random.choice(x, size=500, replace=True).mean()

print(mu, end='\n\n')
print(f"Mean: {mu.mean()}\nStandard Deviation: {mu.std()}")
```

```
[1.32764314 1.91513698 1.47676879 1.94763908 1.97529663 1.87589879
 1.95997702 1.85321002 1.726897   2.05930537]

Mean: 1.811777281313202
Standard Deviation: 0.22333002709822908
```

The mean is roughly equal to the population mean with a small standard deviation. Because of that, it makes sense.

2.3 Question 3 (10pt)

2.4 Question 3.1

Download stock price data for 4 stocks of your choice from January 1, 2020 through December 31, 2022. (All chosen stocks must have price data for the entire time period.) Find the mean and standard deviation of the daily log returns for each stock in your data set.

[6]:
```
import yfinance as yf
import pandas as pd

stocks = ['SPY', 'XLK', 'XLF', 'XLP']
# sp500, tech, financials, cons staples
```

```
df = pd.DataFrame()

for ticker in stocks:
    df[ticker] = yf.download(ticker, "2020-01-01", "2022-12-31")["Adj Close"]

print(df.head())

ret = df.apply(lambda x: np.log(x / x.shift(1))).dropna()

ret.head()
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
                   SPY        XLK        XLF        XLP
Date
2020-01-02  309.694977  90.642479  29.158108  57.774712
2020-01-03  307.349762  89.623375  28.848516  57.682240
2020-01-06  308.522308  89.836891  28.829748  57.802452
2020-01-07  307.654907  89.798073  28.642120  57.358597
2020-01-08  309.294617  90.758949  28.829748  57.571274
```

[6]:

```
                 SPY       XLK       XLF       XLP
Date
2020-01-03 -0.007601 -0.011307 -0.010674 -0.001602
2020-01-06  0.003808  0.002380 -0.000651  0.002082
2020-01-07 -0.002815 -0.000432 -0.006529 -0.007708
2020-01-08  0.005316  0.010644  0.006529  0.003701
2020-01-09  0.006757  0.011272  0.006164  0.007042
```

[7]:

```
print("Mean:")
print(ret.mean())
print("\nStandard Deviation")
print(ret.std())
```

```
Mean:
SPY    0.000279
XLK    0.000420
XLF    0.000211
XLP    0.000338
dtype: float64

Standard Deviation
SPY    0.015824
XLK    0.020199
XLF    0.020023
XLP    0.012657
```

```
dtype: float64
```