Sina Mahbobi
9763428651

Project 1

EE 511
January 22, 2020

Question 1

To simulate a Bernoulli trial, I used numpy's random.random() function to generate a random number between zero and one. I assigned the any value less than 0.5 to Heads, and any value greater than or equal to 0.5 tails.

The number of heads is 24.

The longest run of heads is 4.

Question 1 part a

Repeat the above experiment 20, 100, 200, and 1000 times. Generate a histogram for each showing the number of heads in 50 flips. Comment on the limit of the histogram.

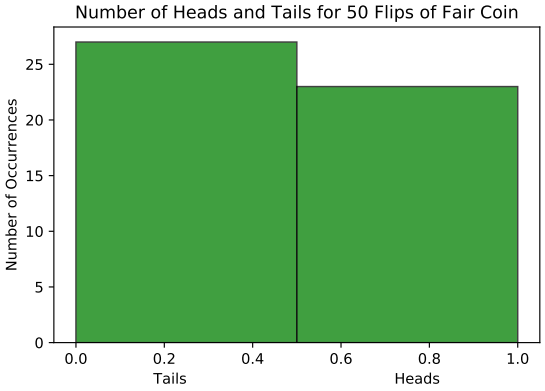

Figure 1: Histogram of Bernoulli Outcomes.

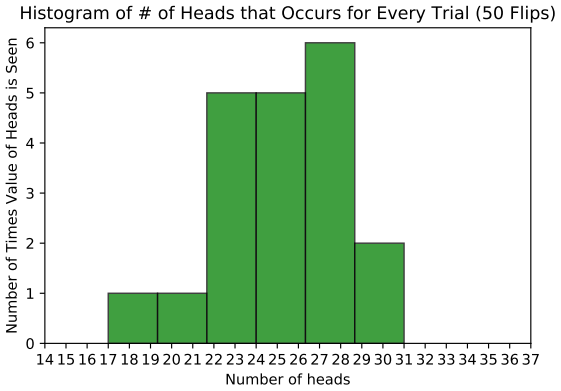

Figure 2: Histogram of 20 Trials.

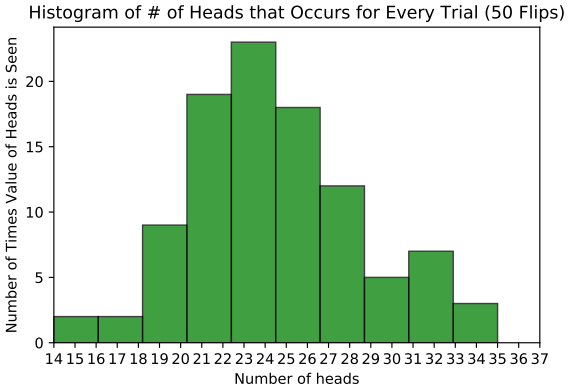

Figure 3: Histogram of 100 Trials.

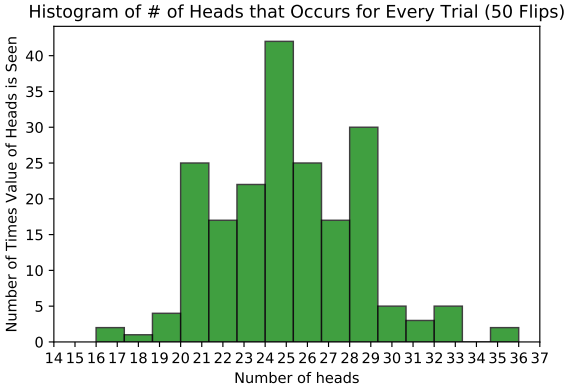

Figure 4: Histogram of 200 Trials.

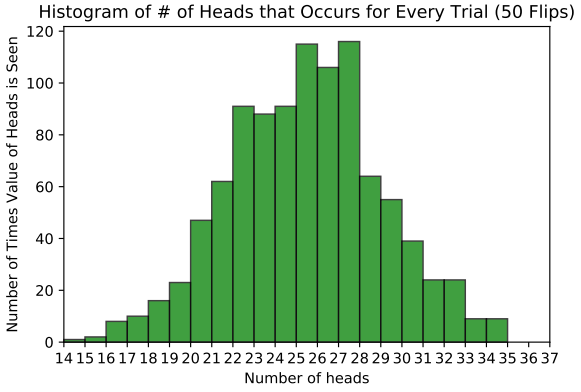

Figure 5: Histogram of 1000 Trials.

Project 1

```
# Code From Question 1

import numpy as np
import matplotlib.pyplot as plt


def flip(p):
    return True if np.random.random() < p else False

# True = Heads (p > 0.5), False = Tails (p <= 0.5 )

num_heads = 0
num_flips = 50
num_repeats = 1000  #20, 100, 200, 1000
long_heads = 0
temp_heads = 0
heads_record = []

# Q1: Simulate tossing a fair coin (a Bernoulli trial) 50 times.  Count the n
# Record the longest run of heads.

for i in range(0,num_flips):
    if flip(0.5):
        num_heads +=1
        temp_heads += 1
        heads_record.append(1)
        if temp_heads >= long_heads:
            long_heads = temp_heads
    else:
        temp_heads = 0
        heads_record.append(0)

print("The number of heads is {}" .format(num_heads))
print ("The longest run of heads {}" .format(long_heads))

f0 = plt.figure(1)
plt.hist(heads_record, bins = 2,  facecolor="g", edgecolor="black", alpha=0.7
plt.xlabel('Tails                                                    Heads')
plt.ylabel("Number of Occurrences")
plt.title('Number of Heads and Tails for 50 Flips of Fair Coin')
f0.savefig('bernoulli_trial.pdf')
# Q1 A: Repeat the above experiment 20, 100, 200, and 1000 times.
```

```
Generate a histogram for each showing the
# number of heads in 50 flips.  Comment on the limit of the histogram.

heads_arr = []
for i in range(0, num_repeats):
    num_heads = 0
    for j in range(0,num_flips):
        if flip(0.5):
            num_heads +=1
    heads_arr.append(num_heads)


f1 = plt.figure(2)

bins = np.arange(51+1)-0.5
plt.xlabel("Number of heads")
plt.hist(heads_arr, bins = bins , facecolor="g", edgecolor="black", alpha=0.7
plt.xticks(range(51))
plt.xlim([14, 37])
plt.ylabel("Number of Times Value of Heads is Seen")
plt.title("Histogram of # of Heads that Occurs for Every Trial (50 Flips)")
plt.show()
f1.savefig('num_repeats1000.pdf')
```

From this code and the respective graphs, it can be seen that the greater the trial runs, the more the histogram starts to look like a standard distribution curve. The values of the number of heads also get compressed closer to the expect value, which is 25 heads for every 50 flips.

Question 2

Simulate tossing a biased coin 200 times where P[HEAD] = 0.80. Count the number of heads. Record the longest run of heads. Generate a histogram for the Bernoulli outcomes.

```
#Code for Question 2

import numpy as np
import matplotlib.pyplot as plt
```

```
def flip(p):
    return True if np.random.random() < p else False

# True = Heads (p > 0.5), False = Tails (p <= 0.5 )

num_heads = 0
num_flips = 200
long_heads = 0
temp_heads = 0
heads_record = []
num_repeats = 10000
# Q1: Simulate tossing a fair coin (a Bernoulli trial) 50 times.  Count the n
# Record the longest run of heads.

for i in range(0,num_flips):
    if flip(0.8):
        num_heads +=1
        temp_heads += 1
        heads_record.append(1)
        if temp_heads >= long_heads:
            long_heads = temp_heads
    else:
        temp_heads = 0
        heads_record.append(0)

print("The number of heads is {}" .format(num_heads))
print ("The longest run of heads {}" .format(long_heads))

f0 = plt.figure(1)
plt.hist(heads_record, bins = 2,  facecolor="orange", edgecolor="black", alph
plt.xlabel('Tails                                                    Heads')
plt.ylabel("Number of Occurrences")
plt.title('Number of Heads and Tails for 200 Flips of Biased Coin')
f0.savefig('BiasedCoin_200.pdf')

heads_arr = []
for i in range(0, num_repeats):
    num_heads = 0
    for j in range(0,num_flips):
        if flip(0.8):
            num_heads +=1
```

```
    heads_arr.append(num_heads)


f1 = plt.figure(2)

bins = np.arange(202)
plt.xlabel("Number of heads")
plt.hist(heads_arr, bins, facecolor="g", edgecolor="black", alpha=0.75)
#plt.xticks(range(203))
plt.xlim([135, 185])
plt.ylabel("Number of Times Value of Heads is Seen")
plt.title("Histogram of # of Heads that Occurs for Every Trial (200 Flips) fo
plt.show()
f1.savefig('num_repeats10000.pdf')
```

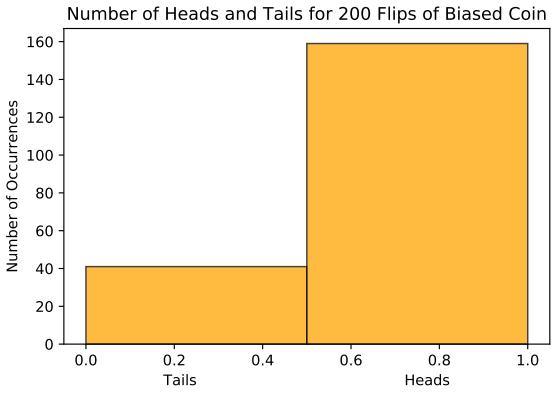

Figure 6: Histogram of 200 Flips.

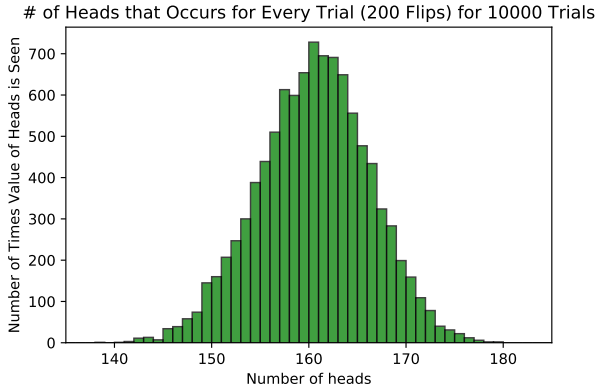

Figure 7: Histogram of 10000 Trials.

Question 3

Code For Question 3

```
import numpy as np
import matplotlib.pyplot as plt

#3. Simulate tossing a fair coin 100 times. Generate a histogram showing the

def flip(p):
    return True if np.random.random() < p else False

num_heads = 0
num_flips = 100
heads_run_length = 0
heads_runs = []




heads_run_length = 0
for j in range(0,num_flips):
    if flip(0.5):
        heads_run_length +=1
    elif heads_run_length != 0:
        #print(heads_run_length)
        heads_runs.append(heads_run_length)
        heads_run_length = 0
```

```
p = plt.figure(1)
bins = np.arange(51)-.5
plt.xlabel("Heads Run Length")
plt.hist(heads_runs, bins = bins, facecolor= "blue", edgecolor="black", alpha
plt.xticks(range(16))
plt.xlim([.5, 12])
plt.ylabel("Value of Heads Run Length Observed")
plt.title("Lengths of Heads Run that Occur in One Trial (100 Flips)")
plt.show()
p.savefig('heads_run_100.pdf')
```

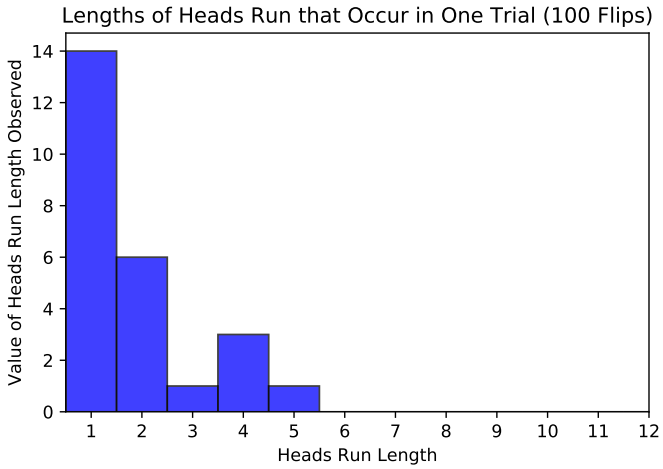

Figure 8: Histogram of 100 Flips.

Question 4

Code For Question 4

```
import numpy as np
import random
import matplotlib.pyplot as plt

def flip(p):
    return True if np.random.random() < p else False


heads_goal = int(input('Please type in a positive number of heads: '))
```

```
num_heads = 0
num_flips = 0


while num_heads < heads_goal:
    if flip(0.5):
        num_heads += 1
        num_flips += 1
    else:
        num_flips += 1


print("Number of heads: {}" .format(num_heads))
print("Number of tosses needed: {}" .format(num_flips))
```

Please type in a positive number of heads: 5

Number of heads: 5

Number of tosses needed: 6

Please type in a positive number of heads: 10

Number of heads: 10

Number of tosses needed: 16

Please type in a positive number of heads: 100

Number of heads: 100

Number of tosses needed: 203

Please type in a positive number of heads: 1000

Number of heads: 1000

Number of tosses needed: 1990

It is evident that as the number of tosses gets larger and larger, the number of tosses needed to get the number of heads requested approaches double that of the heads needed. In other words, the number of heads requested divided by the tosses needed approaches 50 percent.