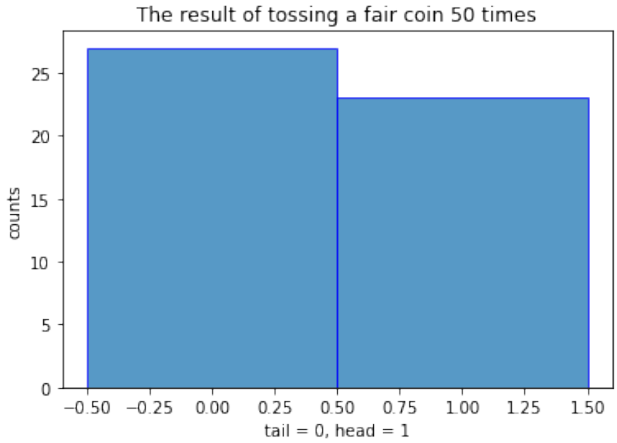
Project #1 Report

EE 511: Spring 2020
Wenjun Li; wenjunli@usc.edu; ID: 8372761120
Submission Date: Jan.21

1. **Simulate tossing a fair coin (a Bernoulli trial) 50 times. Count the number of heads. Record the longest run of heads.**

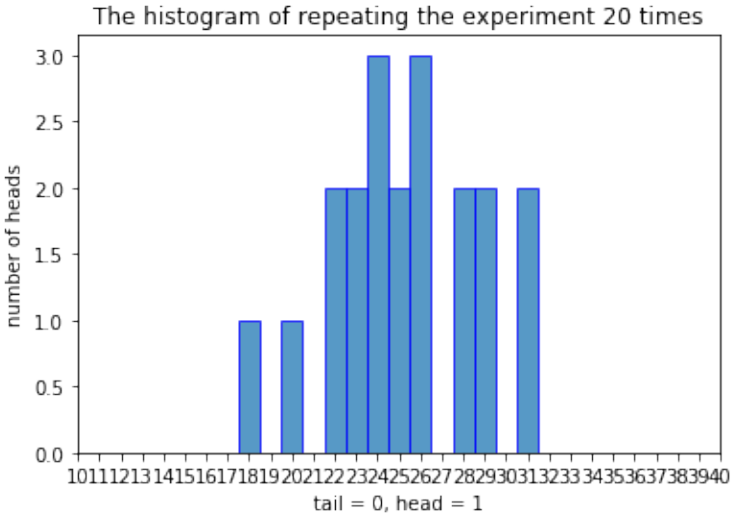
   Number of heads is 23.

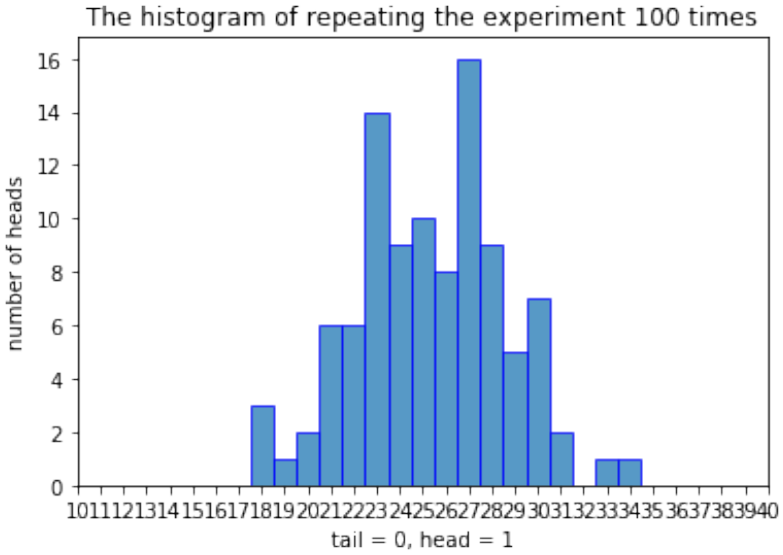
   The longest run of heads is 6.

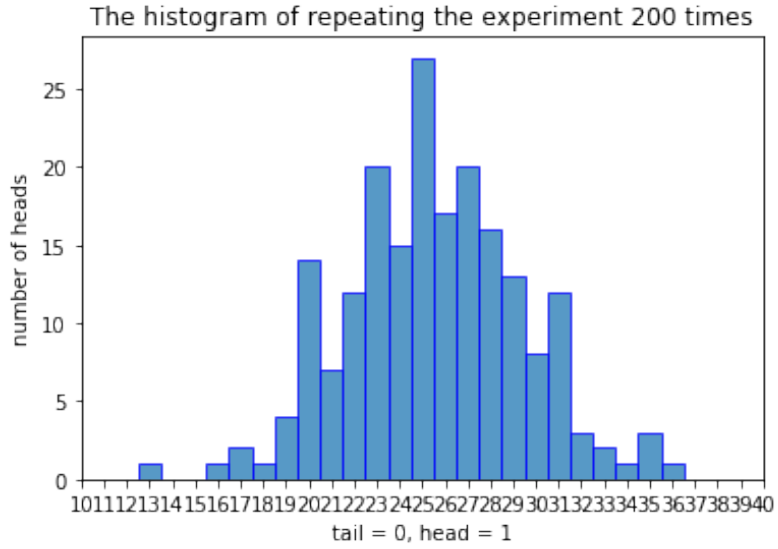


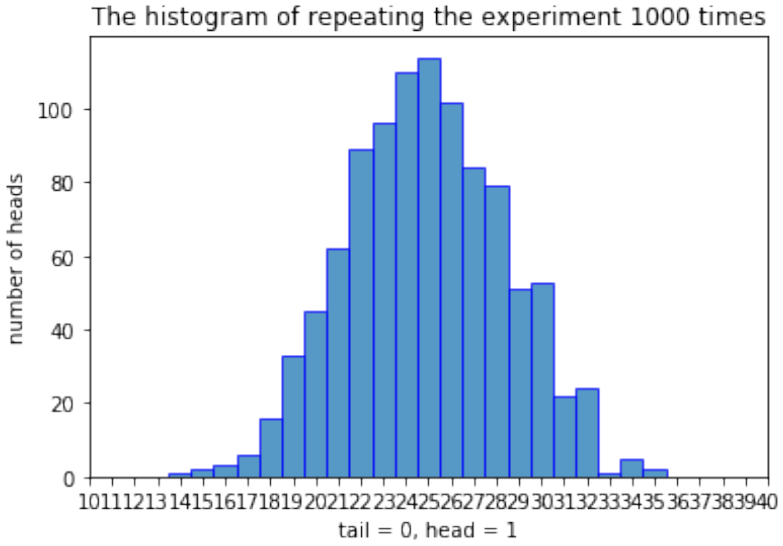
   a. **Repeat the above experiment 20, 100, 200, and 1000 times. Generate a histogram for each showing the number of heads in 50 flips. Comment on the limit of the histogram.**

   The four plots are shown below. As the repetition increases, the limit of the distribution centers at 25. For Bernoulli random variables, the distribution has a mean $\mu = np$ and a variance $\sigma^2 = npq$, where n is the number of trails, p is the positive probability, q is the negative probability. Hence, $\mu = 50 * 0.5 = 25$.




The histogram of repeating the experiment 100 times
number of heads
tail = 0, head = 1


The histogram of repeating the experiment 200 times
number of heads
tail = 0, head = 1


The histogram of repeating the experiment 1000 times
number of heads
tail = 0, head = 1

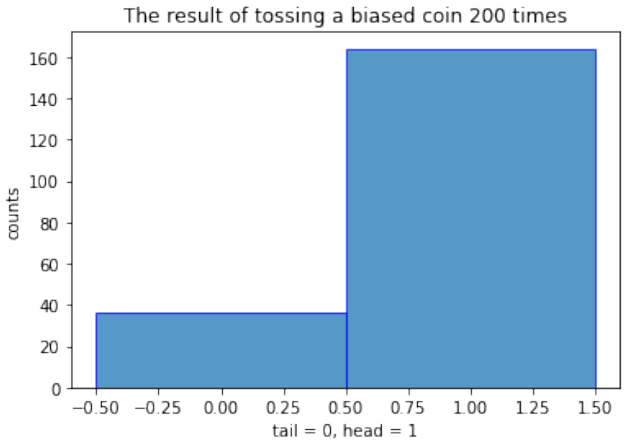
```python
import numpy as np
import matplotlib.pyplot as plt


N_flips = 50
N_heads = 0
history = np.empty([N_flips, 1])
longest_heads = 0

# generate coin-flip sequence
for i in range(0, N_flips):
    if np.random.random() >= 0.5:       # fair coin
        N_heads += 1
        history[i] = 1

    else:
        history[i] = 0

# check the longest run of heads
temp = 0
for i in range(0, N_flips-1):
    if history[i+1] == 1:
        temp += 1
        if temp > longest_heads:
            longest_heads = temp
    else:
        temp = 0

# print results
print("Number of heads is {}".format(N_heads))
print("The longest run of heads is {}".format(longest_heads))

# plot
plt.figure(1)
bins = np.arange(3) - 0.5
plt.hist(history, bins, edgecolor="b", alpha=0.75)
plt.title("The result of tossing a fair coin 50 times")
plt.xlabel("tail = 0, head = 1")
plt.ylabel("counts")
plt.show()


# repeat the experiment for 20 times
```

```
N_repeat = 20    # 20, 100, 200, 1000; switch the number to generate different results
N_heads_hist = np.empty(N_repeat)

for i in range(0, N_repeat):
    N_heads = 0
    for j in range(0, N_flips):
        if np.random.random() >= 0.5:           # fair coin
            N_heads += 1
            history[j] = 1
        else:
            history[j] = 0
    N_heads_hist[i] = N_heads

# plot
plt.figure(2)
bins = np.arange(N_flips+1) - 0.5
plt.hist(N_heads_hist, bins, edgecolor='b', alpha=0.75)
plt.xticks(range(51))
plt.xlim(10, 40)
plt.title("The histogram of repeating the experiment {} times".format(N_repeat))
plt.xlabel("tail = 0, head = 1")
plt.ylabel("number of heads")
plt.show()
```

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.**

Number of heads is 164.

The longest run of heads is 16.

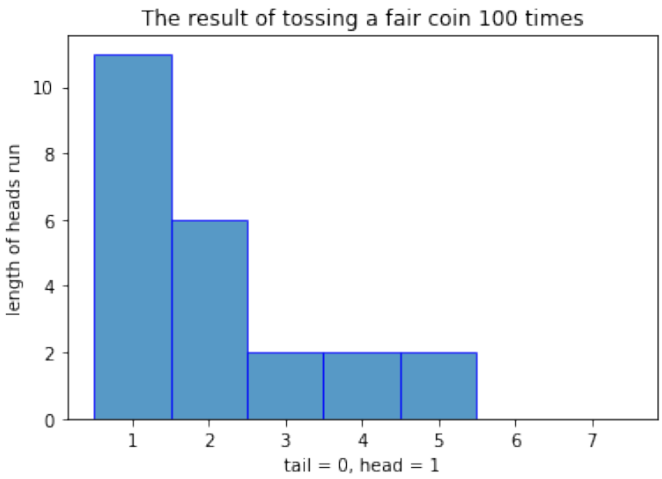


In this experiment, the mean $\mu = np = 200 * 0.8 = 160$.

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


N_flips = 200
N_heads = 0
history = np.empty([N_flips, 1])
longest_heads = 0

# generate coin-flip sequence
for i in range(0, N_flips):
    if np.random.random() >= 0.2:       # biased coin
        N_heads += 1
        history[i] = 1
    else:
        history[i] = 0

# check the longest run of heads
temp = 0
for i in range(0, N_flips-1):
    if history[i+1] == 1:
        temp += 1
        if temp > longest_heads:
            longest_heads = temp
    else:
        temp = 0

# print results
print("Number of heads is {}".format(N_heads))
print("The longest run of heads is {}".format(longest_heads))

# plot
plt.figure(1)
bins = np.arange(3) - 0.5
plt.hist(history, bins, edgecolor="b", alpha=0.75)
plt.title("The result of tossing a biased coin 200 times")
plt.xlabel("tail = 0, head = 1")
plt.ylabel("counts")
plt.show()
```

3. **Simulate tossing a fair coin 100 times. Generate a histogram showing the heads run lengths.**

The plot is shown as below.




```
1.  import numpy as np
2.  import matplotlib.pyplot as plt
3.
4.
5.  N_flips = 100
6.  N_heads = 0
7.  history = np.empty([N_flips, 1])
8.  longest_heads = 0
9.  length_of_heads_run = []
10.
11. # generate coin-flip sequence
12. for i in range(0, N_flips):
13.     if np.random.random() >= 0.5:       # fair coin
14.         N_heads += 1
15.         history[i] = 1
16.     else:
17.         history[i] = 0
18.
19. # record the length of heads run
20. temp = 0
21. for i in range(0, N_flips-1):
22.     if history[i+1] == 1:
23.         temp += 1
24.     else:
25.         length_of_heads_run.append(temp)
26.         temp = 0
27.
28.
29. # plot
30. plt.figure(1)
31. bins = np.arange(8) + 0.5
32. plt.hist(length_of_heads_run, bins, edgecolor="b", alpha=0.75)
```

```
plt.title("The result of tossing a fair coin 100 times")
plt.xlabel("tail = 0, head = 1")
plt.ylabel("length of heads run")
plt.show()
```

4. **Simulate tossing a fair coin and count the number of tosses until reaching a user-specified positive number of heads.**

The left column is the number of heads (self-specified), and the right column is the number of flips needed to reach the number of heads.

Number of heads	Number of flips needed
100	211
200	382
500	989
1000	1990
10000	19850

```
import numpy as np


N_heads_to_reach = 100
N_flips = 0
N_heads = 0

# generate coin-flip sequence
while N_heads < N_heads_to_reach:
    if np.random.random() >= 0.5:       # fair coin
        N_heads += 1
        N_flips += 1
    else:
        N_flips += 1

# print the tossing number needed
print(N_flips)
```