**PROJECT #1 - COIN FLIPS**

**Divyanshu Sahay  
(dsahay@usc.edu)**

Important Note:

Tool Used: **Python 3.7.4.**  
Editor Used: Visual Studio Code.

1. **Simulation of Bernoulli Trials:**

**Problem Statement:** A fair coin (a Bernoulli trial) toss has to be simulated 50 times. Following should be the findings:

* Total number of HEADS.
* Longest run of HEADS.
* Histogram of Bernoulli Outcomes.

Solution:

**Understanding Bernoulli Trials:** The Bernoulli trial refers to the random experiment in Probability and Statistics which always has only two possible outcomes, Success or Failure. In the problem the possibility of getting a HEAD or TAIL is treated as the Bernoulli’s Outcome.

**Methodology:** The Coin toss is simulated by generating a random real number between 0 and 1 using the ***random*** function in python. An outcome less that ‘0.5’ is treated as HEAD and ‘0.5’ is treated as TAILS. All the tosses are recorded in the ***toss*** variable. The total number of heads is shown in output as ***Total Heads*** in Fig. 1***.*** The total number of head count is given by the final value of this variable. The longest run of heads is stored in the ***longest\_run*** variable. The content of toss variable and the longest run can be seen in the output in Fig. 1. Finally, the Histogram plotted for the Bernoulli outcomes shown in Fig. 2. The value of the output will change with ***seed*** provided to the random function.

**Program Written**:

N\_flips = 50

p = 1.0/2.0 # proving the probability of 0.5 to each

np.random.seed(123)

head\_count = 0

longest\_run = 0

toss = np.random.rand(N\_flips)

for i in range(toss.shape[0]):

    if (toss[i] > p):

        toss[i] = 0

        if(head\_count > longest\_run):

            longest\_run = head\_count # calculating longest run

        head\_count = 0

    else:

        toss[i] = 1

        head\_count = head\_count + 1 # incrementing head count

print("longest run of heads: ", longest\_run)

print("Total Head:", sum(toss))

print(toss) # outcome of 50 tosses

x = range(1, N\_flips+1)

plt.hist(toss,2) #plotting histogram

plt.show()

**Results and Observations:**

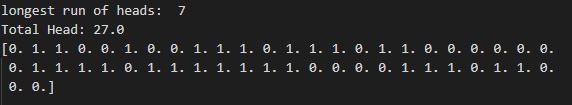
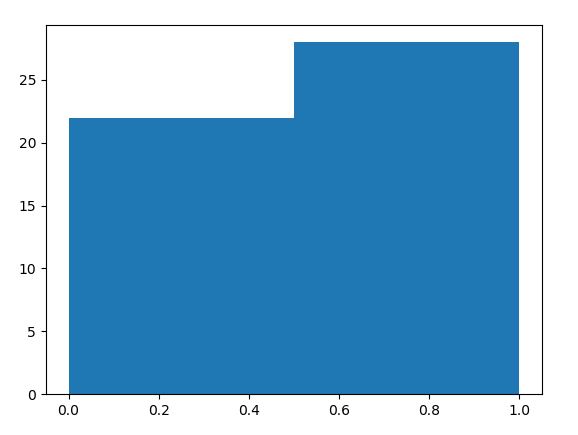


Fig. 1. Output Display



Number of Heads

Probability

Fig. 2. Histogram

We can see in the histogram that the number of heads is 27. The relative frequency thus obtained will be, 27/50 = 0.54. The relative frequency will approximately converge to 0.5 for both tails and heads.

1. **1(a). Repetition of Experiments:**

**Problem Statement**: The above experiment has to be repeated for 20, 100, 200 and 1000 times. For each caser the histogram has to be generated showing 50 flips.

**Methodology**: In this experiment the sample python code provided by the instructor has been used. The approach remains almost the same. The difference is that now the number of heads appearing for every experiment is stored in the ***flip\_results*** array. Its populated using the ***random.binomial*** function which returns a list of samples from a binomial distribution based on the inputted parameters. The **head** variable contains the ordered numbers (from 0 to ***N\_flips*** + 1) representing the possible quantities of heads that can appear in an experiment. Also, for individual experiments, the code makes use of the *Binomial Package* from *Stats Package (****stats.binom.pmf****)* to calculate the probabilities of certain number/quantity of HEADs appearing over ***N\_flips*** when the probability, ***p***, of a HEAD appearing is 0.5.

The model code provided considers this problem statement as tosses done by different classrooms of students (N\_students) in a school. Thus, ***N\_students***, in our case will represent the total number of times the experiment is performed. This value is changed from 20 to 1000.

**Program Written**:

N\_students = 200 # This can be changed to change the number of times the experiment is performed

N\_flips = 50

p = 1.0/2.0

np.random.seed(10)

heads = np.arange(0,N\_flips+1)

binomial\_pmf = stats.binom.pmf(heads, N\_flips, p)

flip\_results = np.array([

# <<---- add numbers here to "force" a particular class / this is populated with

# the HEADS and TAILS depending upon the number of times the experiment is performed

])

# generate samples if no class data

if (flip\_results.shape[0] == 0):

flip\_results=np.random.binomial(N\_flips, p, N\_students)

print("flip\_results")

print(flip\_results)

flip\_counts= np.zeros(N\_flips+1) #[0 for i in range(N\_flips+1)]

for k in range(N\_flips+1):

flip\_counts[k]=(sum(flip\_results==k))

print("flip\_counts")

print(flip\_counts)

print(sum(flip\_counts))

print("number of students =",N\_students)

print("number of flips =",N\_flips)

print("probability of heads =",p)

print("\n\nk n(k) p\_hat(k) MOE(k)\n")

p\_hat = flip\_counts / float(N\_students)

MOE = np.zeros( (2,N\_flips+1) )

MOE[0] = 1.96 \* np.sqrt( p\_hat \* (1.0 - p\_hat) / N\_students )

for k in range(len(p\_hat)):

if flip\_counts[k] == 0 :

MOE[1][k] = 3.0 / N\_students

else:

MOE[1][k] = MOE[0][k]

print(k,flip\_counts[k],"{:0.4g}".format(p\_hat[k]),"{:0.4g}".format(MOE[0][k]),sep=" ")

print("\n\n")

for k in range(len(p\_hat)):

print(p\_hat[k])

# fig = plt.figure()

# plt.stem(heads, binomial\_pmf, 'r', markerfmt='ro', label='model' )

# plt.errorbar(heads, p\_hat, yerr=MOE, fmt='o', label='measured')

# plt.legend(loc=2)

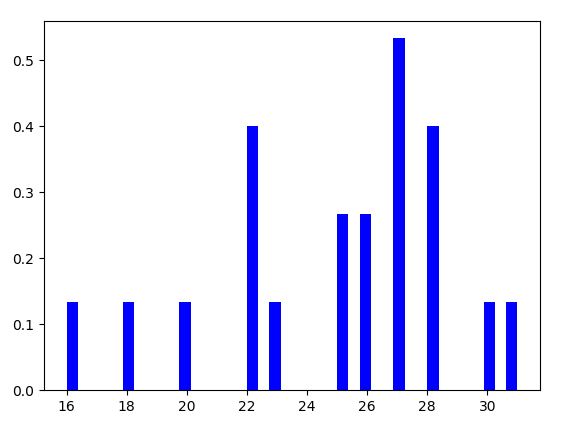
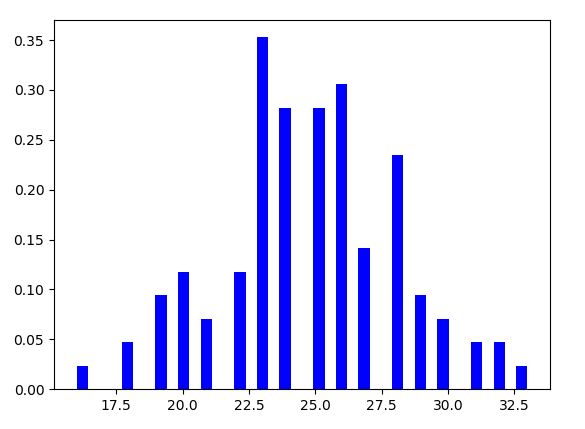
# plt.show()

fig = plt.figure()

plt.hist(flip\_results, 40, density=True, facecolor='b', alpha=1)

plt.show()

**Results and Observations:**

It can be seen in the histograms (Fig. 3, capturing all 4 histograms) that as the number of experiments increase the number of heads appearing in the experiments starts converging to 25 (i.e. it starts to attain a symmetry about the mean. This is in congruence with our intuition of getting 25 heads for each 50 flips.

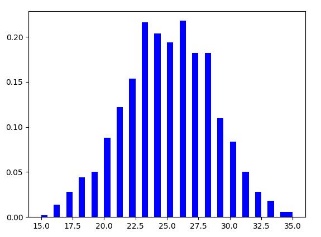
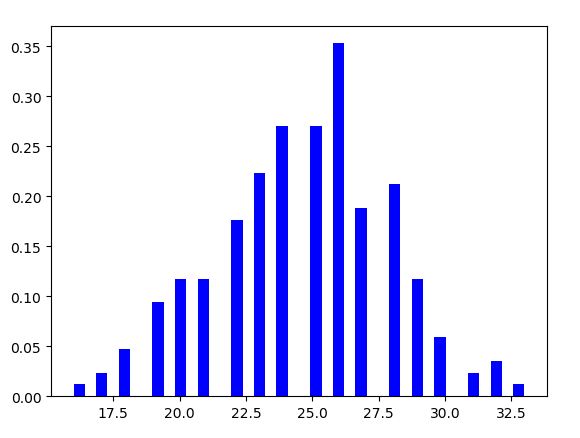
Fraction of Appearance

Fraction of Appearance

Number of Heads

Number of Heads

Experiment Repeated 20 times. Experiment Repeated 100 times



Fraction of Appearance

Fraction of Appearance

Number of Heads

Number of Heads

Experiment Repeated 200 times Experiment Repeated 1000 times

Fig. 3. Histogram for all the four experiments

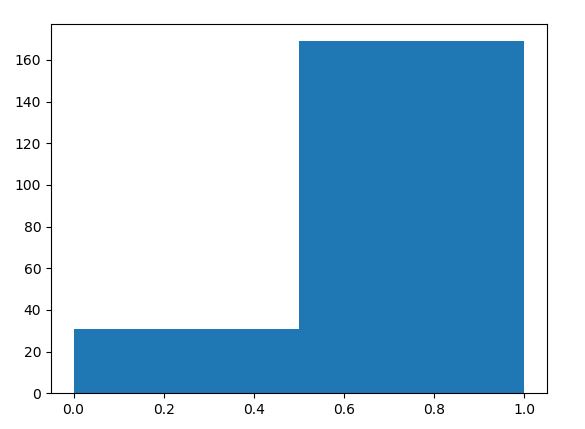
1. **Tossing a Biased Coin:**

**Problem Statement:** A coin needs to be flipped 200 times.This problem is similar to the 1st one. The major difference is that the coin being flipped is biased and the P(HEAD) = 0.8. Following the requirements:

* The total number of HEADs
* Longest run of Heads
* Histogram of Bernoulli’s outcome

**Methodology**: The methodology remains same. The same program is used. The only difference is that the probability, ***p***, is changed to 0.80.

**Results and Observations:**



Number of Heads

Probability

Fig. 4. Histogram (P(HEAD) = 0.80)

It is clear from the histogram in Fig. 4. that as the P(HEAD) is greater the number of times it appears is more. The longest run and the total number of heads can be seen in the output screen in Fig. 5. This is also in congruence with our expectation as 166/200 = 0.83.

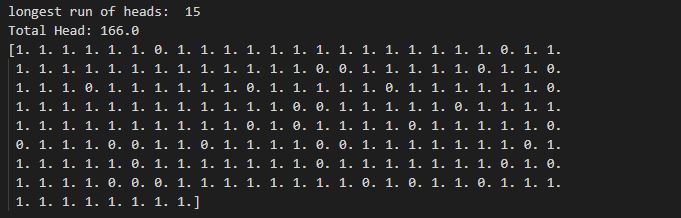


Fig. 5. Output Screen.

1. **Tossing a coin 100 times to study run lengths:**

**Problem Statement:** This problem requires us to toss the coin 100 times. Following are the requirements:

* Generate histogram showing the head length.

**Methodology**: The methodology is similar to Problem 1. However, here we need to record the run length. A new variable ***head\_count\_list*** is used to do the same.

**Program Written:**

N\_flips = 100

p = 1.0/2.0

head\_count = 0

longest\_run = 0

head\_count\_list = []

np.random.seed(10)

toss = np.random.rand(N\_flips)

for i in range(toss.shape[0]):

if (toss[i] > p):

toss[i] = 0

if(head\_count != 0):

head\_count\_list.append(head\_count) # storing the head count in the

list

head\_count = 0

else:

toss[i] = 1

head\_count = head\_count + 1

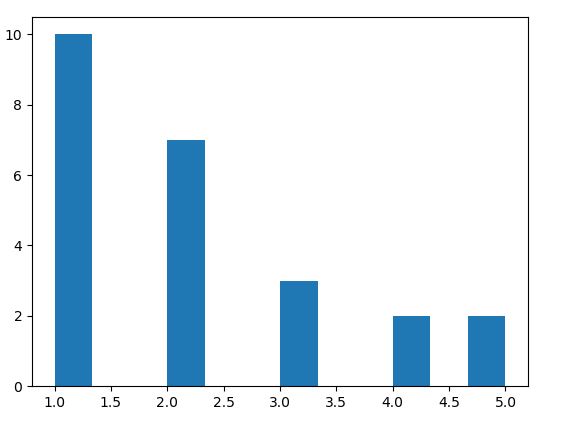
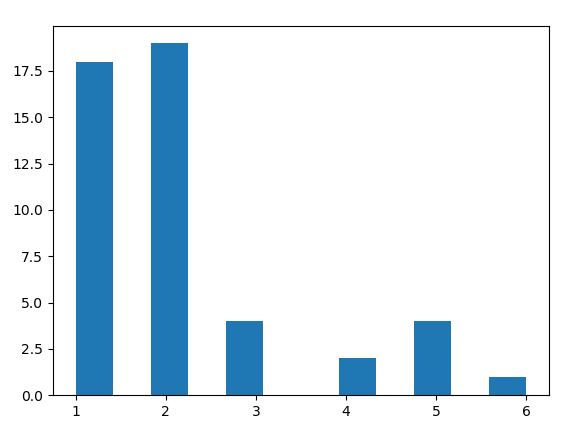
if(head\_count != 0):

head\_count\_list.append(head\_count) # storing the head count in the list

print(head\_count\_list)

plt.hist(head\_count\_list,12) # plotting the histogram

plt.show()

  
 Fig. 6. Histogram. Left (200 times). Right (100 times)

Frequency of Appearance

Frequency of Appearance

Number of Heads

Number of Heads

**Results and Observation:**

It can be seen in the histogram that run length 1 and 2 is most frequent. HTHT.. or HHTTHTTHHTT.. pattern appears the most this directly suggest that the P(HEAD) = 0.5. Again, histogram is in accordance with what we may expect.

1. **Tossing User Defined Heads:**

**Problem Statement:** This problem requires us to toss a fair unless a user-specified number of heads is reached.

**Methodology:** In this problem the ***random.randint*** function is used togenerate 0 and 1 randomly. The user is asked to input the desired quantity of HEADs with the help of ***input*** function. Finally, the number of tosses required to reach that number of head is stored in trial.

**Program Written:**

head\_count = 0

trial = 0

head = int(input ("please enter desired head count ")) # specified headcount

import random

while head\_count < head : #Checking if head\_count reaches its limit

    if (random.randint(0,1) == 1):

        head\_count = head\_count + 1

    trial = trial + 1

head = int(input ("please enter desired head count ")) # user specified

print(trial)    # total number of tosses

print(head\_count) # number of heads appeared

**Results and Observation:**

|  |  |
| --- | --- |
| **HEAD COUNT** | **TRIALS** |
| 10 | 17 |
| 100 | 188 |
| 500 | 1020 |
| 1000 | 1998 |

The results obtained show that the number trails required is almost twice as that of HEADs obtained. So, this is in accordance with the fact that the P(HEAD) is 0.5.