EE511: ZHANG Fan USC ID: 1417685115

Q1:Simulate tossing a fair coin(a Bernoili trial) 50 times. Count the number of heads. Record the longest run of heads. Generate a histogram for the Bernouli outcomes.

**Methods**:

* Record the run length:
  + the goal of the title of statement is to find the consecutive “1”. For example: the runs of lengths of 100111001111 is (1,3,4).
  + In stead of doing loop, my thought to find out and mark the change of digits(1->0 or o->1) of the array. For example, the mark of change of 100111001111 is 1,4,9(0->1) and 2,7,13(1->0). And calculate the length of the “1”.
  + Xor is used to realize the marking. For example, left shift one bit of the original data: 100111001111->[100111001111] and do XOR([100111001111,0],[0, 100111001111]) we will get the mark of (1->0): (1000010000010), then use find command in MATLAB to find the index of 1(that means the location where 1->0). The same as to obtain index of(0->1).
  + After that, the index array of (1->0) minus the index array of (0->1) and then we will get the runs length.
* Use max command to get the longest run of heads
* Use histogram to get the figure.

**Discussions of the Algorithm:**

Compared with the for loop, this Algorithm will save a lot of time since it just do shifting, which especially fit for the architecture of our computer(CPU).

**Codes**:

%Q1

clear

N=100;

TRIES=1;

THREDSHOD=0.5;

for i=1:TRIES

isHEAD=rand(1,N)>THREDSHOD;

runs=reshape(find(xor([0,isHEAD],[isHEAD,0])==1),2,[]);

run=runs(2,:)-runs(1,:)

longest=max(run);

T(i)=sum(runs(2,:)-runs(1,:));

end

histogram(isHEAD);

%title('The Histogram for the Bernoulli Outcomes')

%xlabel('Outcomes:HEADS(1)/TAILS(0)');

%ylabel('Frequency of the Outcome');

%disp('The longest run is:')

%disp(longest)

%histogram(T)

%title('The Histogram of Towssing(TRIES:1)')

%xlabel('The Number of Head');

%ylabel('Frequency of the Same Head Number');

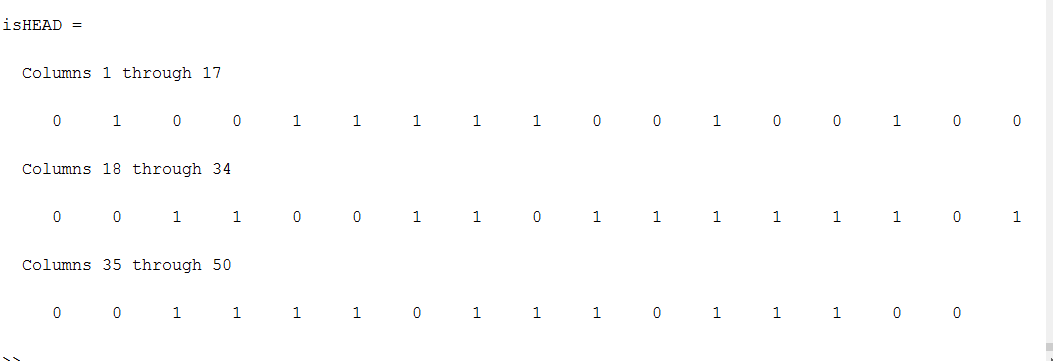
histogram(run)

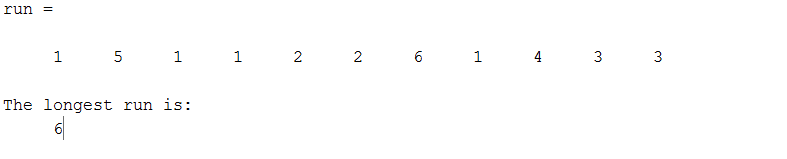
title('The Histogram of Runs Lengths of TOSSING 100 Times')

xlabel('The Frequency of Possible Runs Lengths');

ylabel('The Possible Runs Lengths');

**Results:**





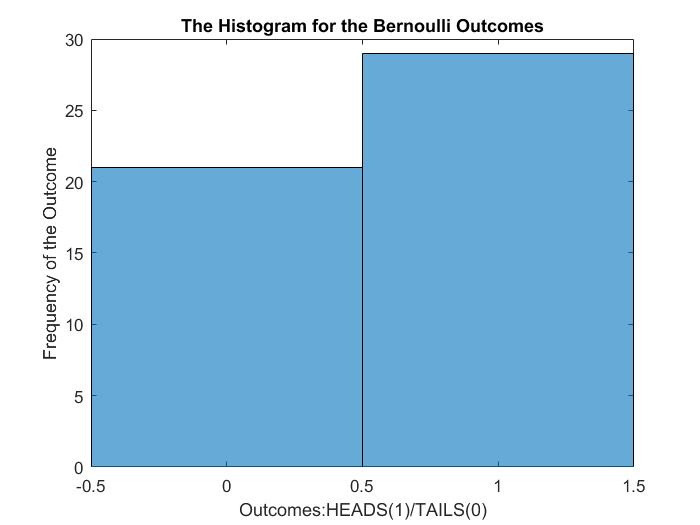


Figure1：The Histogram for the Bernouli Outcomes

Q1a: Reapeart the above experiment 20,100,200 amd 1000 times. Generate a histogram for each showing the number of heads in 50 flips. Comment on the limit of the histogram.

Methodes and codes:

The methods and codes are similar with the Q1’s. Just revise the parameter: TRIES and the xlable and ylable to shows the different figure.

**Results**:

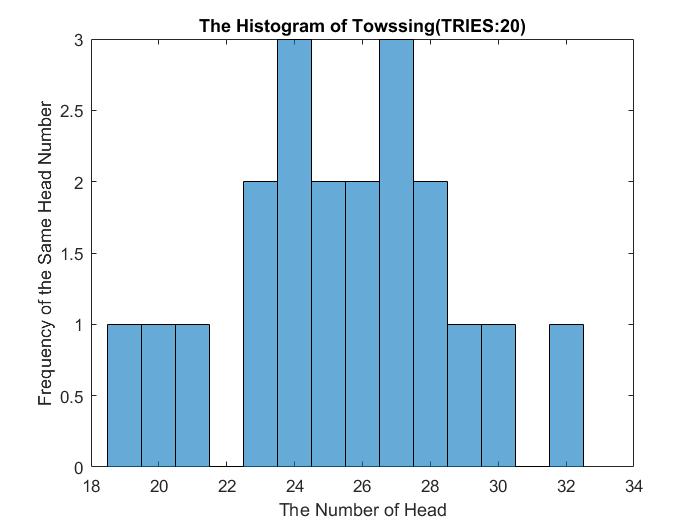


Figure 2: The historam of Towssing(TRIES:20)

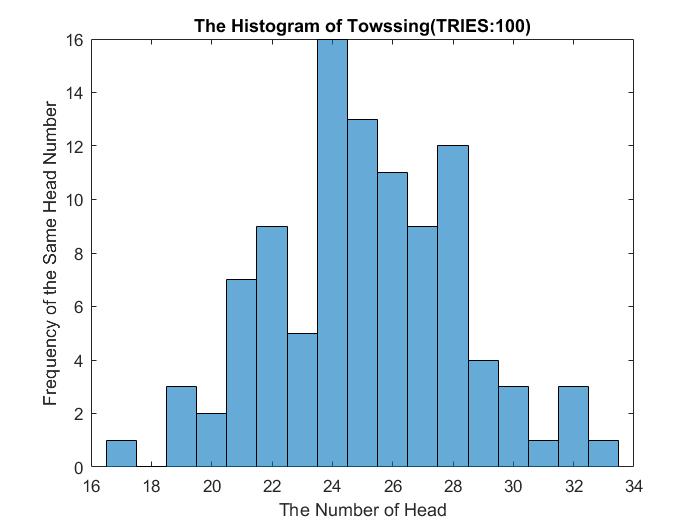


Figure 4: The historam of Towssing(TRIES:200)

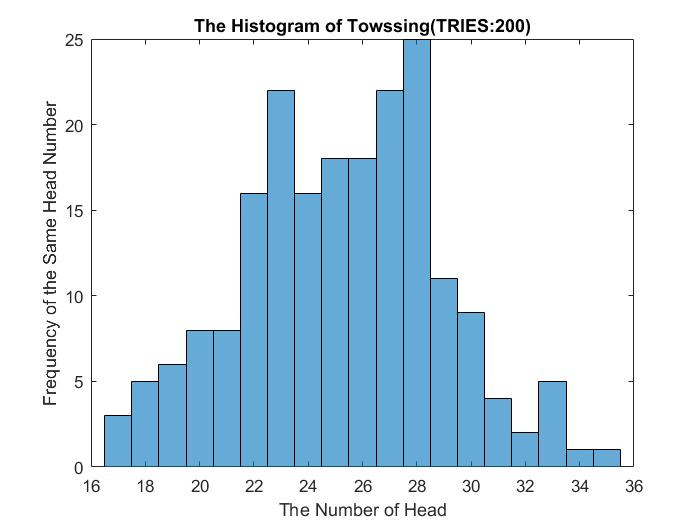


Figure 4: The historam of Towssing(TRIES:200)

**Discussions**:

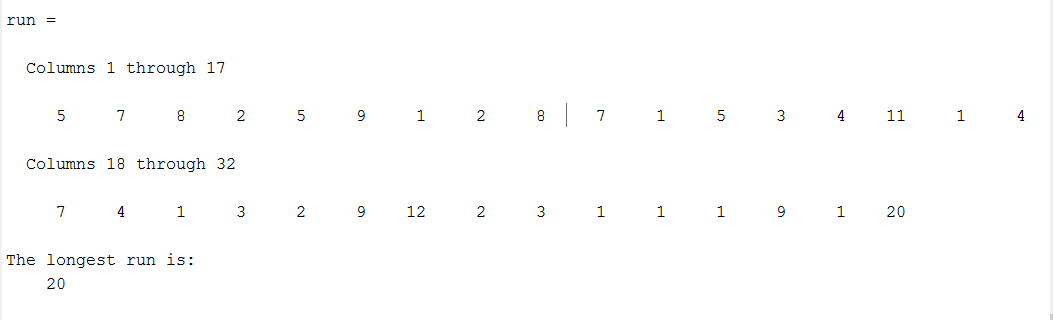
* The disadvantages of histogram is that we cannot get a smooth curve from the figure, which will result in some inaccuracy especially when the **quantity of the data is small.**

Q2： Simulate tossing a biased coin 200 times where P[HEAD]=0.8. Count the number of heads. Record the longest run of heads. Generate a histogram for the Bernoulli outcomes.

**Methods**:

The method is the same as the Q1’s. Set TRRIES=1, THRESHOD=0.2 and N=200.

**Results**:



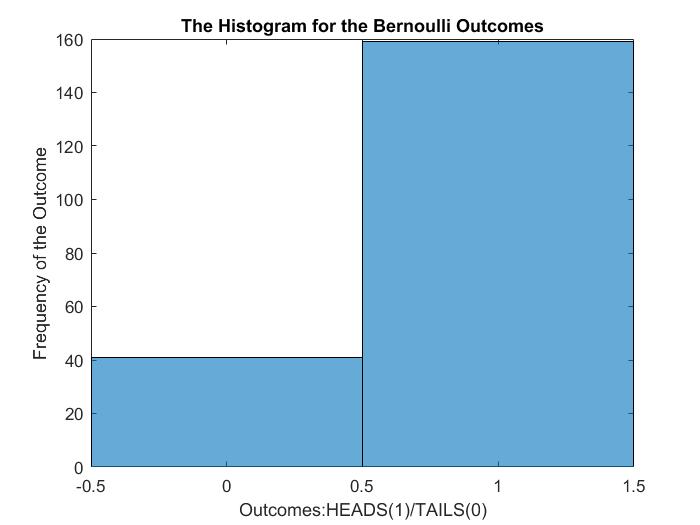


Figure 5: The histogram for the Bernoulli outcomes for P[HEAD]=0.8

Q3: Simulate tossing a fair coin 100 times. Generate a histogram showing the heads run lengths.

Method: The methods used here is the same as the Q1. Based on Q3, set THRESHOD=0.5 and N=100.

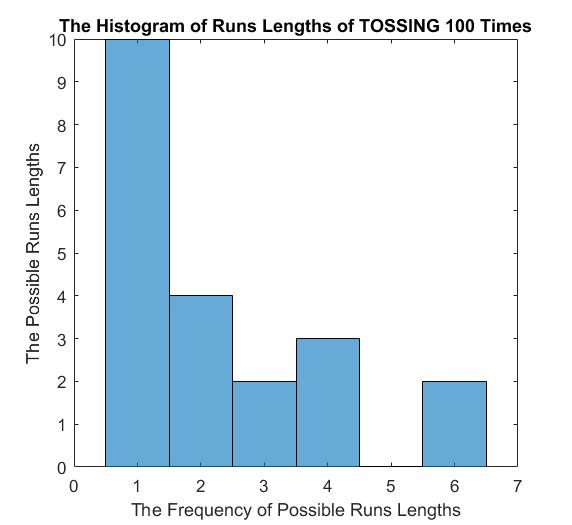


Figure 6: The Histogram of Runs lengths of tossing 100 times.

**Discussion**:

we can find in the figure that the frequency of the run of 1 is the largest since the propability to gain head is ½. So it is more likely to get one TAIL and one HEAD.

Q4：Simulate tossing a fair coin and count the number of tosses until reaching a user-specified positive number heads.

**Methods**: Our goal is to obtain the times you need to toss when the number of heads reach to the set number. So we just do a loop and until the HEAD number reach the to the one we set: x. The code is as the following shows.

%4

clear

x=input('The number of heads:')

TRIES=1;

THREDSHOD=0.5;

Count=0;

HEAD=0;

while(HEAD<x)

HEAD=HEAD+(rand(1)<THREDSHOD);

Count=Count+1;

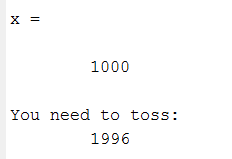
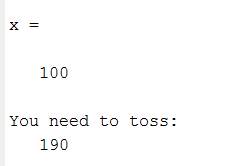
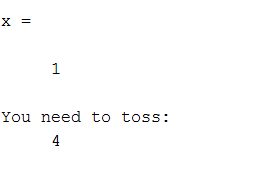
end

disp('You need to toss:')

disp(Count)

disp('times')

**Results**:



**Discussion**:

After several experiments we find that：

* If you want to get 1 head, the propability will always greater than 2.
* The more times you toss, the more closer that the propability reach to ½.

In order to get the more stable results, we improve the trials time and take the average times.

%4

clear

x=input('The number of heads:')

TRIES=1;

THREDSHOD=0.5;

Count=0;

for i=1:100

HEAD=0;

while(HEAD<x)

HEAD=HEAD+(rand(1)<THREDSHOD);

Count=Count+1;

end

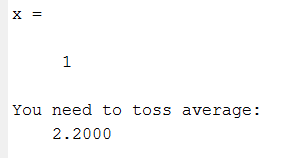
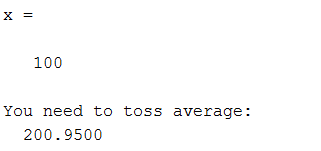
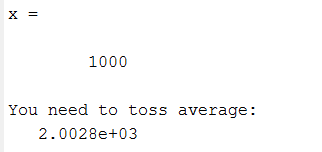
end

disp('You need to toss average:')

disp(Count/100)

disp('times')

**Results(Average times):**



**Discussion:**

This will be more perfect to show the discussions above.