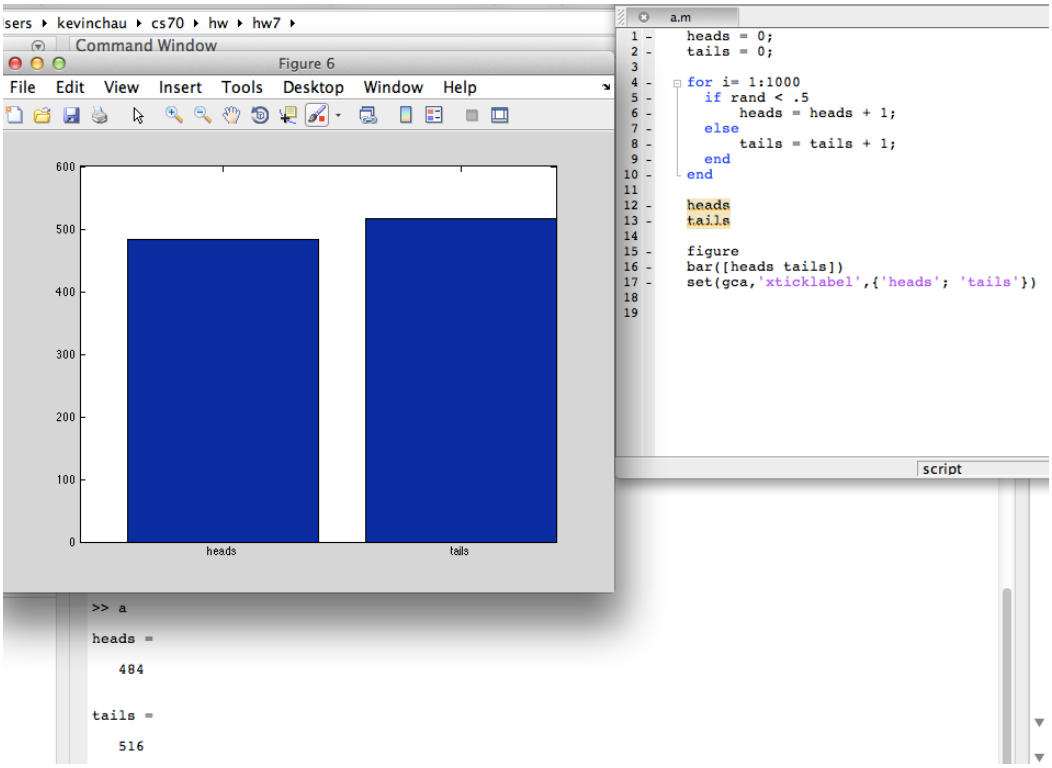
Kevin Chau
23816929
CS70 Homework 7
Partners: Samuel Drake, Ahdil Hameed, Unis Barakat

Problem 1

Code Instructions:

The code for each part is in a separate .m matlab file named after the part letter name. For example, part a is in a matlab file called a.m (all the way up to p.m). Just open each script in matlab and run to see any plots or data generated. The code from each file as well as any plots/figures that were generated when I ran it are included in the writeup below. In particular, I was using Matlab r2013a.

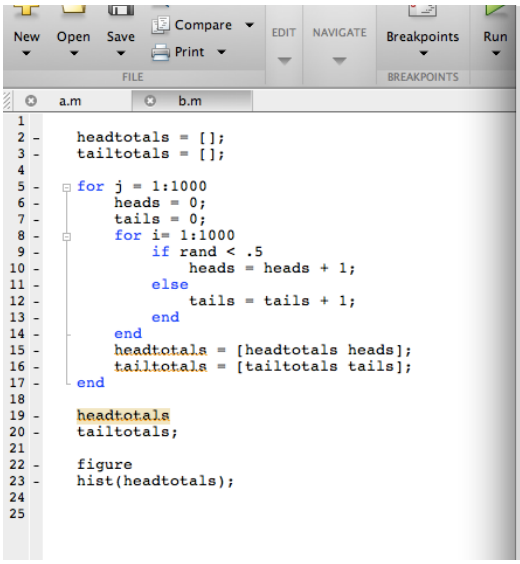
a) Use a random number generator to simulate a sequence of fair coin tosses. Do 1000

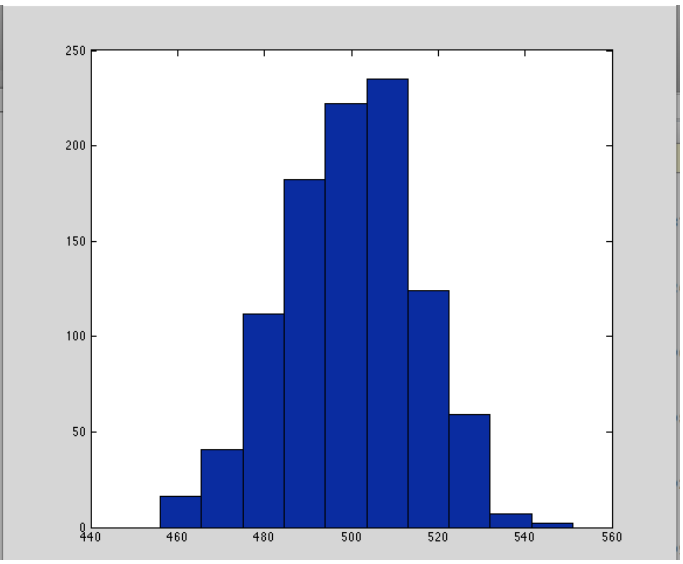


coin tosses. Plot a histogram of how many heads you got vs how many tails.

The rand function in Matlab returns a number in the interval (0,1) with an even distribution. I used the fact that theres a 50% chance that rand generates a number less than 1/2 and a 50% chance it will generate one greater than 1/2 in order to simulate a coin toss, where heads and tails have 50/50 chance. The histogram is just plotted with a bar graph, labeled heads and tails. In order to do 1000 tosses, I just used a for loop over the index “i”.

b) Do the previous part 1000 times. Plot a histogram of how many times you got N heads.



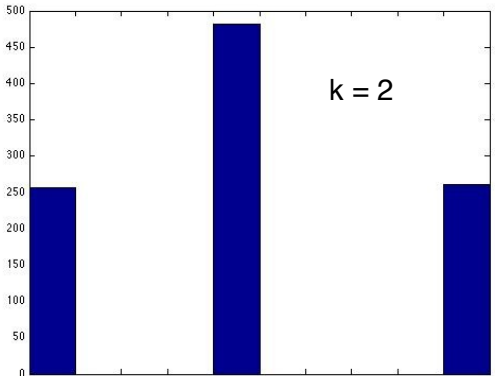


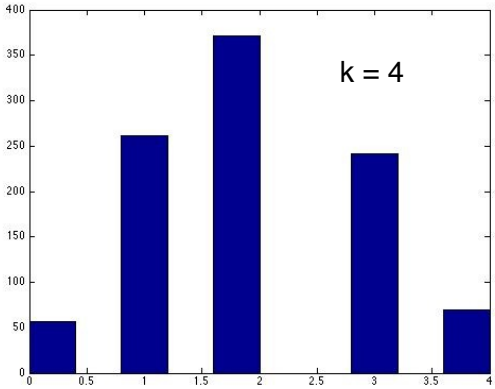
The histogram shows how many times I got N heads when I tossed 1000 coins 1000 times.The distribution is divided into intervals of 10 heads. The median number of heads was a little over 500 out of 1000, so even though a coin has a 50/50 chance of landing on either side, the result of tossing 1000 coins does not exactly match up with the probability (provided my simulation is a good model for a coin toss). 1000 is still too small of a number to get an even distribution.

c) Consider the 1000 in part (a). Now, let that be a parameter k that tells how many coins you toss in one experiment. Do part (b) again for the following sequence of ks: 2, 4, 10, 50, 100, 500, 10000, 100000.

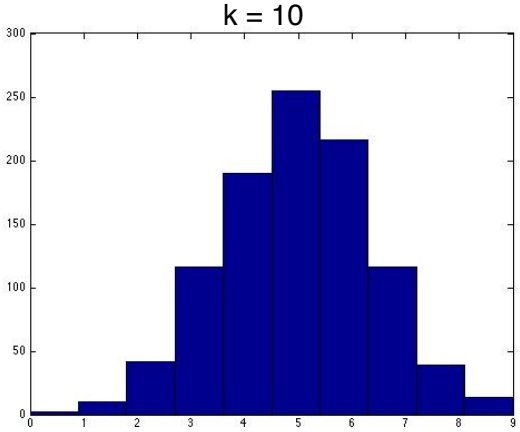
Code:

```
for k = [2 4 10 50 100 500 10000 100000];
  figure
  headtotals = [];
  tailtotals = [];
  for j = 1:1000
     heads = 0;
     tails = 0;
     for i= 1:k;
        if rand < .5
           heads = heads + 1;
        else
           tails = tails + 1;
        end
     end
     headtotals = [headtotals heads];
     tailtotals = [tailtotals tails];
  end
  headtotals;
  tailtotals;
  hist(headtotals);
  hold; end;
```

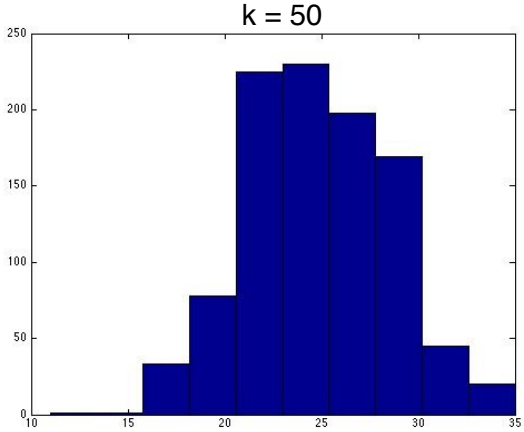




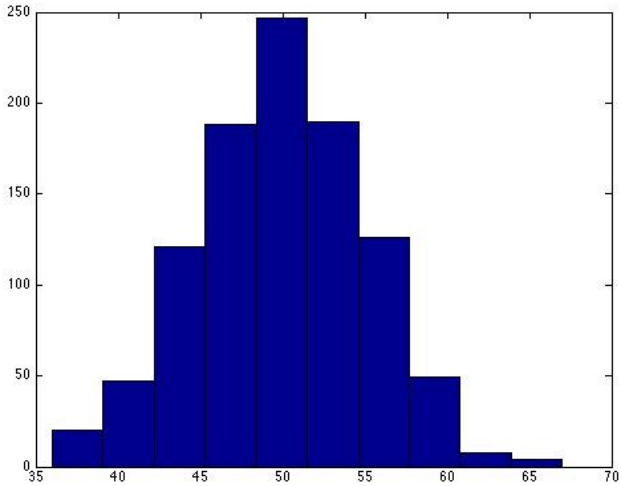
k = 10



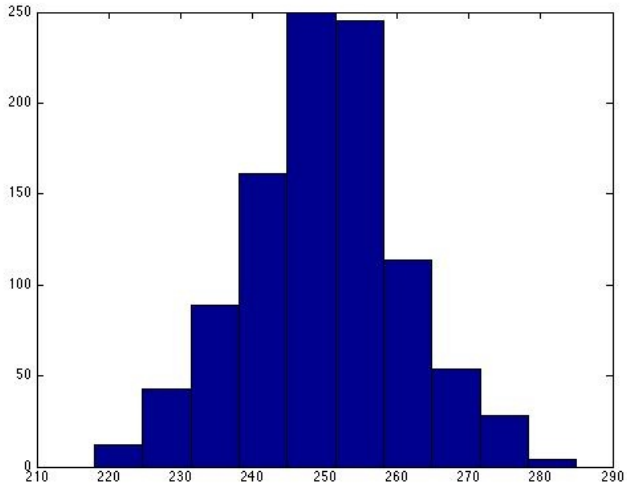
k = 50



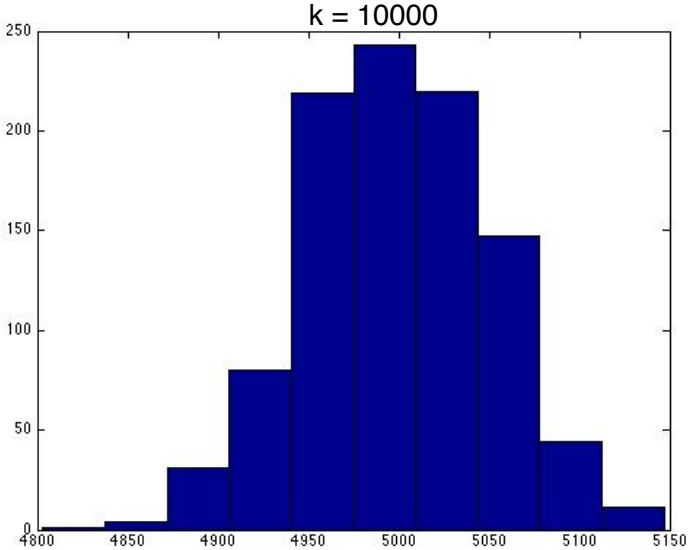
k = 100



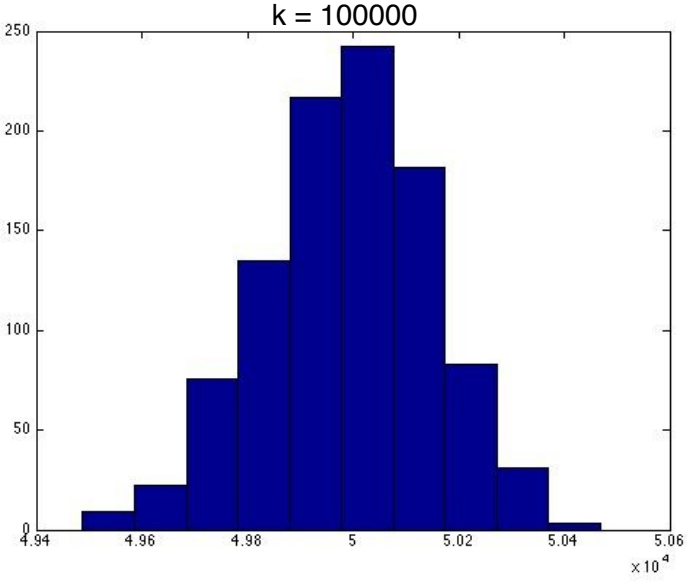
k = 500



k = 10000



k = 100000

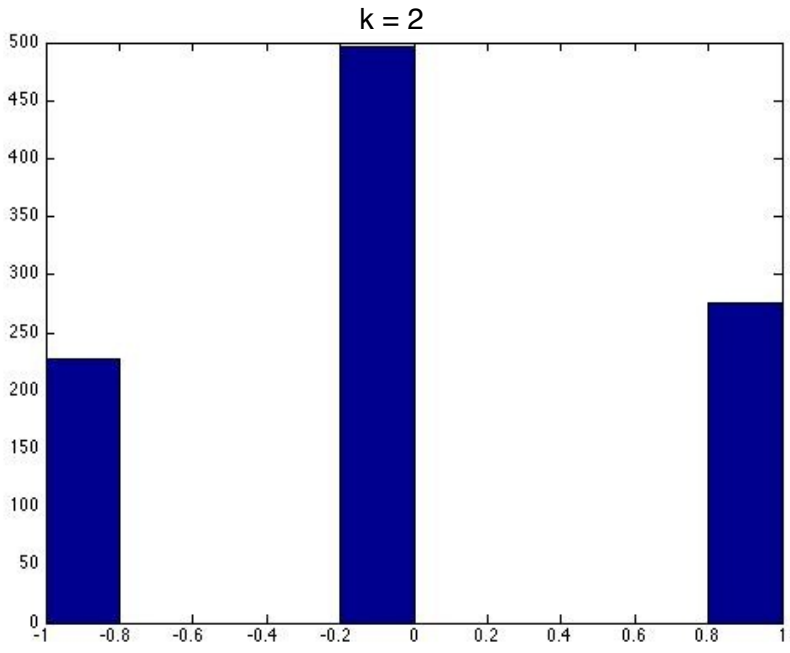


From the graphs in this section, we see that on an absolute x-scale, the higher the k value is, the wider the spread in N heads per run is. For example, k = 100000 has a spread of about 400 heads between the median and the rightmost side of the distribution, while k = 10000 only has a spread of about 150.

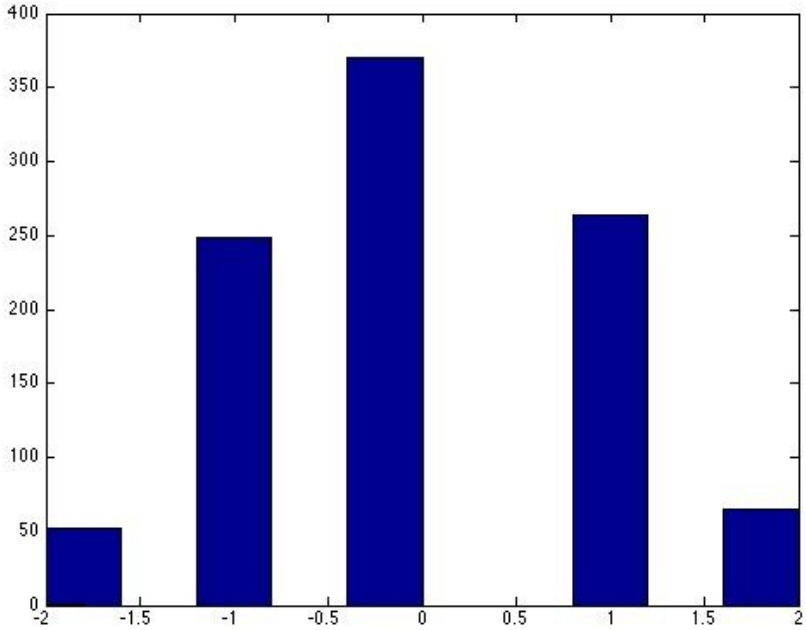
d) Notice that the horizontal axis has different scales as k varies. Suppose you wanted to "center" these histogram plots. To where would you move the origin as k varies? What is f(k) such that each N heads gets a shift to N-f(k). Place the histograms one above the other.

CODE:

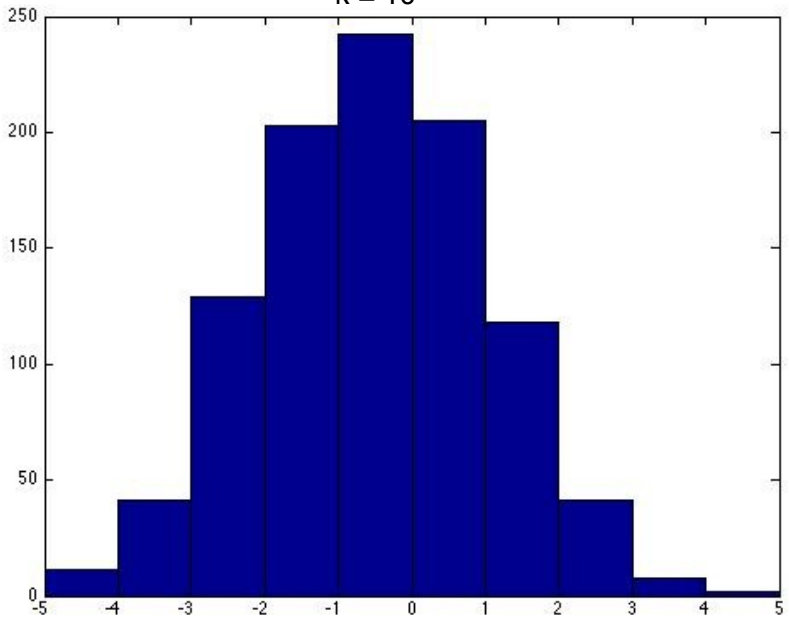
```
for k = [2 4 10 50 100 500 10000 100000];
  figure
  headtotals = [];
  tailtotals = [];
  for j = 1:1000
     heads = 0;
     tails = 0;
     for i= 1:k;
        if rand < .5
           heads = heads + 1;
        else
           tails = tails + 1;
        end
     end
     headtotals = [headtotals heads - k/2];
     tailtotals = [tailtotals tails - k/2];
  end
  headtotals;
  tailtotals;
  hist(headtotals);
  hold;
```



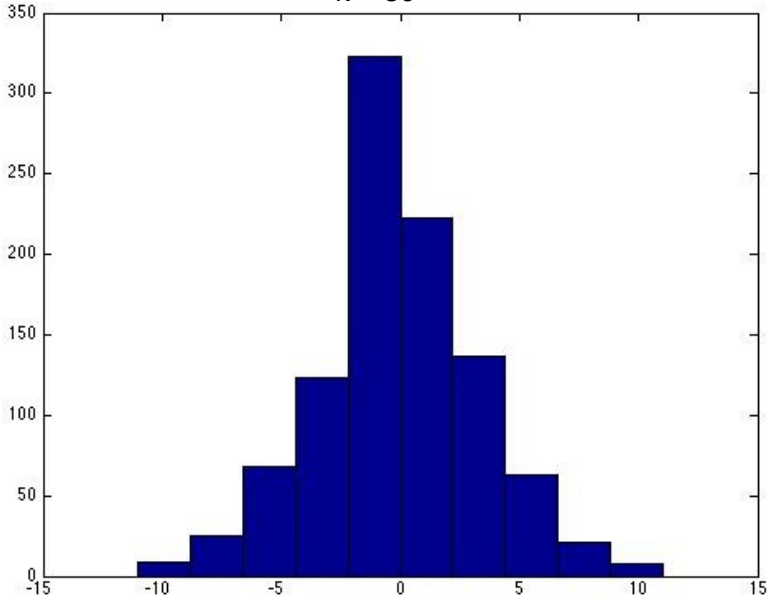
k = 4


400
350
300
250
200
150
100
50
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

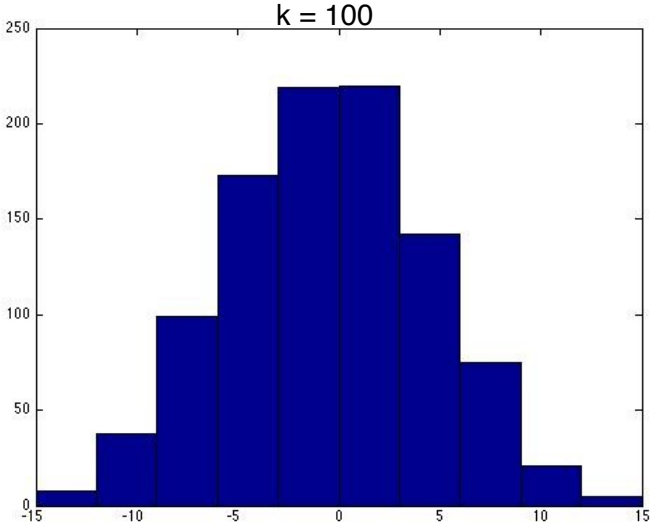
k = 10


250
200
150
100
50
0
-5
-4
-3
-2
-1
0
1
2
3
4
5

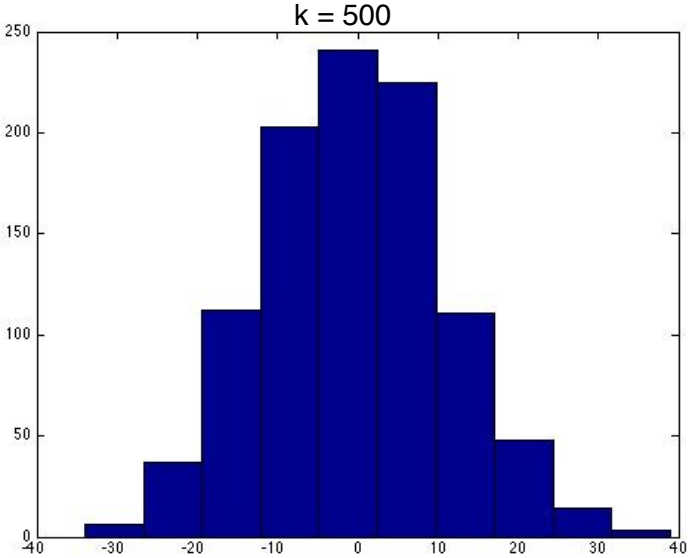
k = 50


350
300
250
200
150
100
50
0
-15
-10
-5
0
5
10
15

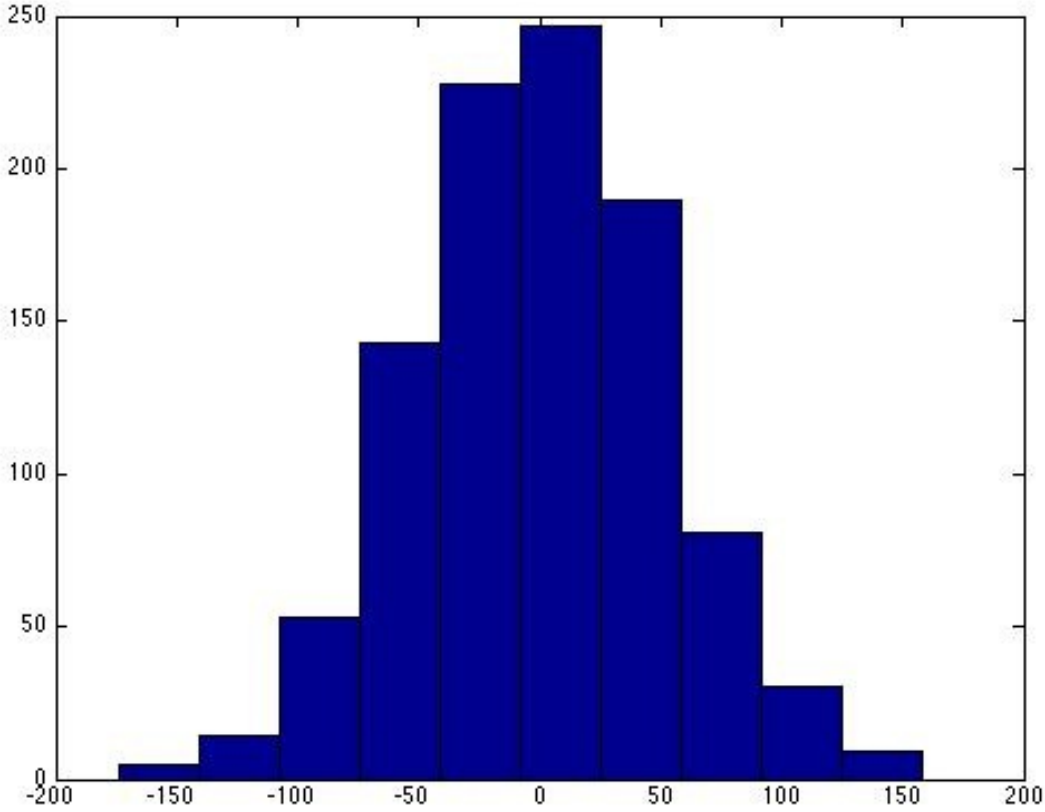
k = 100

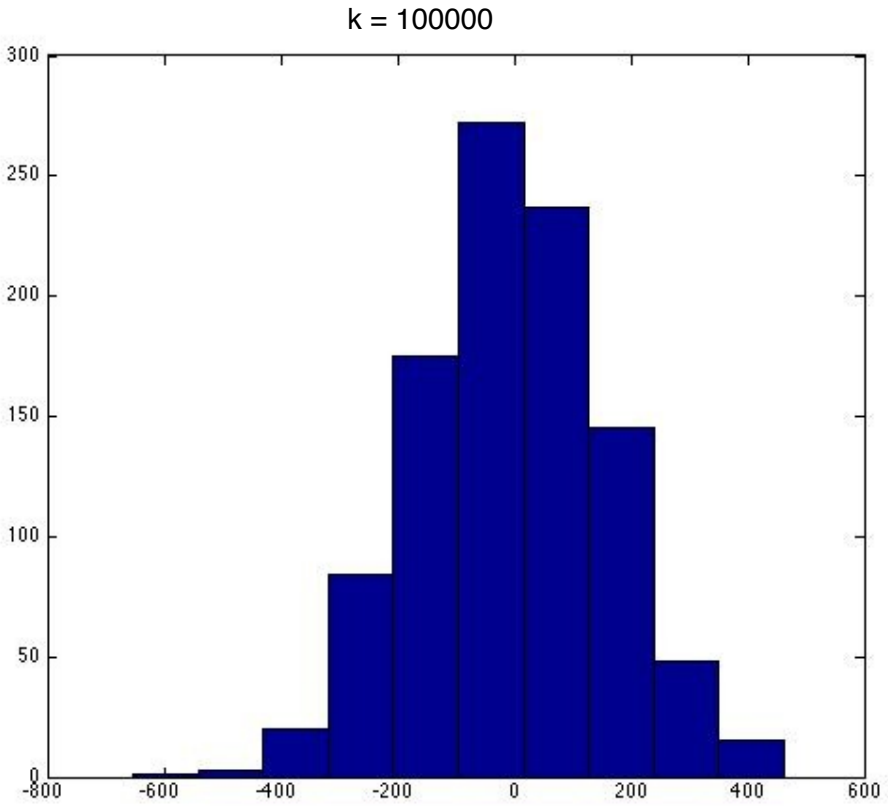


k = 500



k = 10000





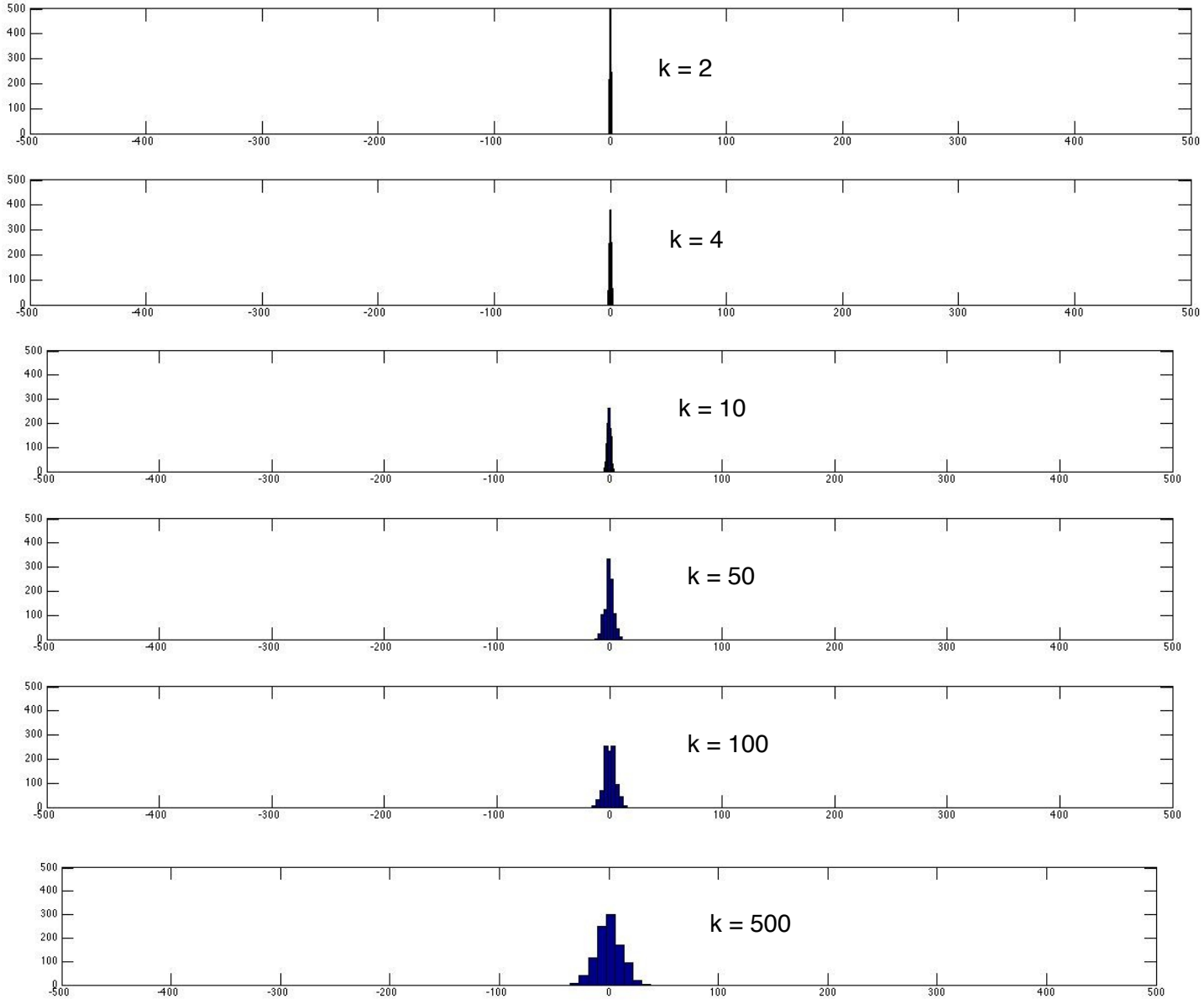
end
I shifted each histogram so that they were centered around the origin. So 0 corresponds to getting roughly the same amount of heads and tails, while being on the right means getting more heads, and being on the left means getting more tails. The shifting function f(k) that i chose to do this was f(k) = k/2, since we expect our origin to correspond to where we get heads half of k times.

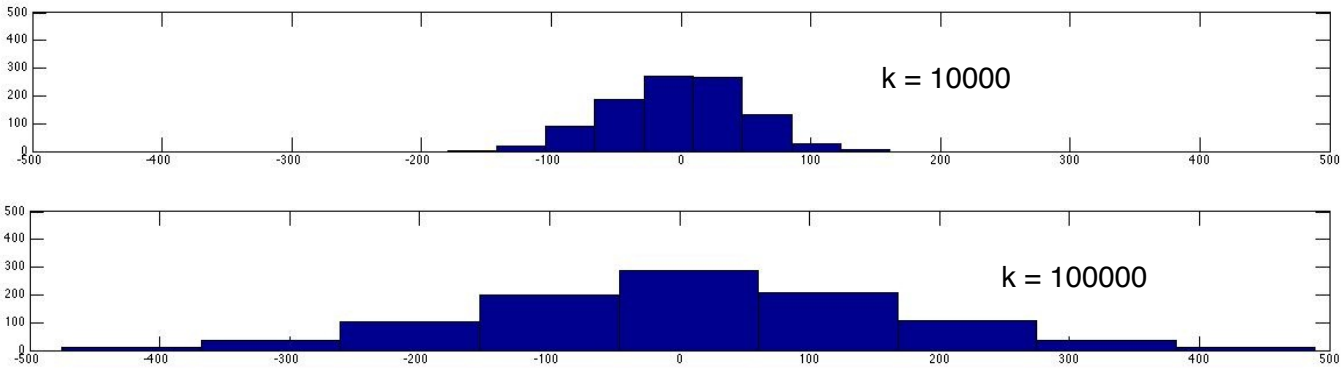
e) Repeat the plots of the previous part except this time, choose a common set of units — so one inch on the paper
should correspond to the same number of "ticks" of N. (e.g. one inch could correspond to 100. So the total
potential range for k = 100 would just be 1 inch while the potential range for k = 1000 would be 10 inches.)
Code:

```
for k = [2 4 10 50 100 500 10000 100000];
  figure
  headtotals = [];
  tailtotals = [];
  for j = 1:1000
      heads = 0;
      tails = 0;
      for i= 1:k;
          if rand < .5
              heads = heads + 1;
          else
              tails = tails + 1;
          end
```

```
    end
    headtotals = [headtotals heads-k/2];
    tailtotals = [tailtotals tails-k/2];
  end
  headtotals;
  tailtotals;
  hist(headtotals,9);
  axis([-500 500 0 500]);
  hold;
end
```

To do this part, I just used the axis function in matlab to make sure every histogram has the same relative units. So each graph is in the range of -500 to 500 for N—the smaller values of k will of course have much narrower distributions compared to the large values of k.

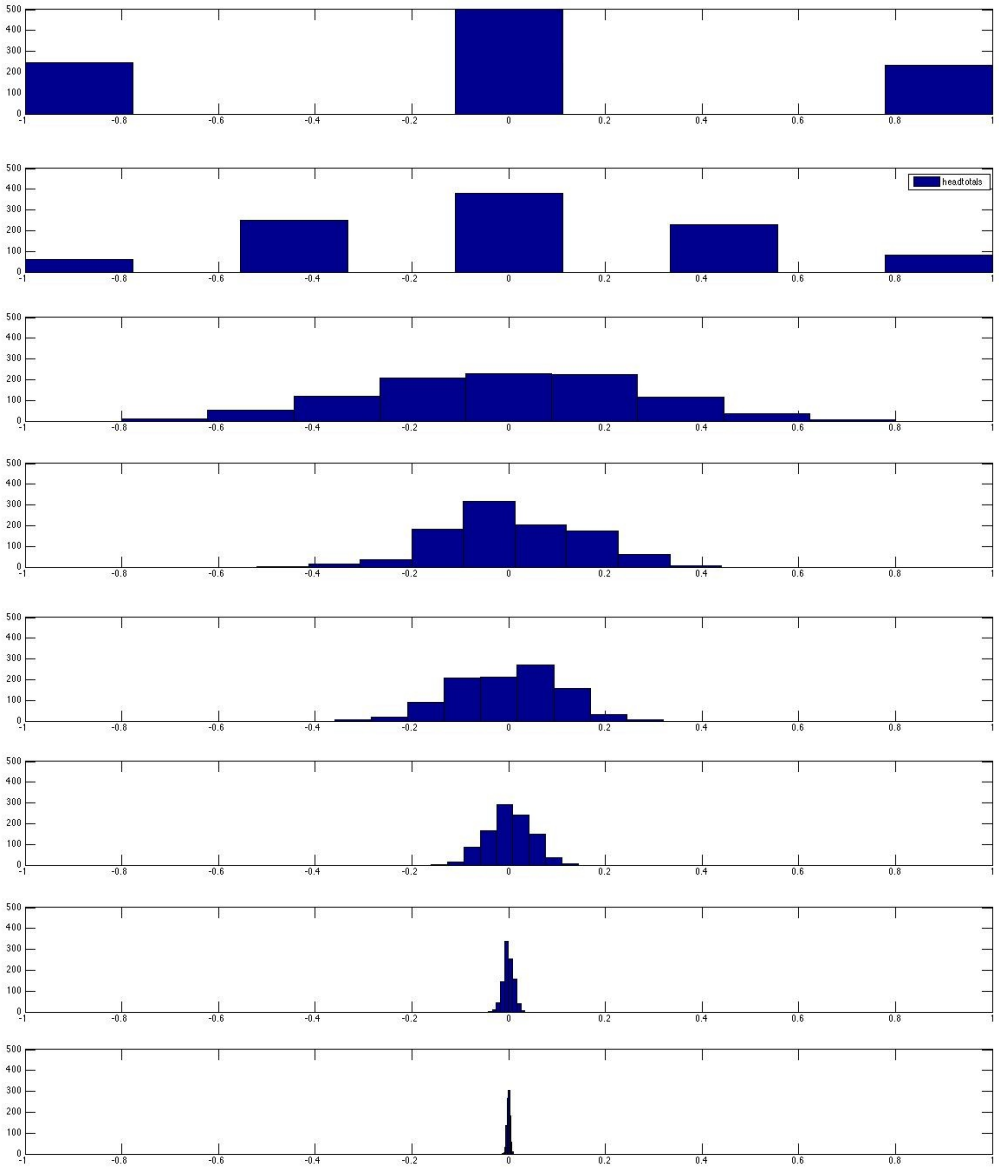




f) Repeat the plots of the previous part except this time, choose a normalized set of units. So the left-most point
should correspond to the case of tossing all tails. And the right-most point should correspond to the case of
tossing all heads. (this might never happen in your 1000 runs)
CODE:

```
for k = [2 4 10 50 100 500 10000 100000];
  figure
  headtotals = [];
  tailtotals = [];
  for j = 1:1000
      heads = 0;
      tails = 0;
      for i= 1:k;
          if rand < .5
              heads = heads + 1;
          else
              tails = tails + 1;
          end
      end
      headtotals = [headtotals (heads-k/2)/(k/2)];
      tailtotals = [tailtotals (tails-k/2)/(k/2)];
  end
  headtotals;
  tailtotals;
  hist(headtotals,9);
  axis([-1 1 0 500]);
  hold;
end
```

For this part, I normalized the x axis of the histogram by dividing the value of N after shifting the origin by k/2. This made it so that the very left side corresponding to -1 means all tails were flipped, and +1 means all heads were flipped.



The k values here are in the same order as the previous part.

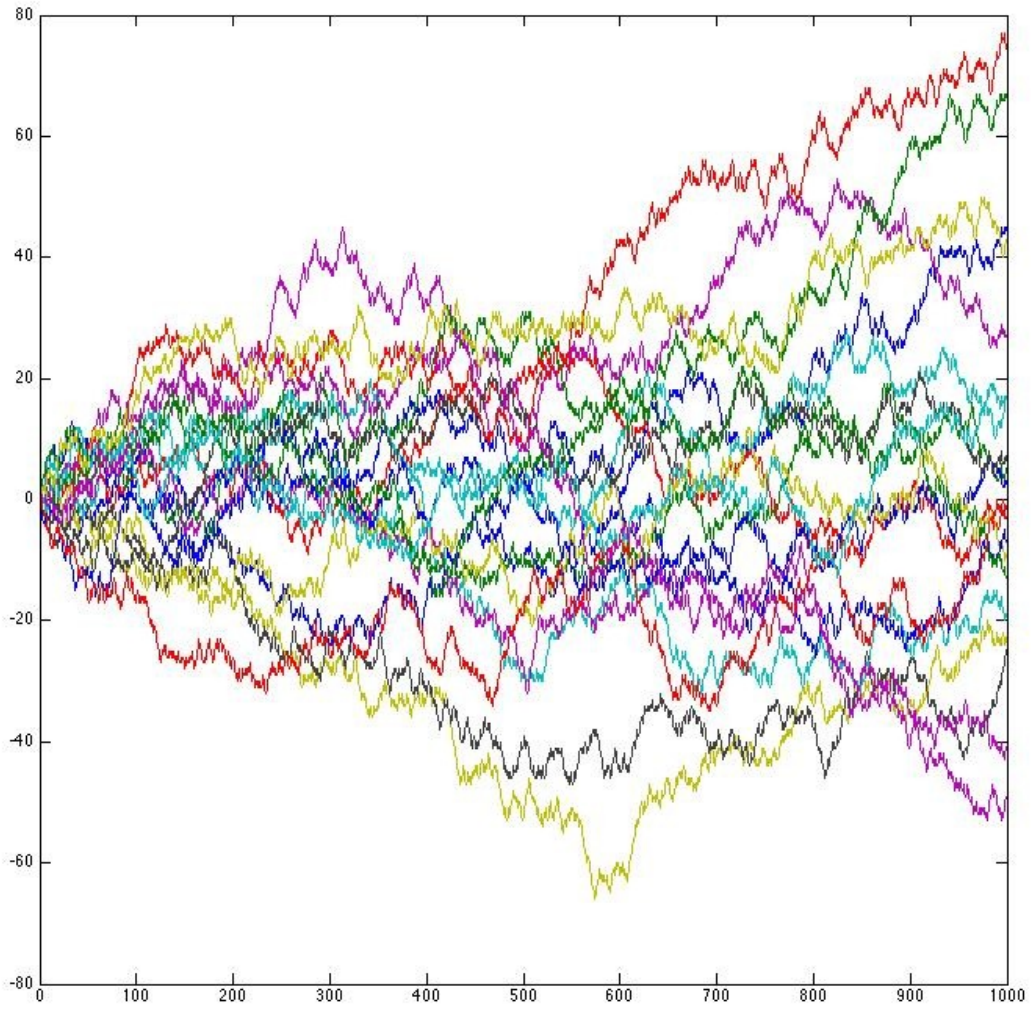
g) Comment on what you observed in the three sets of plots you have seen above.

Notice that on the relative scale, larger k's seem to have less deviation from the origin, as indicated by the narrower peaks, while smaller k's tend to have -1 and +1 which much greater prevalence. This is opposed to the absolute scale, which shows that smaller k's have smaller deviations from the origin. However, the relative scale is actually much more informative because it shows that when you use large enough k's, we won't ever get any weird outcomes such as flipping all heads or flipping all tails. In other words, larger k means that the distribution starts matching up with the probability, despite the absolute value of the deviations growing bigger.

h) Now, we will change gears a little bit. Consider the following visualization of a sequence of coin flips. We start
at zero. For every head we get, we add one. For every tail we get, we subtract one. So a sequence of 1000 coin
tosses would be a path that starts at (0,0), and then goes to either (1,1) or (1,-1), and continues wandering till

(1000,y) somewhere. Plot 20 such paths on the same plot based on randomly flipped coins. Each sample path
should have 1000 coin tosses.
CODE:

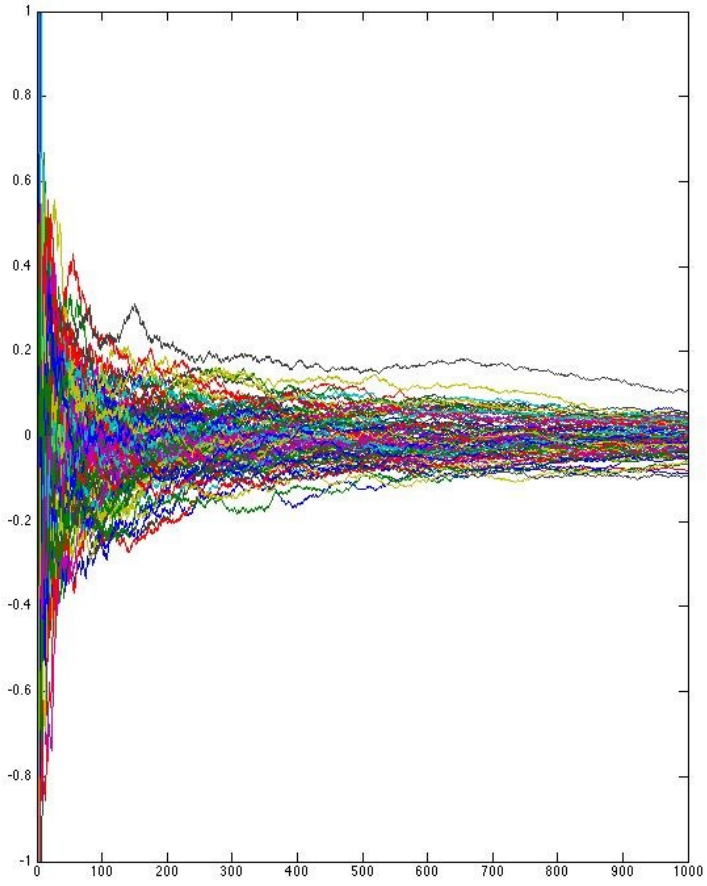
```
figure;
for j = 1:20;
cointotal = 0;
coinarray = [];
for i = 1:1000
    if rand < .5        %%heads
       cointotal = cointotal + 1;
       coinarray = [coinarray cointotal];
    else
       cointotal = cointotal - 1;
       coinarray = [coinarray cointotal];
    end
end
plot(1:1000,coinarray);
hold all;
end;
```



The plot shows 20 such runs of 1000 coin tosses. Notice that as we do more and more tosses, the likelihood that we end up far from the origin increases. In other words, if we do fewer tosses, we are more likely to have an even amount of heads and tails. However, at 1000, the difference between heads and tails isn't any worse than about 80, which isn't too terrible out of 1000 tosses.

Part i) Code:

```
figure;
for j = 1:100;
tosses = 0;
cointotal = 0;
coinarray = [];
for i = 1:1000;
   tosses = tosses + 1;
   if rand < .5       %%heads
      cointotal = cointotal + 1;
      coinarray = [coinarray cointotal/tosses];
   else
      cointotal = cointotal - 1;
      coinarray = [coinarray cointotal/tosses];
   end
end
coinarray
plot(1:1000,coinarray);
hold all;
end;
```



After normalizing part h, our graph reflects the observations we made from normalizing the histogram in part f. We see that when we toss less coins (smaller k), we are more likely to get outlier results such as all heads or all tails. When we toss a very large number of coins, we are closer to the origin, although the absolute size of the standard deviation increases.

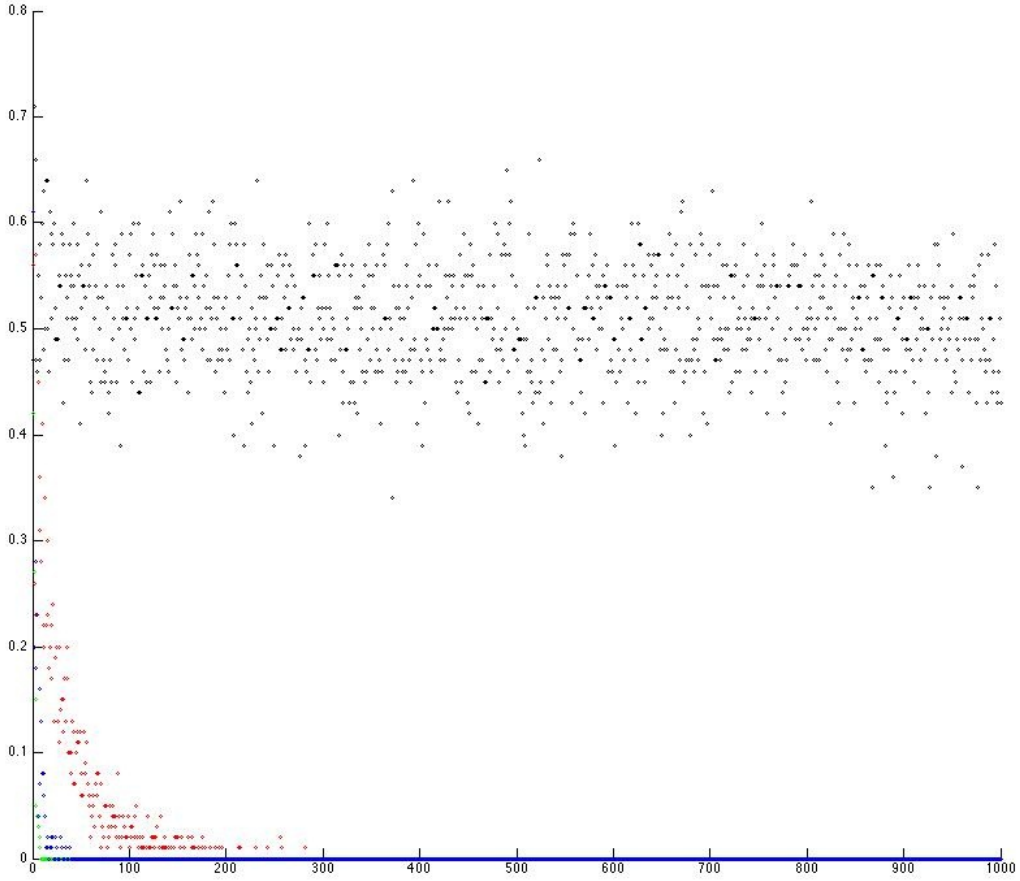
Part j) CODE:

```
figure;
q = .4;
m = 100;
for k = 1:1000;
frequency =0;
   for j = 1:m;
      heads = 0;
      headsfraction = 0;
      for i = 1:k;
         if rand < .5       %%heads
            heads = heads + 1;
         end
      end
      headsfraction = heads/k;
      if headsfraction <= q
         frequency = frequency + 1;
      end
   end
scatter(k,frequency/m,5,[1 0 0]);       %%red
hold all;
end

q = .1;
for k = 1:1000;
frequency =0;
   for j = 1:m;
      heads = 0;
      headsfraction = 0;
      for i = 1:k;
         if rand < .5       %%heads
            heads = heads + 1;
         end
      end
      headsfraction = heads/k;
      if headsfraction <= q
         frequency = frequency + 1;
      end
   end
scatter(k,frequency/m,5,[0 1 0]);   %%green
hold all;
end
```

```
q = .25;
for k = 1:1000;
frequency =0;
   for j = 1:m;
      heads = 0;
      headsfraction = 0;
      for i = 1:k;
         if rand < .5        %%heads
            heads = heads + 1;
         end
      end
      headsfraction = heads/k;
      if headsfraction <= q
         frequency = frequency + 1;
      end
   end
scatter(k,frequency/m,5,[0 0 1]);  %%blue
hold all;
end

q = .5;
for k = 1:1000;
frequency =0;
   for j = 1:m;
      heads = 0;
      headsfraction = 0;
      for i = 1:k;
         if rand < .5        %%heads
            heads = heads + 1;
         end
      end
      headsfraction = heads/k;
      if headsfraction <= q
         frequency = frequency + 1;
      end
   end
scatter(k,frequency/m,5,[0 0 0]);     %%black
hold all;
end
```

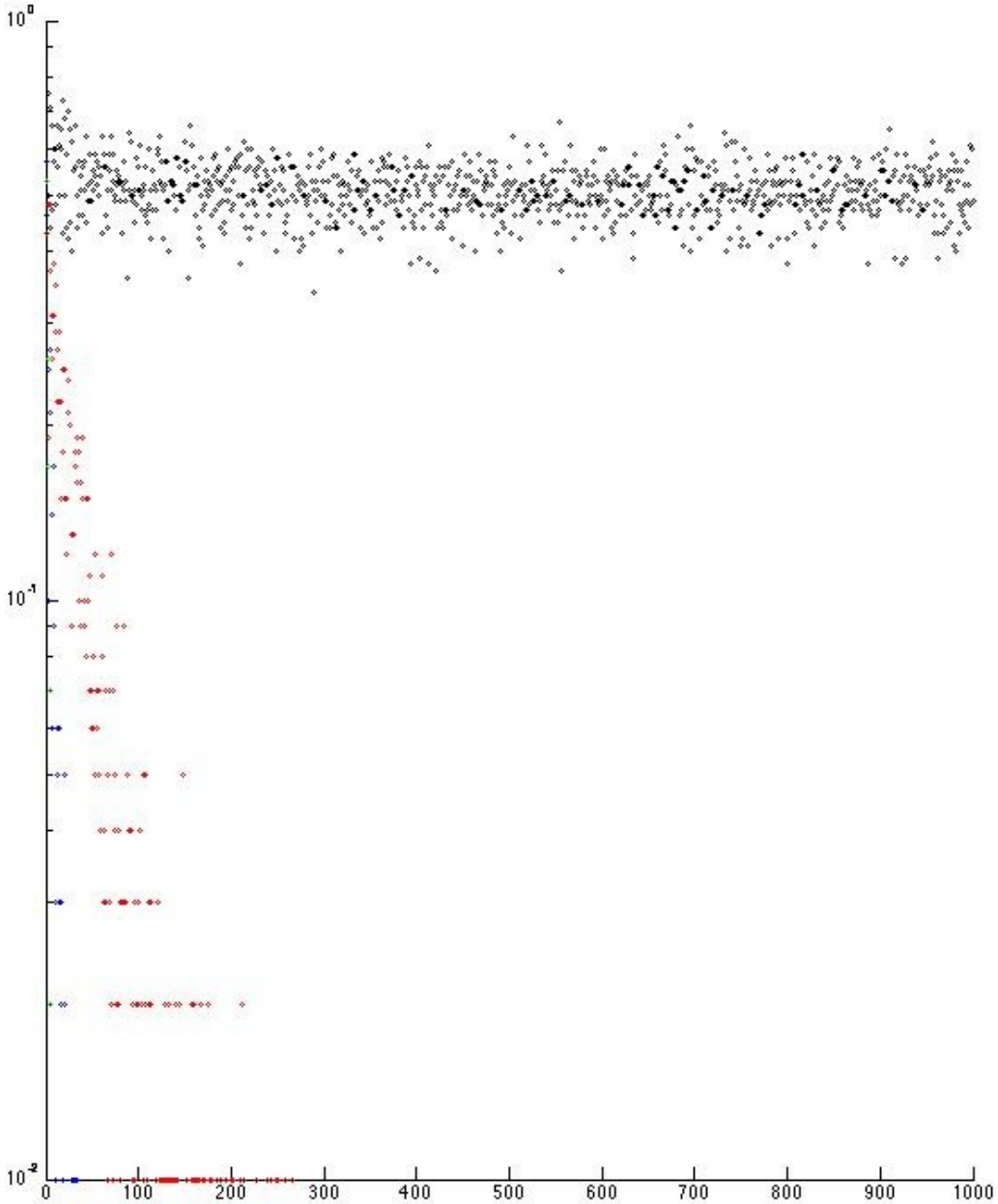


Note: I could only get the plot to generate when I made m a small number such as 100. Even with m = 1000, the computation time in matlab was so long that it didn't generate the plot. In theory, my code should work for m =10000 and will produce a plot; however given my time constraints i was unable to figure out whether there was a bug in my code or if there's a more efficient way to compute this part.

Part k) CODE:

```
figure;
q = .4;
m = 10000;
for k = 1:1000;
frequency =0;
   for j = 1:m;
      heads = 0;
      headsfraction = 0;
      for i = 1:k;
         if rand < .5       %%heads
            heads = heads + 1;
         end
      end
      headsfraction = heads/k;
      if headsfraction <= q
         frequency = frequency + 1;
```

```
        end
    end
scatter(k,frequency/m,5,[1 0 0]);       %%red
hold all;
end
set(gca,'YScale','log');
```



The only difference between this part and the previous part was that I used the set() function to change the Y scale to logarithmic. We see that the scatter plot spreads out more in the Y direction, and against a logarithmic axis the plots seems suggest straight lines (while on the linear axis they looked like decaying exponentials).