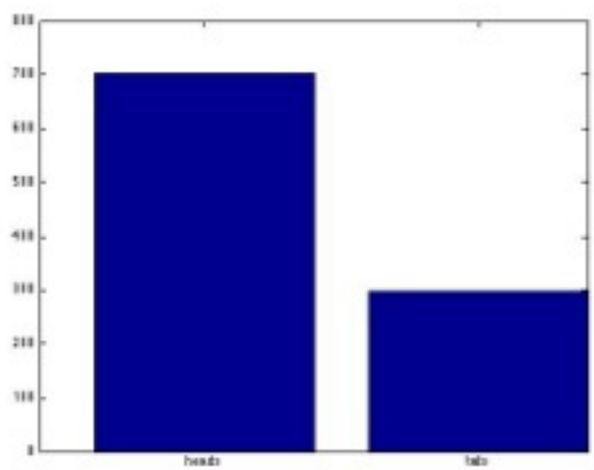
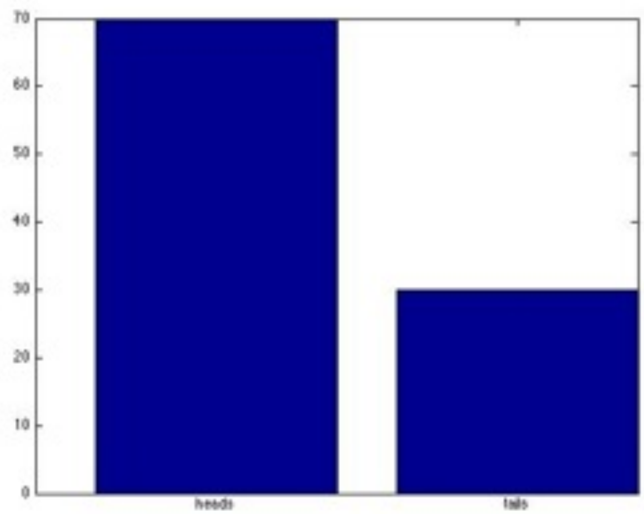
a)
```
heads = 0;
tails = 0;
p =  .5;       %%probability of heads
k = 1000;        %%%number of flips

for i= 1:k
  if rand <= p
      heads = heads + 1;
  else
      tails = tails + 1;
  end
end

heads
tails

figure
bar([heads tails])
set(gca,'xticklabel',{'heads'; 'tails'})
hold;
```



We expect to see that that the biased coin results in more heads than tails, as opposed to the fair coin which has equal probabilities for both.

b)
```
heads = 0;
tails = 0;
p =  .7;       %%probability of heads
k = 100;        %%%number of flips

for i= 1:k
  if rand <= p
```

```
      heads = heads + 1;
  else
      tails = tails + 1;
  end
end

heads
tails

figure
bar([heads tails])
set(gca,'xticklabel',{'heads'; 'tails'})
hold;
```



If we toss 100 coins, we expect to see about 70 of the coins heads.

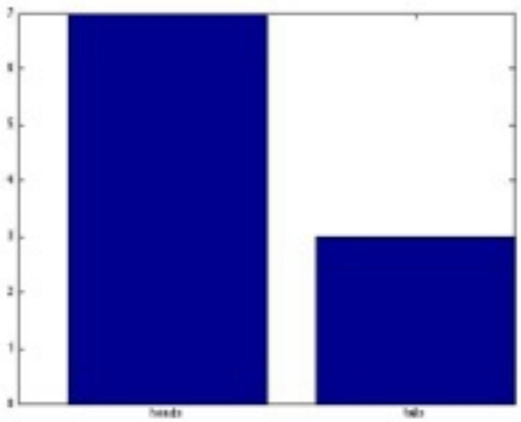
k = # of trials
p = probability of heads
p' = probability of tails

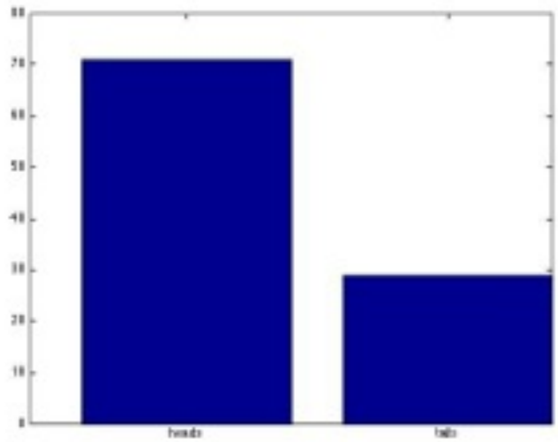
approximately:
#Heads = kp
#tails = kp'

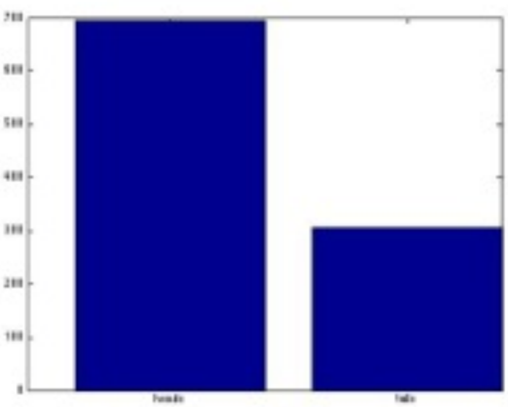
c)

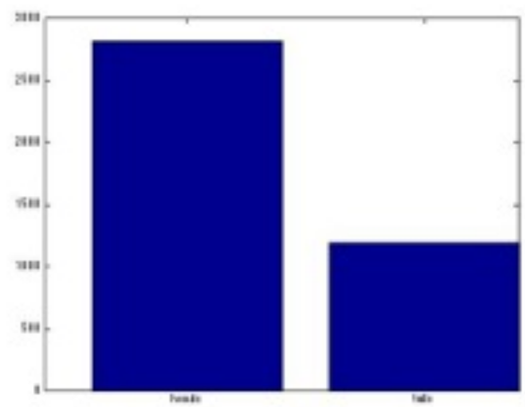
```
heads = 0;
tails = 0;
p =  .7;       %%probability of heads

for k = [10,100,1000, 4000];
    heads = 0;
    tails = 0;
    figure
    for i = 1:k
        if rand <= p
            heads = heads + 1;
        else
            tails = tails + 1;
        end
    end
```

```
    heads
    tails
    bar([heads tails])
    set(gca,'xticklabel',{'heads'; 'tails'})
end
```









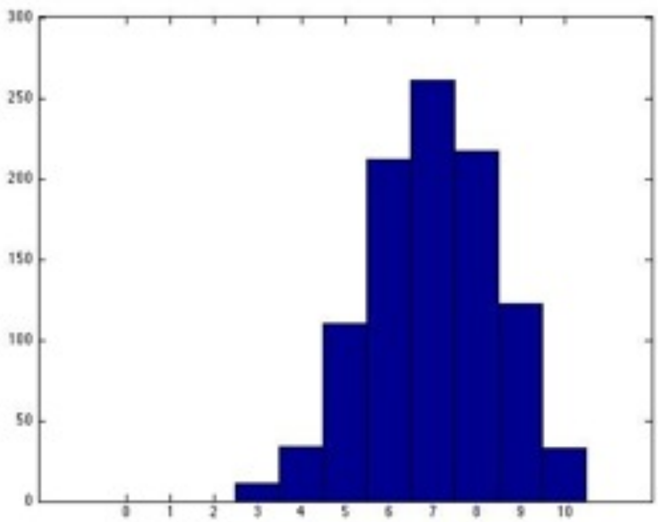
The total number of heads and tails line up with the equations we made in part b. This is what i expected after doing parts a and b. The equations seem to match up pretty well for all of these values of k, although I suspect I got lucky and using the law of large numbers I expect that larger k's will have results that match up closer to our equation.

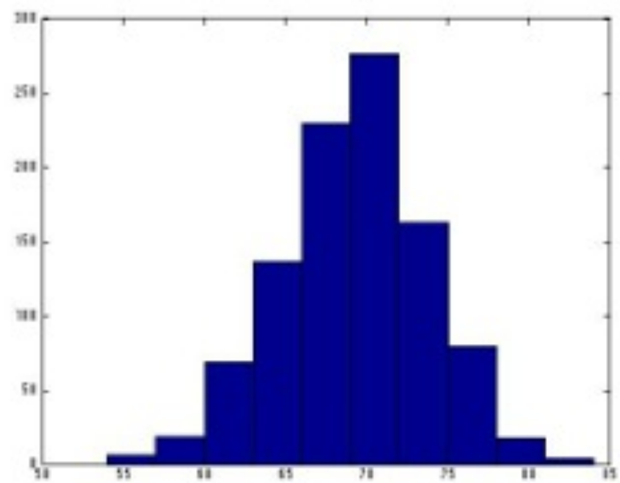
d)

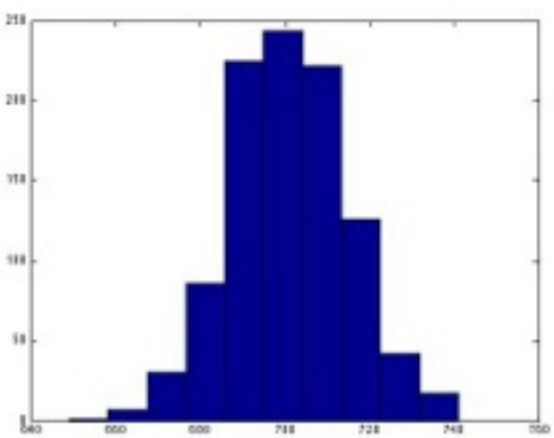
```
heads = 0;
tails = 0;
p =  .7;       %%probability of heads
m = 1000;

S_k= [];
    for j = 1:m
        heads = 0;
```

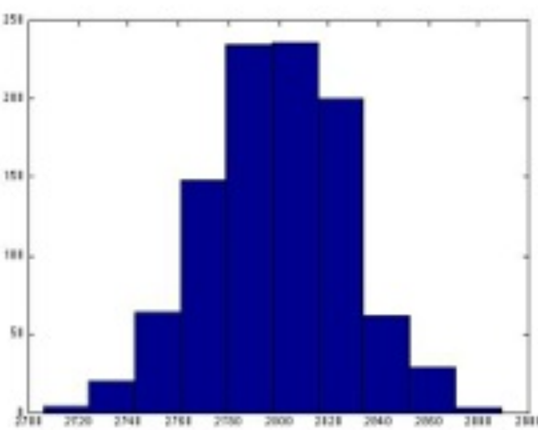
```
        tails = 0;
        for i = 1:10        %%k = 10
            if rand <= p
                heads = heads + 1;
            else
                tails = tails + 1;
            end
        end
        S_k = [S_k heads];
    end
S_k
figure
hist(S_k,[0 1 2 3 4 5 6 7 8 9 10])

S_k= [];
    for j = 1:m
        heads = 0;
        tails = 0;
        for i = 1:100       %%k = 100
            if rand <= p
                heads = heads + 1;
            else
                tails = tails + 1;
            end
        end
        S_k = [S_k heads];
    end
S_k
figure
hist(S_k)

S_k= [];
    for j = 1:m
        heads = 0;
        tails = 0;
        for i = 1:1000      %%k = 1000
            if rand <= p
                heads = heads + 1;
            else
                tails = tails + 1;
            end
        end
        S_k = [S_k heads];
    end
S_k
figure
hist(S_k)

S_k= [];
```

```
    for j = 1:m
        heads = 0;
        tails = 0;
        for i = 1:4000      %%k = 4000
            if rand <= p
                heads = heads + 1;
            else
                tails = tails + 1;
            end
        end
        S_k = [S_k heads];
    end
S_k
figure
hist(S_k)
```





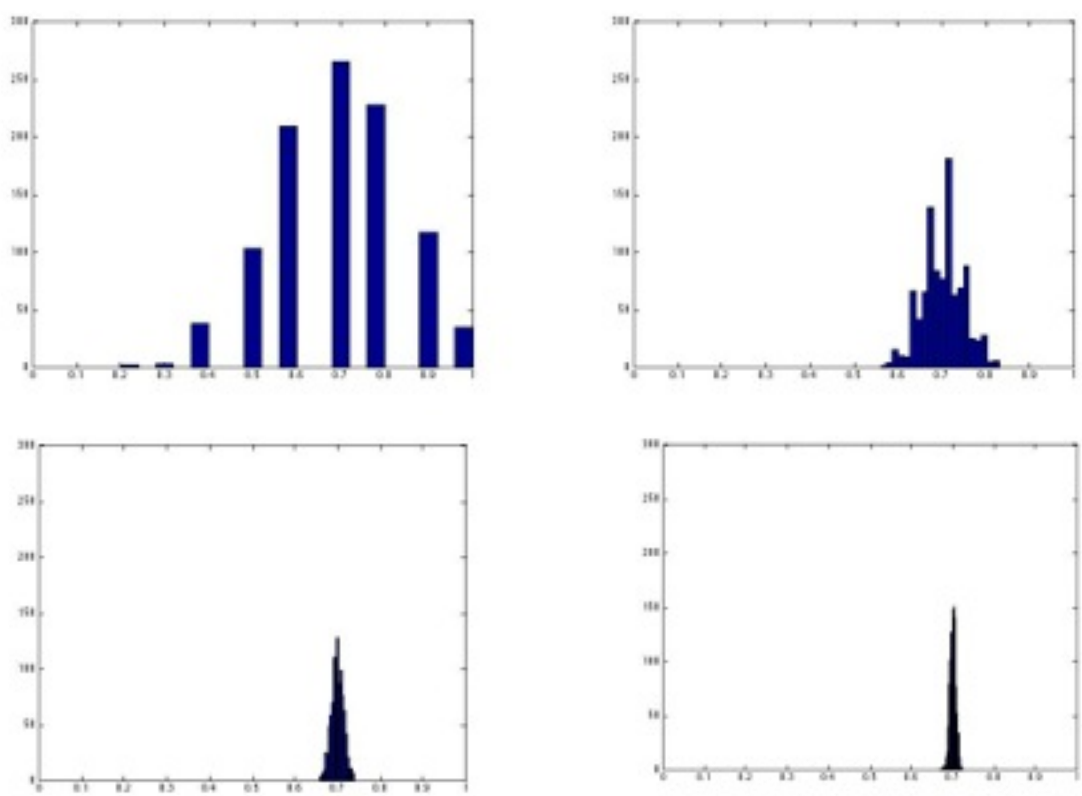




The results from this part confirm out intuition from part c. When we do a large number of trials, we see that our results tend to group around the value that we get from the equation in part b. The histogram shows us that the absolute spread in values is increasing, but it may be the case that the relative spread for the larger k's is smaller.

e)

```
heads = 0;
tails = 0;
p =  .7;       %%probability of heads
m = 1000;

for k = [10 100 1000 4000]
    S_k= [];
    for j = 1:m
        heads = 0;
        tails = 0;
        for i = 1:k        %%k = 10
            if rand <= p
                heads = heads + 1;
            else
                tails = tails + 1;
            end
        end
        S_k = [S_k heads/k];
    end
    S_k
    figure
    hist(S_k,20)
    axis([0 1 0 300])
end
```

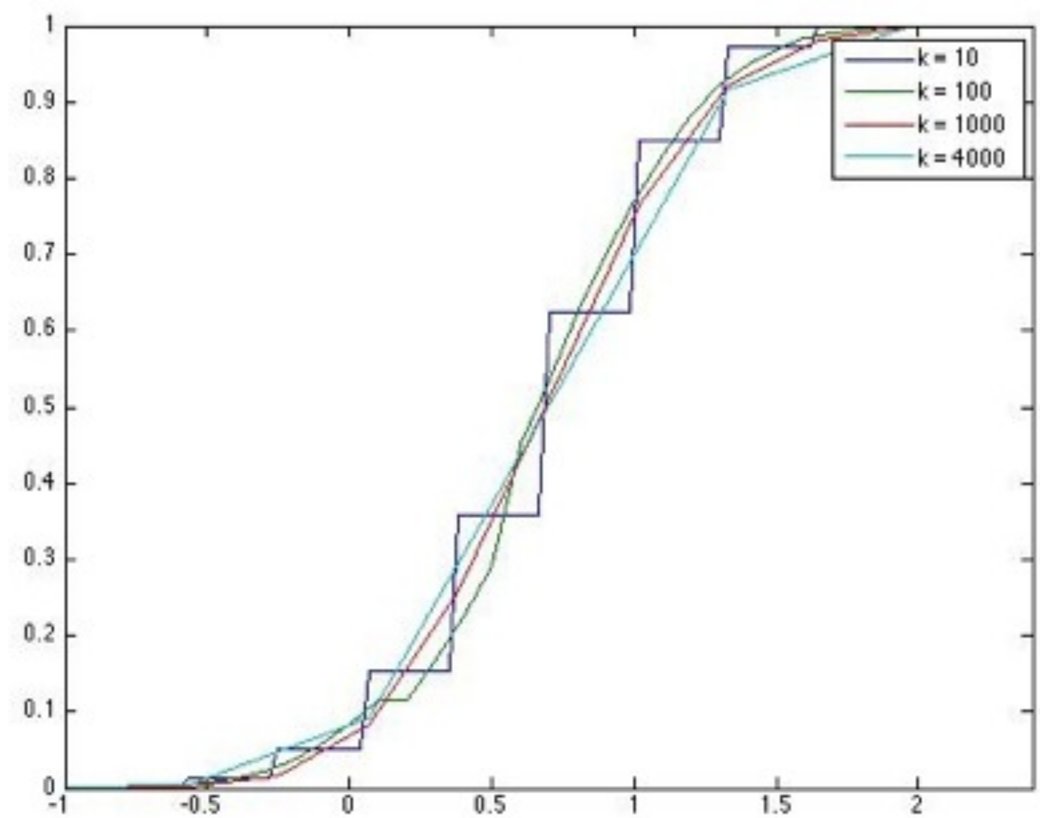


This part confirms what we suspected in the previous part. As k increases, the relative spread decrease significantly. With k = 4000, the histogram is quite sharp. In other words, as k becomes very large, the relative variance decreases, so our results match up with the expected probabilities more closely (they match the formula in part b with less percent error).

f)

```
function f();
    do_p();
end




function out = flip_coin_k_times_fraction (k)

    % flip an biased coin k times,
    % return fraction of heads

    out = 0;
    for idx = 1:1:k
        out = out + (rand() <= 0.7);
    end

    out = out/k;

end



function out = flip_coin_k_times_n_times_fraction (n, k)

    % run flip_coin_k_times_fraction n times

    out = zeros(n, 1);
    for idx = 1:1:n
        out(idx, 1) = flip_coin_k_times_fraction(k);
    end

end




function out = do_p ()

    n = 10000;

    ks = [10, 100, 1000, 4000];
    nks = size(ks, 2);
```

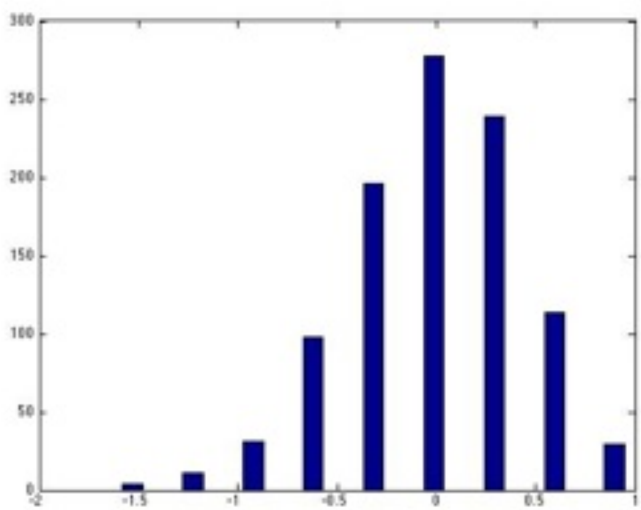
```
    qs = 0:0.01:1;
    nqs = size(qs, 2);

    figure();
    hs = [];
    ls = {};

    for kidx = 1:1:nks

        k = ks(1, kidx);
        head_fractions = flip_coin_k_times_n_times_fraction(n, k);

        ys = zeros(1, nqs);
        for qidx = 1:1:nqs
            q = qs(1, qidx);
            ys(1, qidx) = size(find(head_fractions <= q),1) / n;
        end
        h = plot((qs-0.7)*sqrt(k)+0.7, ys);
        hold all;
        hs = [hs, h];
        ls = [ls, ['k = ', num2str(k)]];
        axis([-1 2.4 0 1])

    end

    legend(hs, ls);

end
```

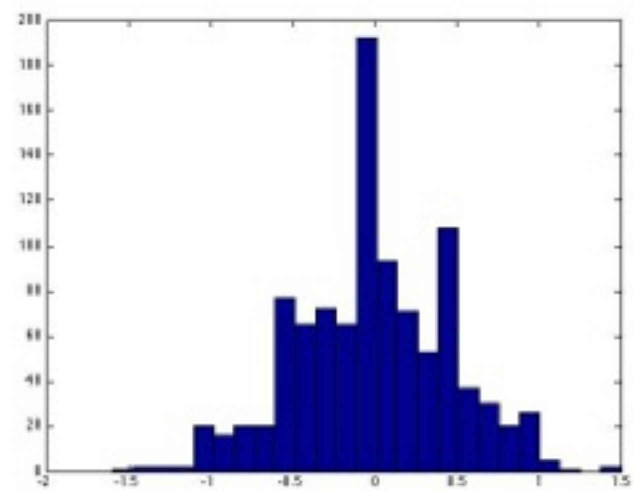


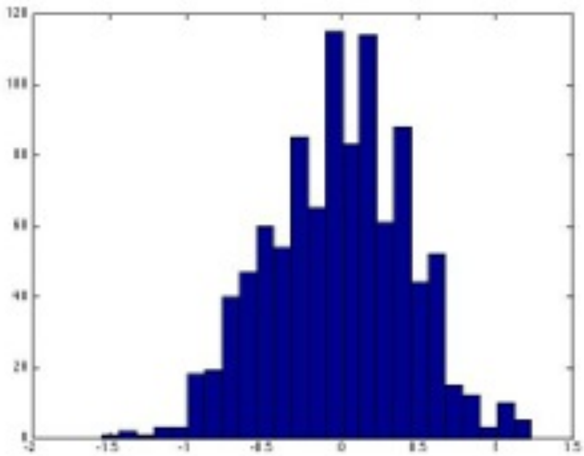
When we scale by sqrt(k), our S curves still fall right on top of each other, despite the fact that we have a biased coin now. The graph above is centered at q=p=.7

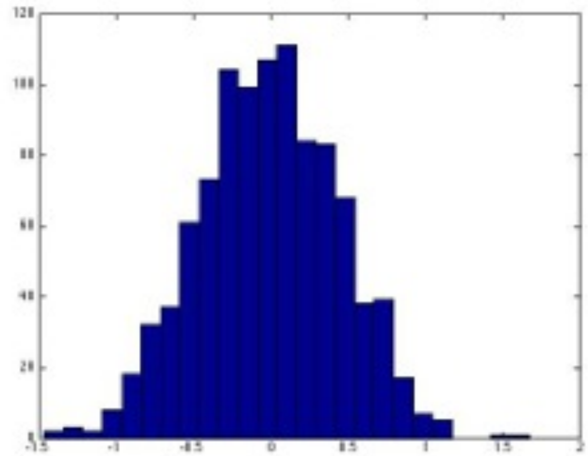
g)

```
heads = 0;
tails = 0;
p =  .7;      %%probability of heads
m = 1000;

for k = [10 100 1000 4000]
   S_k= [];
   for j = 1:m
      heads = 0;
      tails = 0;
      for i = 1:k       %%k = 10
         if rand <= p
            heads = heads + 1;
         else
            tails = tails + 1;
         end
      end
      S_k = [S_k (heads-k*p)/sqrt(k)];
   end
   S_k
   figure
   hist(S_k,25)
end
```









The normalization in part f produced a similar result for the histograms on this part. For the s curves in part f, having the curves fall on top of each other shows that with this scaling/ normalization, the variance for all k is the same. Since they coincide on the s curve graph, we expect to see that the histograms have the same peak, and in fact all k's have a histogram peak at 0. We also see that the histograms have the same variance on this scaling—the histograms seem to be clustered inside -1 to 1son all the graphs, with few outliers outside.

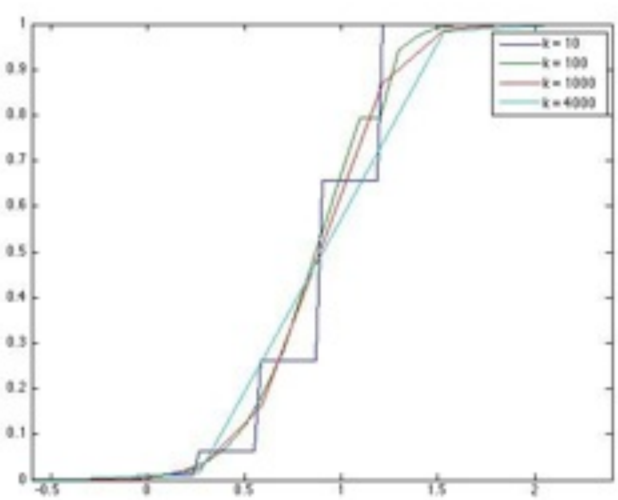
h)

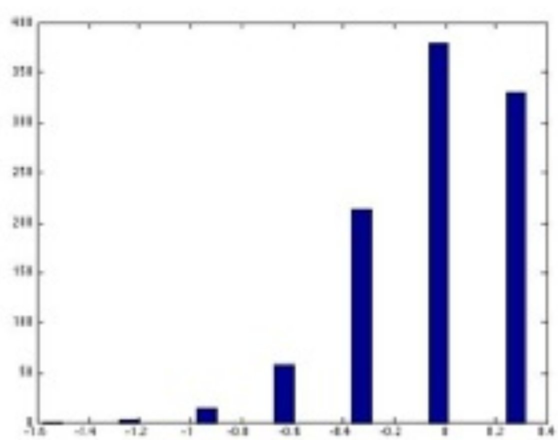
```
function h();
    for p = [.9 .6 .5 .4 .3]
        do_p(p);
        do_g(p);
    end
end




function out = flip_coin_k_times_fraction (k,p)

    % flip an biased coin k times,
    % return fraction of heads

    out = 0;
    for idx = 1:1:k
        out = out + (rand() <= p);
    end

    out = out/k;

end



function out = flip_coin_k_times_n_times_fraction (n, k,p)
```

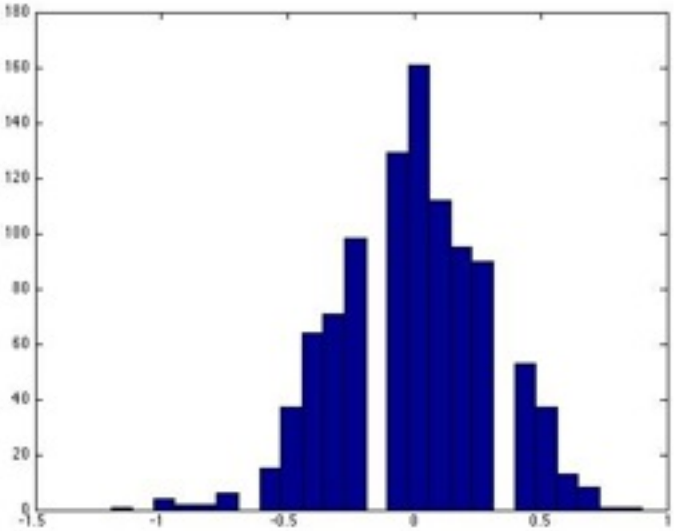
```
    % run flip_coin_k_times_fraction n times

    out = zeros(n, 1);
    for idx = 1:1:n
        out(idx, 1) = flip_coin_k_times_fraction(k,p);
    end

end




function out = do_p (p)

    n = 10000;

    ks = [10, 100, 1000, 4000];
    nks = size(ks, 2);

    qs = 0:0.01:1;
    nqs = size(qs, 2);

    figure();
    hs = [];
    ls = {};

    for kidx = 1:1:nks

        k = ks(1, kidx);
        head_fractions = flip_coin_k_times_n_times_fraction(n, k, p);

        ys = zeros(1, nqs);
        for qidx = 1:1:nqs
            q = qs(1, qidx);
            ys(1, qidx) = size(find(head_fractions <= q),1) / n;
        end
        h = plot((qs-p)*sqrt(k)+p, ys);
        hold all;
        hs = [hs, h];
        ls = [ls, ['k = ', num2str(k)]];
        axis([p-1.5 p+1.5 0 1])

    end

    legend(hs, ls);

end
```

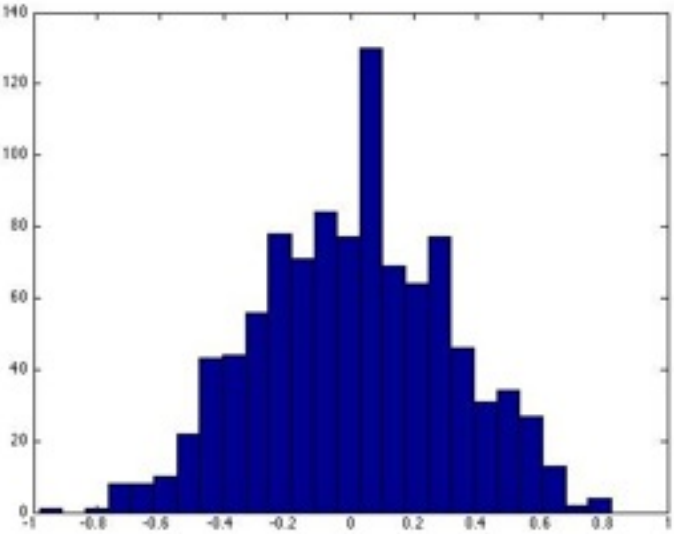
```
function out = do_g (p)
heads = 0;
tails = 0;
m = 1000;

for k = [10 100 1000 4000]
    S_k= [];
    for j = 1:m
        heads = 0;
        tails = 0;
        for i = 1:k        %%k = 10
            if rand <= p
                heads = heads + 1;
            else
                tails = tails + 1;
            end
        end
        S_k = [S_k (heads-k*p)/sqrt(k)];
    end
    S_k
    figure
    hist(S_k,25)
end
end
```

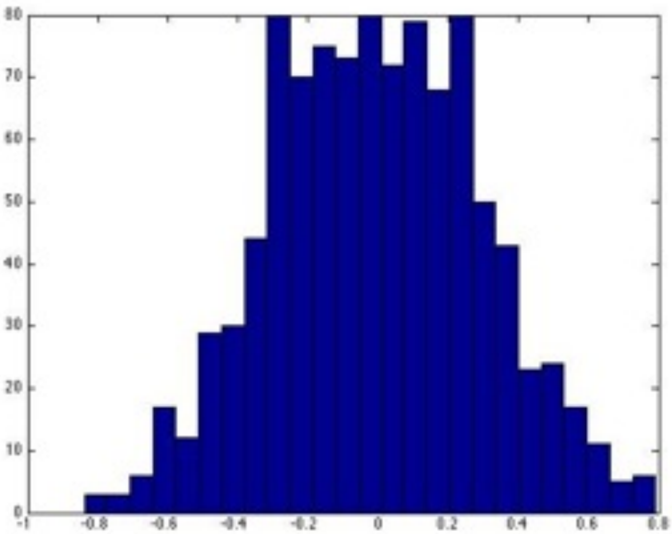
p = .9



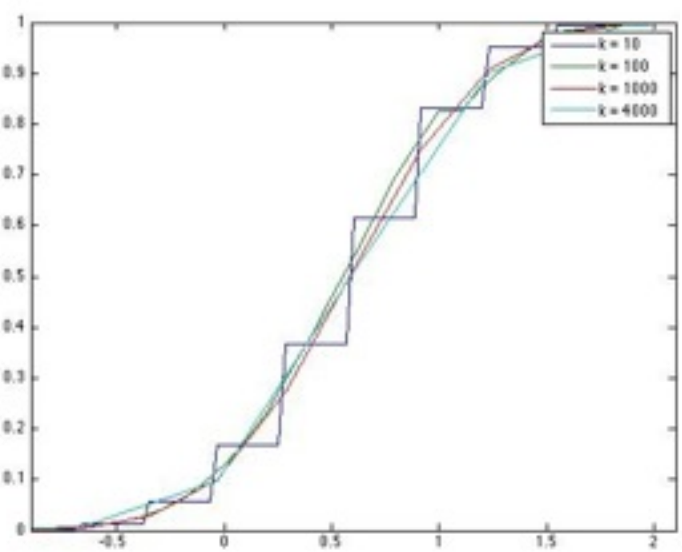


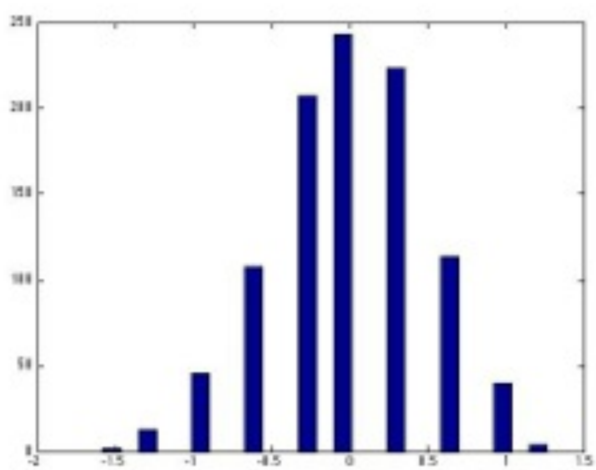


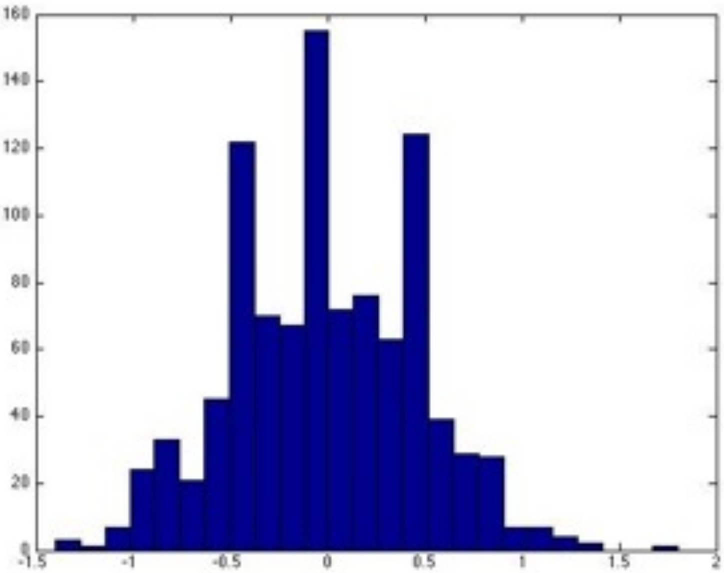


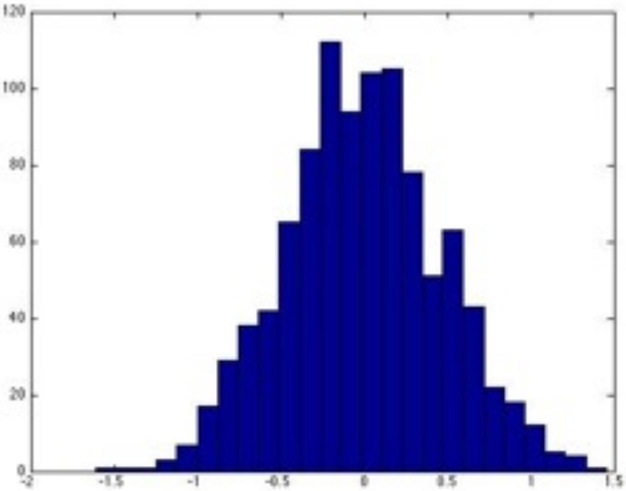


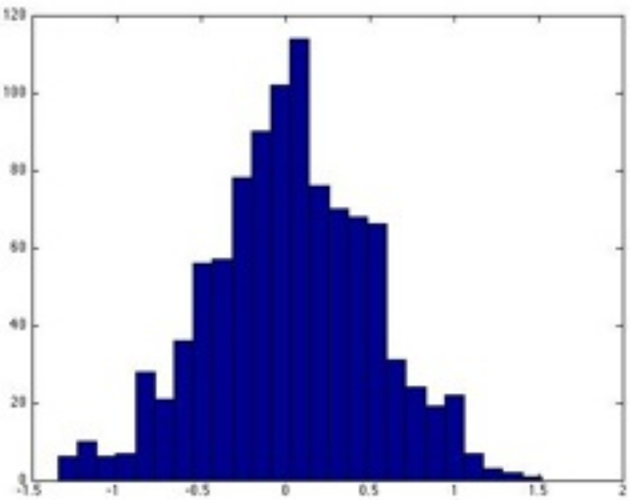
p = .6



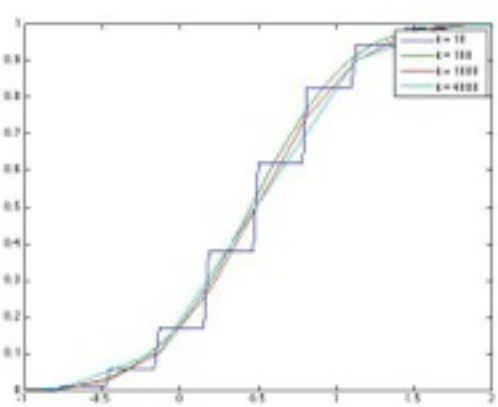


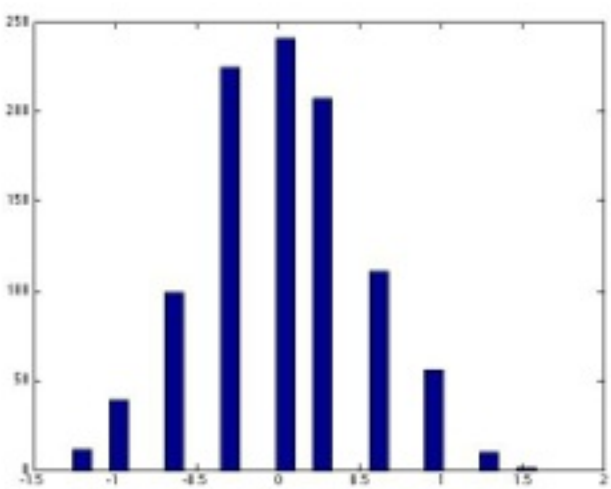

160
140
120
100
80
60
40
20
0
-1.5
-1
-0.5
0
0.5
1
1.5
2

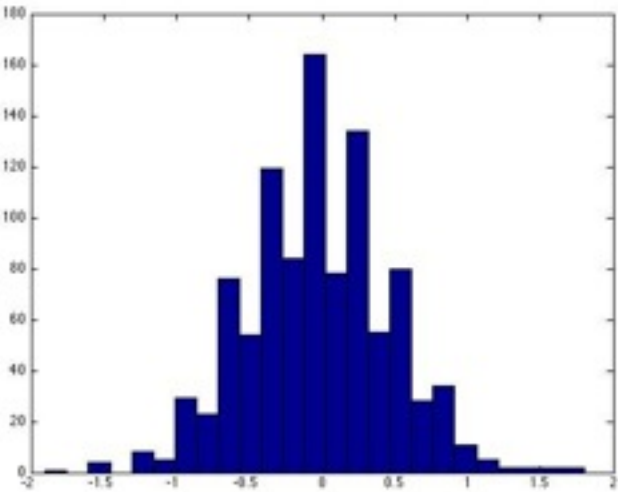

120
100
80
60
40
20
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5

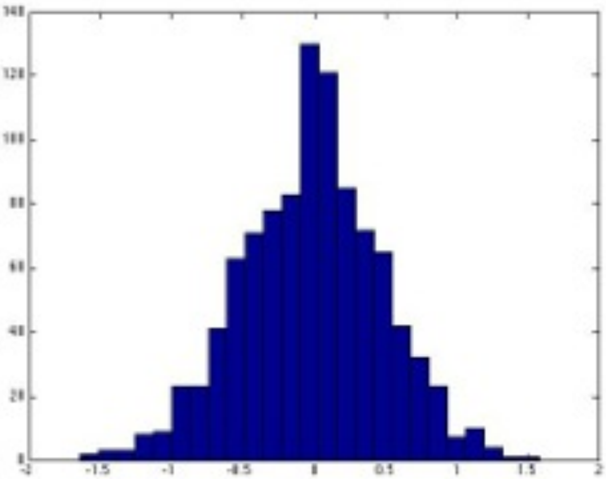

120
100
80
60
40
20
0
-1.5
-1
-0.5
0
0.5
1
1.5
2

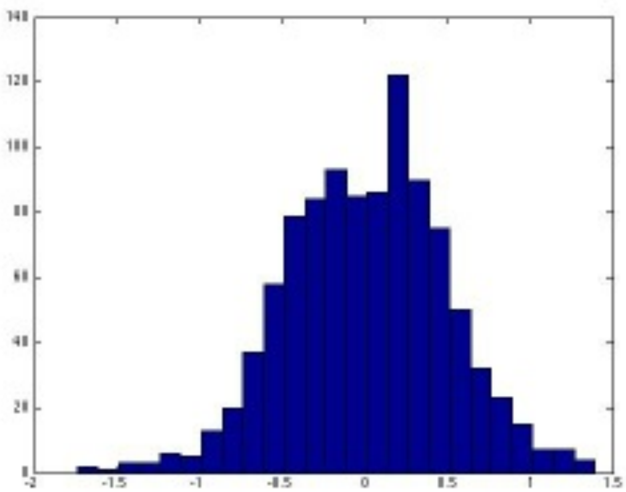
p = .5




250
200
150
100
50
0
-1.5
-1
-0.5
0
0.5
1
1.5
2







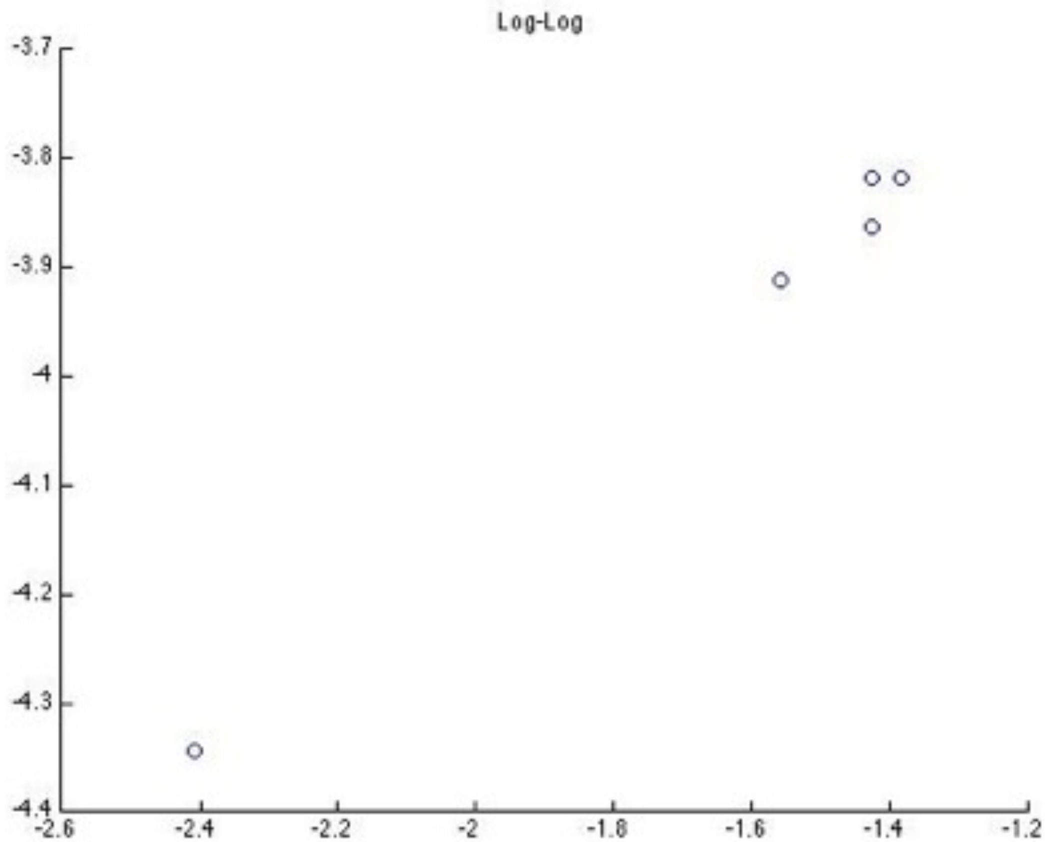
The histogram and S-curve graphs look very similar for p= .4 and p = .3. I have included them in the zip file, but due to time constraints and space on this pdf i have chosen not to include them. Overall, we see that for any probability p the S curves fall right on top of eachother, and similarly, the histograms have the same variance—most results fall between x = -1 and x = 1.

i)

```
function i()
   do_o();
end

function out = flip_coin_k_times_fraction (k, p)

   % flip a biased coin k times,
   % return fraction of heads
```

```
    out = 0;
    for idx = 1:1:k
        out = out + (rand() <= p);
    end

    out = out/k;

end

function out = flip_coin_k_times_n_times_fraction (n, k, p)

    % run flip_coin_k_times_fraction n times

    out = zeros(n, 1);
    for idx = 1:1:n
        out(idx, 1) = flip_coin_k_times_fraction(k,p);
    end

end
function out = do_o (p)
    figure
    hold all
    ps = [.9 .6 .5 .4 .3];
    yys = [];
    for p = [.9 .6 .5 .4 .3]

    n = 10000;

    ks = [1000];
    nks = size(ks, 2);

    qratios = [0.25, 0.5, 0.75];
    nqrs = size(qratios, 2);

    qmarkers = zeros(nks, nqrs);

    for kidx = 1:1:nks

        k = ks(1, kidx);
        head_fractions = flip_coin_k_times_n_times_fraction(n, k , p);

        sorted_head_fractions = sort(head_fractions);
        for qridx = 1:1:nqrs
            qr = qratios(1, qridx);
            qmarkers(kidx, qridx) = sorted_head_fractions(ceil(qr*n), 1);
        end

    end
```

```
    ys = (qmarkers(:,3) - qmarkers(:,1))';
    yys =[yys ys];
    end
    yys
    log(yys)

    scatter(log(ps.*(1-ps)), log(yys));
    title('Log-Log')

end
```



On the log-log scale, we see that the scatter plot for different p's is roughly a straight line through the origin. This means we can use a normalization of sqrt(p(1-p)) for the histograms in the next part.

j)

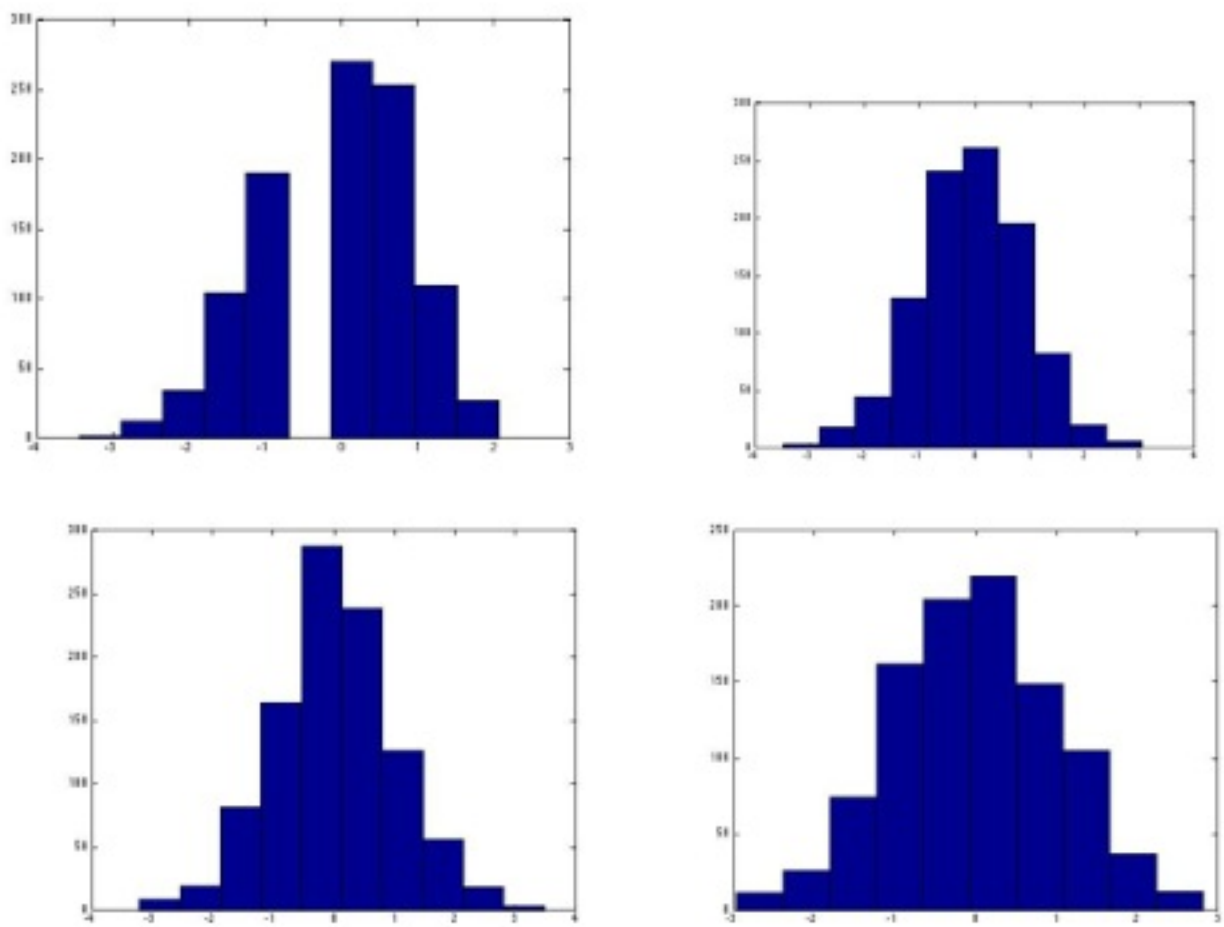
```
for p = [.7 .9 .6 .5 .4 .3]
heads = 0;
tails = 0;
m = 1000;
```

```
for k = [10 100 1000 4000]
    S_k= [];
    for j = 1:m
        heads = 0;
        tails = 0;
        for i = 1:k
            if rand <= p
                heads = heads + 1;
            else
                tails = tails + 1;
            end
        end
        S_k = [S_k (heads-k*p)/(sqrt(k)*sqrt((p*(1-p))))];
    end
    S_k;
    figure
    hist(S_k)

end
end
```

j)

```
for p = [.7 .9 .6 .5 .4 .3]
heads = 0;
tails = 0;
m = 1000;

for k = [10 100 1000 4000]
    S_k= [];
    for j = 1:m
        heads = 0;
        tails = 0;
        for i = 1:k
            if rand <= p
                heads = heads + 1;
            else
                tails = tails + 1;
            end
        end
        S_k = [S_k (heads-k*p)/(sqrt(k)*sqrt((p*(1-p))))];
    end
    S_k;
    figure
    hist(S_k)

end
end
```

For this part, I'll just show the plots for p = .7. The other plots are very similar. Again we see that this normalization puts our histograms within the same range across different values of k. For the case of p = .7, the histogram is roughly situated inside -2 and 2. In other words, the variance for different k's on this scale is the same, like in the previous part.



k)

```
function k();
   for p = [.7 .9 .6 .5 .4 .3]
      do_p(p);
   end
end
```

```
function out = flip_coin_k_times_fraction (k,p)

    % flip an biased coin k times,
    % return fraction of heads

    out = 0;
    for idx = 1:1:k
        out = out + (rand() <= p);
    end

    out = out/k;

end



function out = flip_coin_k_times_n_times_fraction (n, k,p)

    % run flip_coin_k_times_fraction n times

    out = zeros(n, 1);
    for idx = 1:1:n
        out(idx, 1) = flip_coin_k_times_fraction(k,p);
    end

end



function out = do_p (p)

    n = 10000;

    ks = [100, 1000, 4000];
    nks = size(ks, 2);

    qs = 0:0.01:1;
    nqs = size(qs, 2);

    figure();
    hs = [];
    ls = {};

    for kidx = 1:1:nks

        k = ks(1, kidx);
        head_fractions = flip_coin_k_times_n_times_fraction(n, k, p);

        ys = zeros(1, nqs);
```

```
        for qidx = 1:1:nqs
            q = qs(1, qidx);
            ys(1, qidx) = size(find(head_fractions <= q),1) / n;
        end
        h = plot((qs-p)*sqrt(k)*sqrt(p*(1-p))+p, ys);
        hold all;
        hs = [hs, h];
        ls = [ls, ['k = ', num2str(k)]];
        axis([p-1.5 p+1.5 0 1])

    end

    legend(hs, ls);

end
```

For part k, all the s curves line up with each other.