Rene Zamudio (008634559) – STAT 572 – Dr. Sung Kim – Homework 2

Part A: In-class assignment:
(a) Use the inverse CDF method to generate a random sample of size 1000 from Binomial distribution with n=20 and p=0.3. Give the relative frequency histogram of your random sample and compare with the theoretical pdf of Binomial distribution.
(b) Repeat (a) with n=10 and p=0.05. Discuss your findings.

Steps for coding:

My approach here was to use the hint give in class, to use BINOCDF. I looked into how it works and practiced with it at first. Next, the plan was to create a loop that will generate uniform random numbers and run the results through another loop. The purpose of the second loop is to not have to repeat and if-else, where the BINOCDF and U values are compared. The desired results will be stored as Xi. I will change n and p for part A and B.

Part A)

Code:
```
clc, clear
n = 1000;
int=0:20;
bin=binocdf(int, 20, 0.3);

% figure(2), plot(xx,y)

for i = 1:n
    u = rand(1);
    for j = 2:length(int)
        if u <= bin(1)
            x(i) = 0;
        elseif u > bin(j-1) && u <= bin(j)
            x(i) = j-1;
        end
    end
end
histogram(x)
title('Inverse CDF with n = 20, p = 0.30')
```

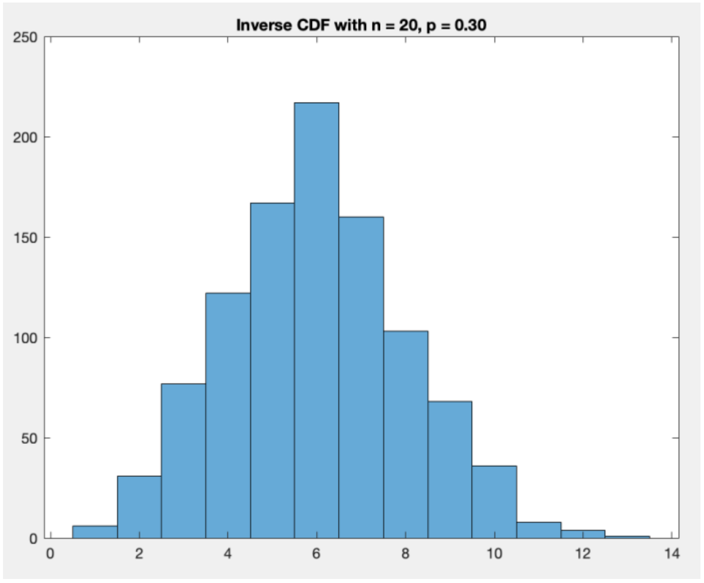
Part B)

Code:
```
clc, clear
n = 1000;
int=0:20;
bin=binocdf(int, 20, 0.3);

% figure(2), plot(xx,y)

for i = 1:n
    u = rand(1);
    for j = 2:length(int)
        if u <= bin(1)
```

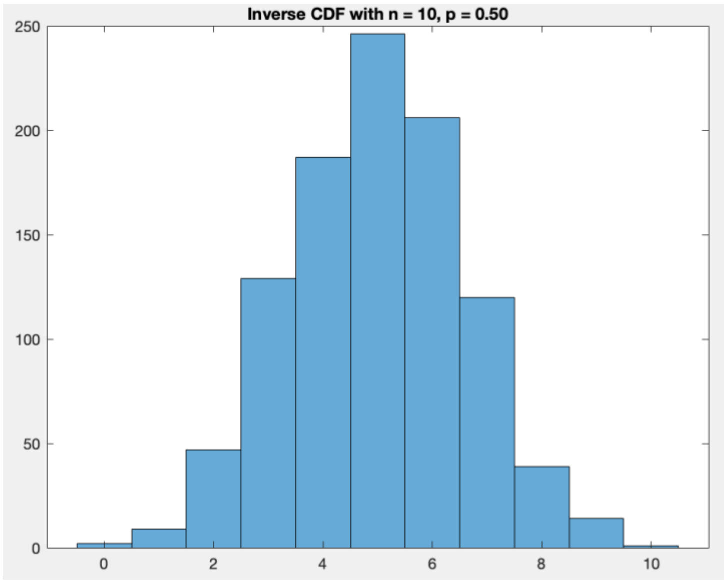
```
            x(i) = 0;
        elseif u > bin(j-1) && u <= bin(j)
            x(i) = j-1;
        end
    end
end
histogram(x)
title('Inverse CDF with n = 20, p = 0.30')
```

Output:

PART A



PART B



Discussion:
As mentioned in the Steps part above, I used the hint for this code and it worked out really well. The loop made it all easier as well. I can see that based on the given n's and p's, the inverse CDF histograms worked perfectly. In part A, n = 20, and p = 0.3; therefore, np = 6 and we can see that the center of the histogram is at 6, which means it is the mode and approximately the mean as well. In part B, n = 10, and p = 0.5; therefore, np = 5 and we can see that the center of the histogram is at 5, which means it is the mode and approximately the mean as well. Both of the histograms and results agree with the Binomial Distribution and the mean = np. I would like to add on that making loops helps so much with shortening code and working faster.

Rene Zamudio (008634559) – STAT 572 – Dr. Sung Kim – Homework 2

Part B: Textbook problems: Chapter 4: #6, 7, 9

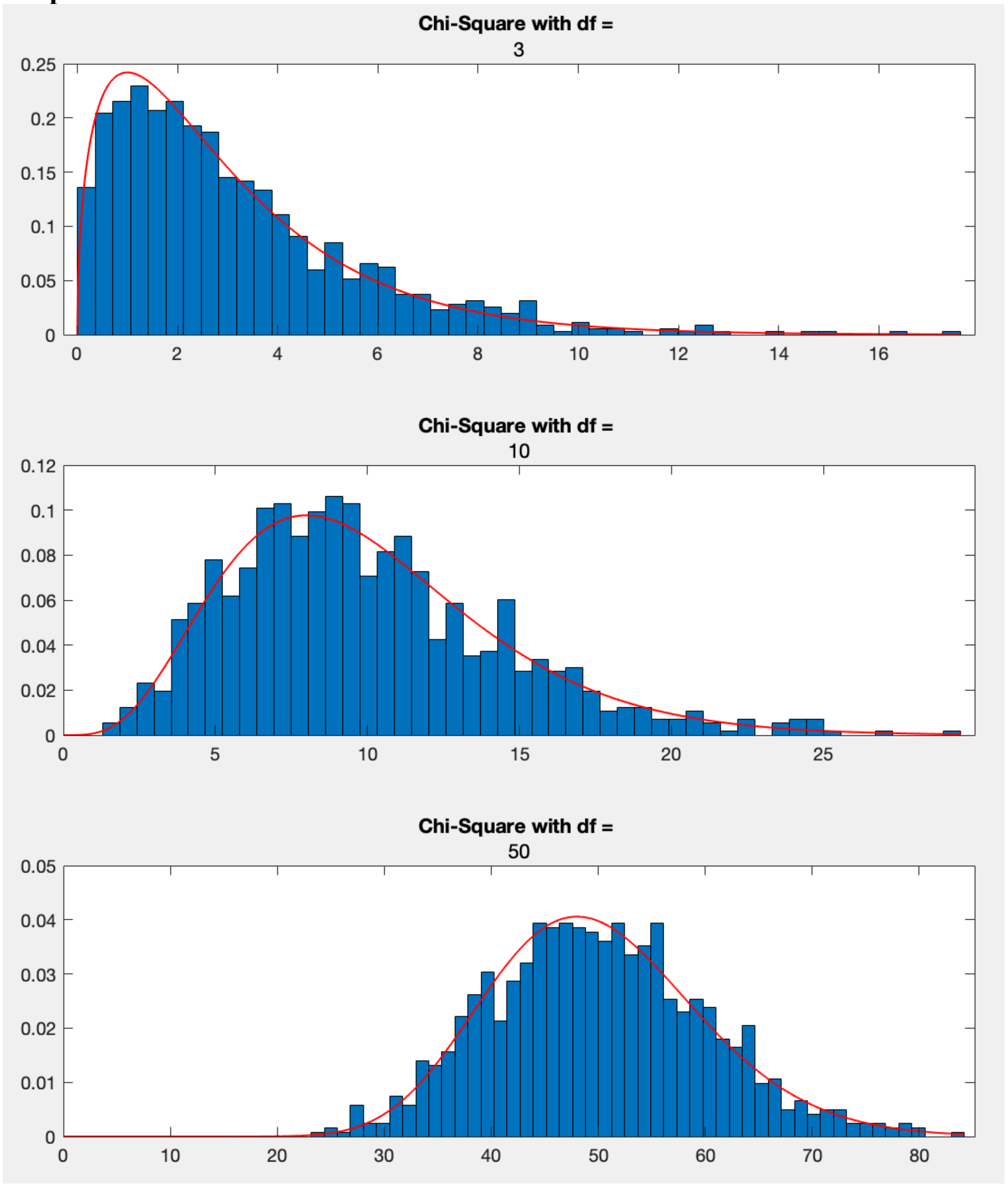
Exercise 4.6 Pg 115
Write a function that will generate chi-square random variables with ν degrees of freedom by generating ν standard normals, squaring them and then adding them up. This uses the fact that X = Z1^2 + ... + Zν^2 is chi-square with ν degrees of freedom. Generate some random variables and plot in a histogram. The degrees of freedom should be an input argument set by the user.

Additional:
Test your function with three different degrees of freedom of your choice.
Draw relative frequency histogram of your random samples and superimpose the true densities. Discuss your findings.

Steps for coding:
As we learned in STAT 520, the summation of n standard normal squared will give us a chi square distribution with n degrees of freedom. We are now putting it into test and graphing the frequency histogram, along with the superimposed density. I wanted to make this as "automatic" as possible so I created two loops, which are detailed with the comments below. The inner loop creates the standard normal squares and stores them as Zi. After that loop is over, the sum is stored as Xj (a chi square). This is done 1000 times to create the desired plots. The degrees of freedom values that I chose to use are 3, 10, and 50.

Code:
```
clear,clc
% prompt = 'Choose the degrees of freedom: ';
% df = input(prompt);
df = [3 10 50]; % choosing my three different df
n = 1000;

for k=1:length(df)       % loop for the three degrees of freedom
    for j=1:n            % loop for n = 1-1000
        for i=1:df(k)    % loop for creating Zi^2's
            Z(i)=randn(1)^2; % Z vector
        end
        X(j)=sum(Z);     % adding up Zi^2 and storing as Xi
    end
    subplot(3,1,k)       % to view all three histograms in one figure
    [N, h] = hist(X,50); % histogram of X for each df
    % Change bar heights.
    N = N/(h(2)-h(1))/n;
    % Superimposing the density curve
    x = 0:.01:max(X);
    y = chi2pdf(x,df(k));
    plot(x,y), axis equal, bar(h,N,1), hold on
    plot(x,y,'r', 'LineWidth',1), hold off
    title('Chi-Square with df = ', df(k))
end
```

Output:



Discussion:
This was probably my most favorite of all the homework questions because of how well it worked and how neat everything came out in the output. When it comes to chi-square, we know

that the expected value = mean = degrees of freedom. We can see that all three of these plots help support that idea. The mode of the first two plots are less than the expected value, but since there is noticeable skewness, we can see how the mean will be greater than the mode in those examples. In the last plot, the mean and mode are more aligned and helps support that a chi square will become more bell-shaped as the degrees of freedom increase. Superimposing the density was a little more challenging, but then I read about how to use the CHI2PDF function. I then created the x and y values to help plot this and just added it on top in red. I believe that they fit nicely. Once again, loops helped shorten everything.

Exercise 4.7 Pg 115
An alternative method for generating beta random variables is described in Rubinstein [1981]. Generate two variates Y1 = U1^1⁄α and Y2 = U2^1⁄β, where the Ui are from the uniform distribution. If Y1 + Y2 ≤ 1, then X = Y1/(Y1 +Y2), is from a beta distribution with parameters α and β. Implement this algorithm.

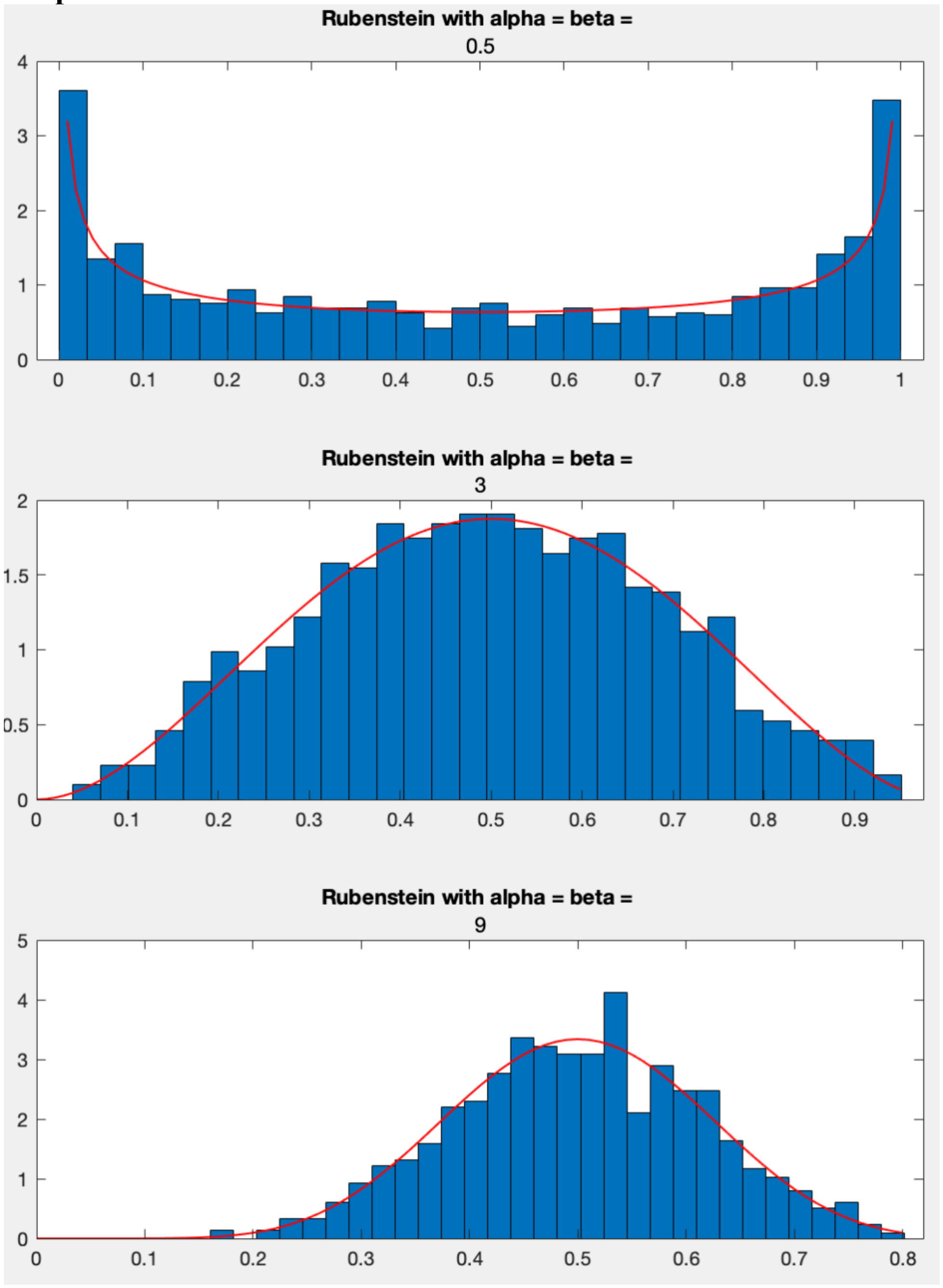
Additional:
use (a) alpha=beta=0.5. (b) alpha=beta=3, 9. Draw relative frequency histogram of your random samples and superimpose the true densities. Discuss your findings.

Steps for coding:
The first important step to this was to make the function for the Rubinstein [1981] to generate beta random variables. Not only will it help the code look neater, but it will allow me to refer to the function in the future. Only one loop was necessary for this code since the Rubenstein function that is given did all of the work. This allowed me to get created with storing values. I chose to use a multidimensional array to keep track of all the data. I have tried to do this in the past homework, but the dimensions of the arrays needed to match; this time, they do match with all having 1000 indexes. I then chose to use subplots to keep them all in the same figure.

Code:

```
n = 1000;
a_b = [0.5, 3, 9]; % alpha and beta values

for i = 1:3
    data(:,:,i) = rubinstein(a_b(i),a_b(i),n);
    subplot(3,1,i)
    [N, h]=hist(data(:,:,i),30);
    % Change bar heights.
    N = N/(h(2)-h(1))/n;
    % Now get the theoretical probability density.
    x = 0:0.01:max(data(:,:,i));
    y = betapdf(x, a_b(i), a_b(i));
    plot(x,y), axis equal, bar(h,N,1), hold on
    plot(x,y,'r', 'LineWidth',1), hold off
    title('Rubenstein with alpha = beta = ', a_b(i))
end


--------------------------------------------------------------------------
function data = rubinstein(alpha,beta,n)

i=1; % counter for index
while i <= n
    Y1 = rand(1)^(1/alpha); % Generating the two variates
    Y2 = rand(1)^(1/beta);  % and using rand for uniform distr.
    if Y1+Y2 <= 1
        data(i)=Y1/(Y1+Y2);
        i=i+1; % only inc. i if data has new value to guarantee n size
    end
end
```

Output:



Discussion:
Once I got the main idea of it, which was to ensure that Rubenstein worked, I wanted to fit all of the plots into the same output. This allowed me to get a little more creative with loops and multi-dimensional arrays. That was honestly fun to try out because, as I mentioned in the Steps part above, I have wanted to try it out before, but the sizes would not match. We can see here that everything worked nicely and the manner in which equal alpha and beta values worked matched

nicely with the graphs. The expected values = mean = alpha/(alpha+beta) = ½ = 0.5 were all the same and we can see that all of the graphs agree. The center of each was expected to be at 0.5. The first plot was the only one that had alpha and beta less than 1 which is what make the interesting U-shape. Another example of a different plot would have been, alpha=beta=1, which would have given us a uniform distribution. I used BETAPDF to help superimpose and once again it was done in red.

Exercise 4.9 Pg 116
Implement example 4.5 in MATLAB. Generate 100 random variables. What is the relative frequency of each value of the random variable 1, ..., 5 ? Does this match the probability mass function?

Additional:
Repeat with n=1000 and compare the results.

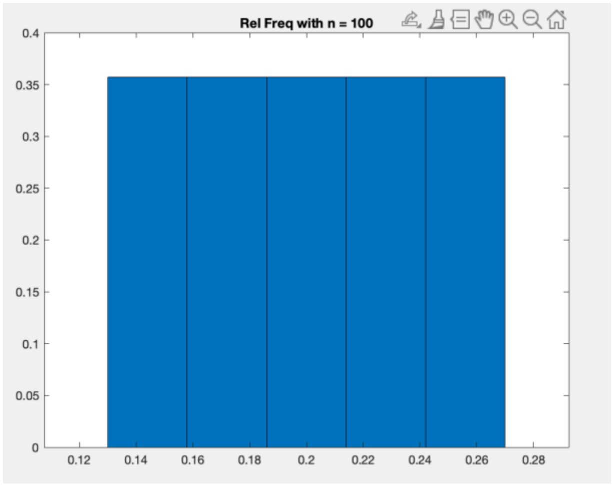
Steps for coding:
This was probably the toughest for me. I first misunderstood the meaning of example 4.5 and referred to exercise 4.5 instead. Once that was fixed, I referred to the notes about the accept-reject method. I looked at the examples provided as well and eventually came up with the following. I have separate codes for part A and part B since I felt rushed. I also have provided a histogram for each, along with the text frequencies. The histograms do not really help in this case, but I included them since it was part of my code and output. I have provided comments throughout the code.

Code:
PART A
```
n = 100; % sample size of 100
pr = [.15 .22 .33 .10 .2]; % 5 given values from Ex 5
N = length(pr);
c = .33;
irv = 0;
irej = 0;
% accept-reject method is used to generate rv's
while irv <= n
    Y = unidrnd(5);      % random 1-5
    U = rand(1);         % uniform random
    if U <= pr(Y)/c      % cannot be greater for this method
        irv = irv + 1;   % next irv
        X(irv) = Y;      % Xi =Y
    else
        irej = irej + 1;
    end
end

for i = 1:N
    freq(i) = length(find(X==i)); % frequency of X==i occurring
end

relf = freq/n %relative frequency

[N, h]=hist(relf,5);
% Change bar heights.
N = N/(h(2)-h(1))/n;
plot(X,Y), axis equal, bar(h,N,1), hold on
title('Rel Freq with n = 100')
```

PART B
```
n = 1000; % sample size of 100
pr = [.15 .22 .33 .10 .2]; % 5 given values from Ex 5
```

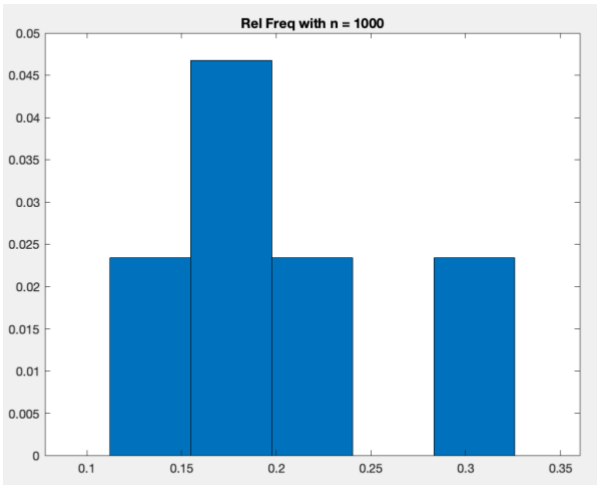
```
N = length(pr);
c = .33;
irv = 0;
irej = 0;
% accept-reject method is used to generate rv's
while irv <= n
    Y = unidrnd(5);     % random 1-5
    U = rand(1);        % uniform random
    if U <= pr(Y)/c     % cannot be greater for this method
        irv = irv + 1;  % next irv
        X(irv) = Y;     % Xi =Y
    else
        irej = irej + 1;
    end
end

for i = 1:N
    freq(i) = length(find(X==i)); % frequency of X==i occurring
end

relf = freq/n %relative frequency

[N, h]=hist(relf,5);
% Change bar heights.
N = N/(h(2)-h(1))/n;
plot(X,Y), axis equal, bar(h,N,1), hold on
title('Rel Freq with n = 1000')
```

Output:
PART A



```
relf =

     0.1700    0.2700    0.2400    0.1300    0.2000
```

PART B



```
relf =

    0.1560    0.2170    0.3260    0.1120    0.1900
```

Discussion:
As mentioned previously, this was the toughest, in my opinion. I completely redid it with little time to spare since I had an exam today, but I believe that it is now complete. We have 5 frequency values being printed for each of the parts. We can see that in Part B the frequency outputs have numbers that are far closer to the Pr = [.15 .22 .33 .10 .2] values, which makes sense since we are increasing the sample size, n. In Part A, that is not the case, which also make sense since n is only 100 (decent, but not the best of the two). Overall, PART B is far closer to the PMF given.