STAT572 - Homework Assignment 3

Juan Carlos Apitz, ID 012523821

September 25, 2014

In-Class Exercises

Exercise 3: Generate Random Sample Using the Accept-Reject Method

In this exercise we generate the random sample for an rv $X \sim f(x) = 20x(1-x)^3$, $0 < x < 1$.The rejection rate for one run to generate a sample size of 1,000 is $\dfrac{1{,}096}{2096} = 0.5229$, or 52.29%. Which is expected since $c = 2.1$.

```
Editor - /home/jcapitz/Documents/Stat572/labs_inclass/9_18_14/accept_reject_inclass.m
gammars.m | csgammp.m | question3pt1.m | ex4_7.m | accept_reject_inclass.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Modified Example 4.4                                            %
% Computational Statistics Handbook with MATLAB, 2nd Edition      %
% Wendy L. and Angel R. Martinez                                  %
% In-class Excercise September 18, 2014                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c = 2.1;    % constant
n=1000;  % generate 100 rv's
% set up the arrays to store variates
x = zeros(1,n); % random variates
xy = zeros(1,n); % corresponding y values
rej = zeros(1,n); % rejected variates
rejy = zeros(1,n); % corresponding y values
irv=1;
irej=1;
while irv <= n
   y = rand(1);  % random number from g(y)
   u = rand(1);  % random number for comparison
   if u <= (20*y*(1-y)^3)/c;
      x(irv)=y;
      xy(irv) = u*c;
      irv=irv+1;
   else
      rej(irej)= y;
      rejy(irej) = u*c; % really comparing u*c<=2*y
      irej = irej + 1;
   end
end
figure(1)
plot(x,xy,'o',rej,rejy,'*')
axis ([0 1 0 c])
title('Accepted/Rejected')

figure(2)
[fr,x]=hist(x);
h=x(2)-x(1);
bar(x,fr/(n*h),1)
title('Frequency Histogram of Random Sample')
```

Figure 1: Code of the in-class Accept/Reject random sample exercise.

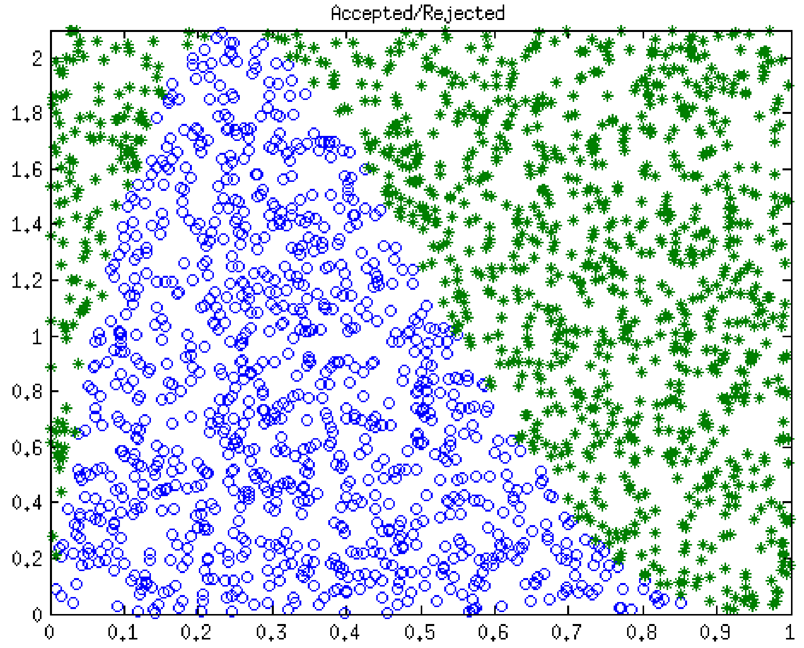


Figure 2: Plot of the in-class Accept/Reject random sample exercise.

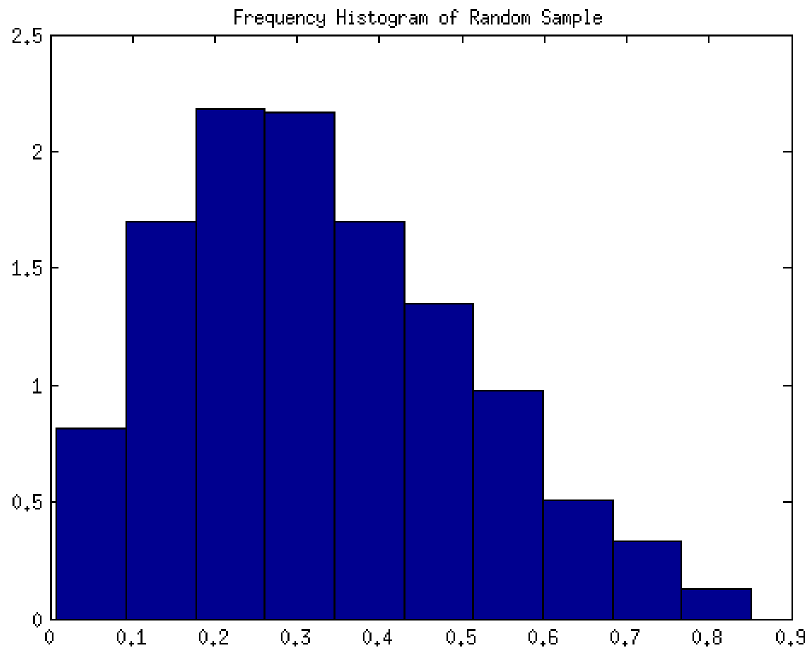


Figure 3: Plot of the in-class Accept/Reject random sample exercise.

Homework 3

Exercise 4.6

For this function I defined two input variables: v, which is the desired degrees of freedom for the chi-squared distribution and n, which is the desired sample size of the chi-square random variable. The simulation below is for $v = 26$ and $n = 1000$.

Editor - /home/jcapitz/Documents/Stat572/homework/hw3/chigen.m

chigen.m | question3pt16.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Exercise 4.6: write a function to generate chi-square r.v from    %
% standard normal r.v.                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [X] = chigen(v,n)
X = zeros(1,n);
% generate the normal rv
for i = 1:n
    Z = randn(1,v);
    X(i) = sum(Z.^2);
end
hist(X);
title(['Chi-square Random Variable with ' num2str(v) ' Degrees of Freedom'])
```

Command Window

```
>> X = chigen(26,1000);
>>
```

Figure 4: Code for function chigen(). It generates Chi-square sample from standard normal sample.

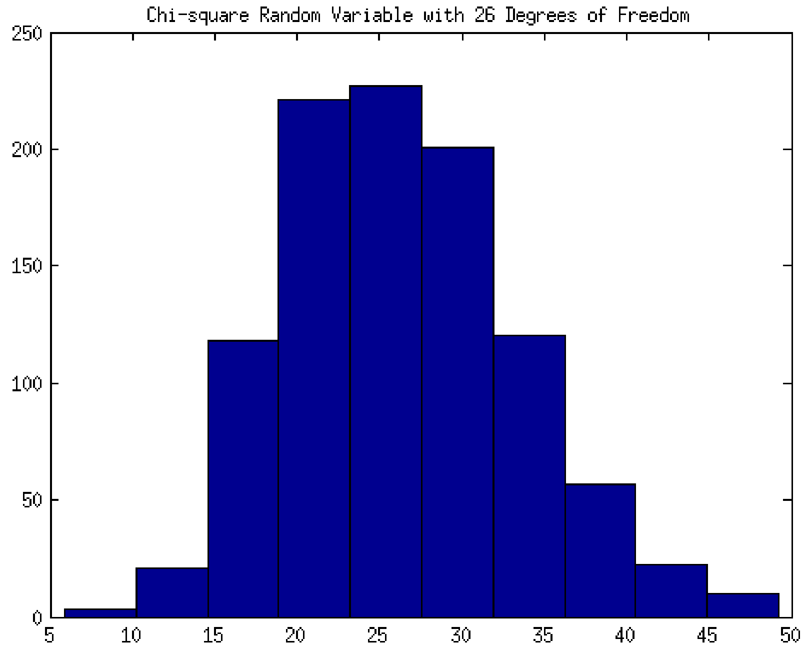
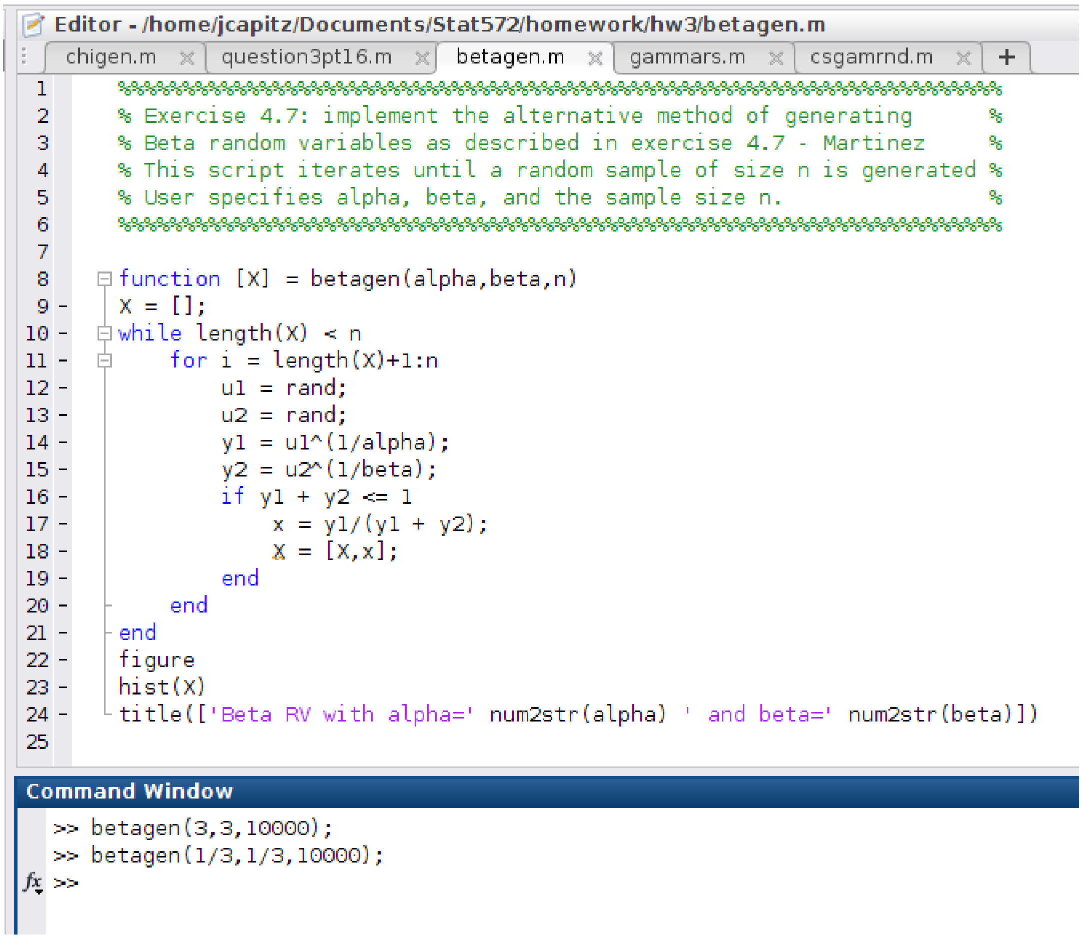


Figure 5: Chi-square random sample histogram, with $n = 1000$ and 26 degrees of freedom.

Exercise 4.7

This is an implementation of the algorithm described in exercise 4.7. Figure 6 shows the function I developed to implement the algorithm and generate random samples from the Beta distribution. The user needs to specify the sample size n, the parameters α and β. As an example I ran the algorithm with $n = 10,000$ and $\alpha = 3$, $\beta = 3$, see figure 7. I also ran it with $\alpha = 0.333$, $\beta = 0.333$, see figure 8. Both histograms show that the random sample comes from a Beta distribution with the given parameters.



Editor - /home/jcapitz/Documents/Stat572/homework/hw3/betagen.m

chigen.m | question3pt16.m | betagen.m | gammars.m | csgamrnd.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Exercise 4.7: implement the alternative method of generating      %
% Beta random variables as described in exercise 4.7 - Martinez     %
% This script iterates until a random sample of size n is generated %
% User specifies alpha, beta, and the sample size n.                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [X] = betagen(alpha,beta,n)
X = [];
while length(X) < n
    for i = length(X)+1:n
        u1 = rand;
        u2 = rand;
        y1 = u1^(1/alpha);
        y2 = u2^(1/beta);
        if y1 + y2 <= 1
            x = y1/(y1 + y2);
            X = [X,x];
        end
    end
end
figure
hist(X)
title(['Beta RV with alpha=' num2str(alpha) ' and beta=' num2str(beta)])
```

Command Window

```
>> betagen(3,3,10000);
>> betagen(1/3,1/3,10000);
>>
```

Figure 6:

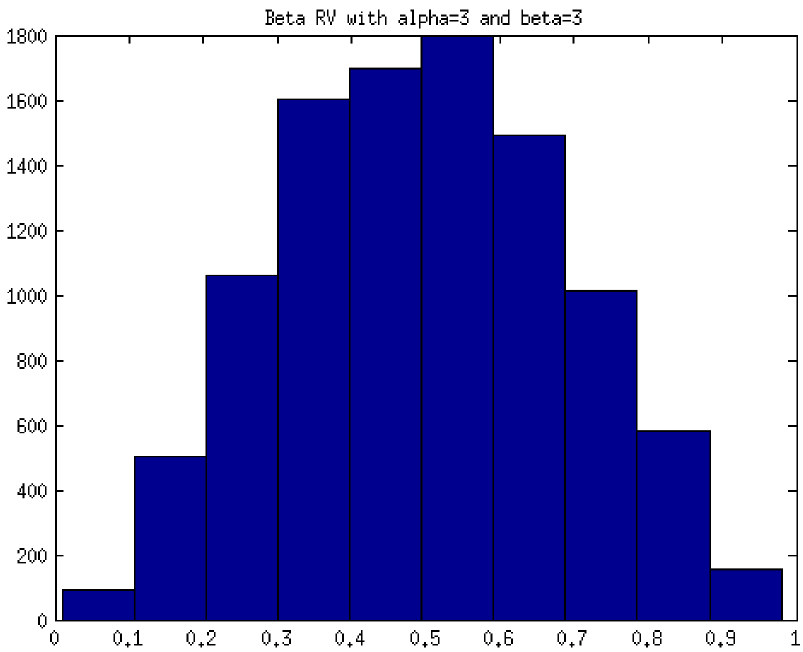


Figure 7: Beta random sample, $\alpha = 3$, $\beta = 3$.

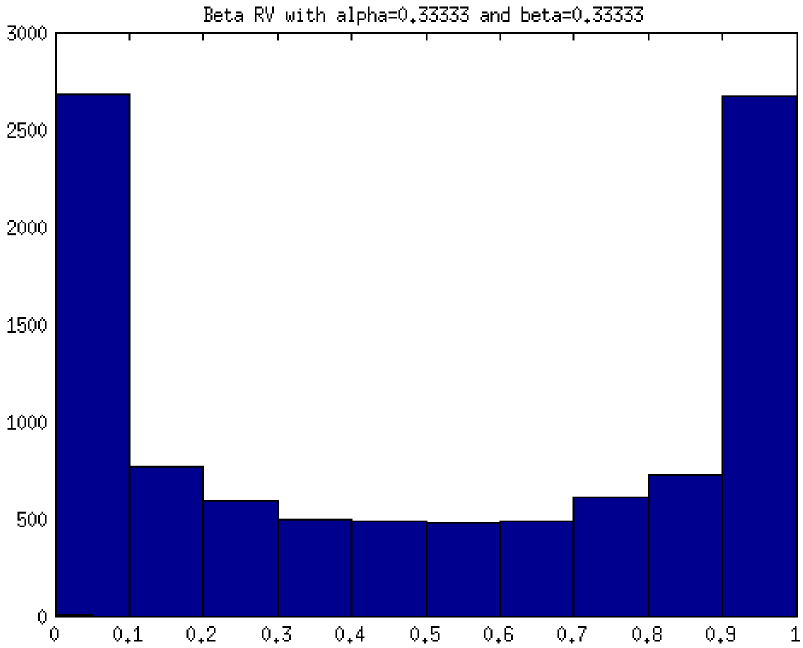


Figure 8: Beta random sample, $\alpha = 0.333$, $\beta = 0.333$.

Exercise 4.8

Rejected is the number of elements in the MATLAB vector rej. The number is $1,037$. As a percentage, the number of rejected is $\dfrac{1,037}{2,037} = 0.5091$, or 50.91%. The probability that any value is accepted is given by $\sum_j P(j \text{ is accepted and } y = j) = \sum_j \dfrac{p_j}{c} = \dfrac{1}{c}$. Therefore, on average we need to generate c values for y before one is accepted. In this example $c = 2$ and we observe an acceptance rate of about 50%, which is what would expect based on the theoretical probability of $\dfrac{1}{c}$. Figures 9 and 10 show code implementation and graphical results.

Editor - /home/jcapitz/Documents/Stat572/homework/hw3/question4pt8.m

question3pt16.m | betagen.m | gammars.m | csgamrnd.m | question4pt8.m

```
% Example 4.4
% Computational Statistics Handbook with MATLAB, 2nd Edition
% Wendy L. and Angel R. Martinez

c = 2;    % constant
n=1000;  % generate 1000 rv's
% set up the arrays to store variates
x = zeros(1,n); % random variates
xy = zeros(1,n); % corresponding y values
rej = zeros(1,n); % rejected variates
rejy = zeros(1,n); % corresponding y values
irv=1;
irej=1;
while irv <= n
    y = rand(1);   % random number from g(y)
    u = rand(1);   % random number for comparison
    if u <= 2*y/c;
        x(irv)=y;
        xy(irv) = u*c;
        irv=irv+1;
    else
        rej(irej)= y;
        rejy(irej) = u*c; % really comparing u*c<=2*y
        irej = irej + 1;
    end
end
figure(1)
plot(x,xy,'o',rej,rejy,'*')
axis ([0 1 0 c])
title('Accepted/Rejected')
```

Command Window

```
>> question4pt8
>> length(rej)

ans =

        1037

>>
```

Figure 9:

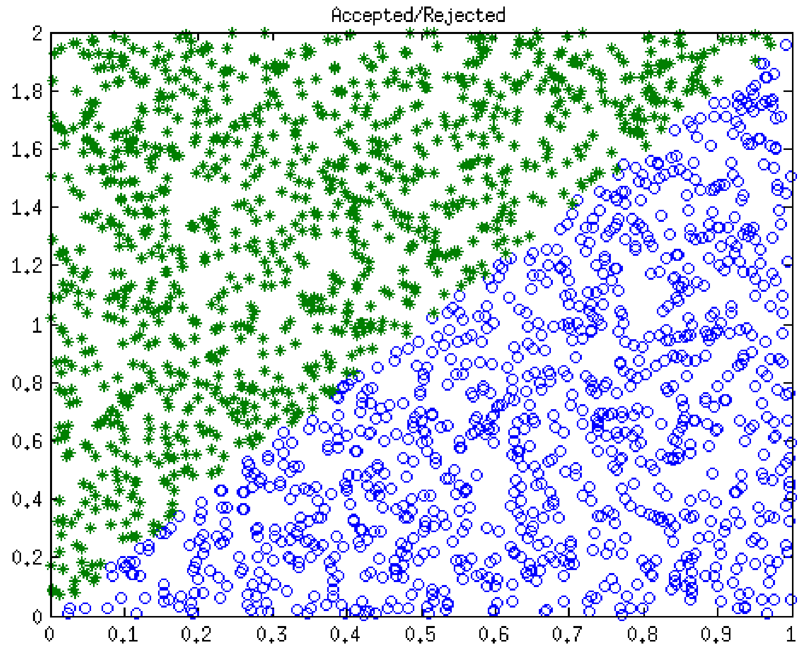

Accepted/Rejected
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Figure 10:

Exercise 4.9

In this exercise we generate a random sample of size $n = 100$ from the probability mass function:

$$p(y) = \begin{cases} 0.15, & \text{if } Y = 1 \\ 0.22 & \text{if } Y = 2 \\ 0.33 & \text{if } Y = 3 \\ 0.10 & \text{if } Y = 4 \\ 0.20 & \text{if } Y = 5 \end{cases}$$

Figure 11 shows the empirical results f and the theoretical values p. We see that the distribution of the random closely matches said pmf. I also calculated the number of rejected vs accepted. Generated 191 variables and rejected 91, for a rejection ratio of 47.64%, which makes sense because $c = 1.65$, so we expect to reject less than half. Figures 12 and 13 show the graphical results.

Editor - /home/jcapitz/Documents/Stat572/homework/hw3/question4pt9.m

question4pt8.m | question4pt9.m | betagen.m | +

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Exercise 4.9: implement the example 4.5 in Martinez                  %
% This is a discrete implementation of the Accept/Reject method        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; n=100; p=[0.15,0.22,0.33,0.10,0.20]; q=ones(1,5)./5;
X=zeros(1,n); R=zeros(1,n); counta=0; countr=0; c=1.65;  f=zeros(1,5);
for i=1:n
    while X(i) == 0
        Y=unidrnd(5);
        U=rand;
        if U <= p(Y)/(c*q(Y))
            X(i)=Y;
            counta=counta+1;
        else
           R(i)=Y;
           countr=countr+1;
        end
    end
end

for i=1:5
    f(i)=sum(X(:)==i)/n;
end

bar(f,1)
title('Relative Frequency Distribution')
ylabel('% frequency')
xlabel('x variate')
```

Command Window

```
>> f

f =

    0.1300    0.2400    0.3500    0.0900    0.1900

>> p

p =

    0.1500    0.2200    0.3300    0.1000    0.2000
```

Figure 11:

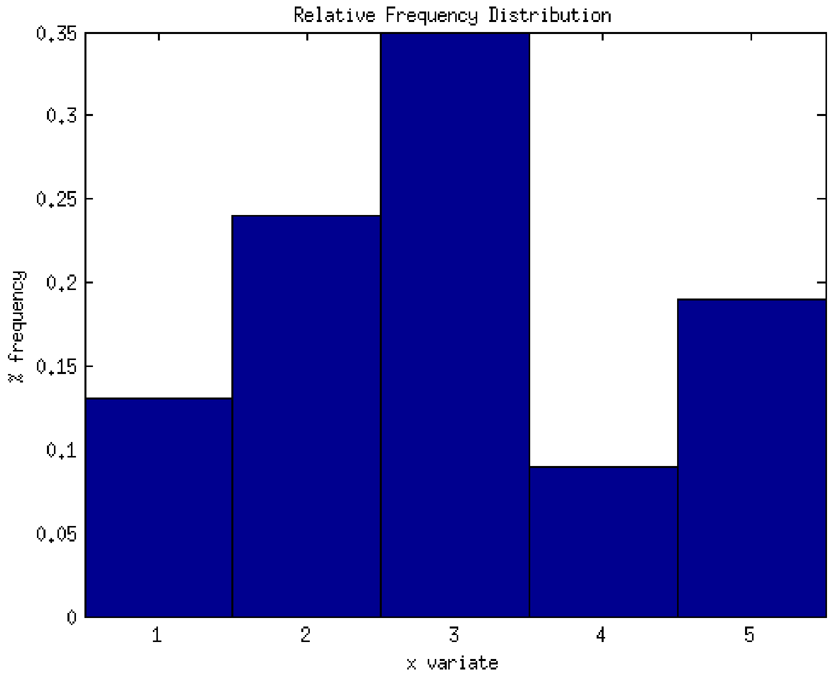


Figure 12: Empirical results of generating a discrete random sample from $p(y)$.

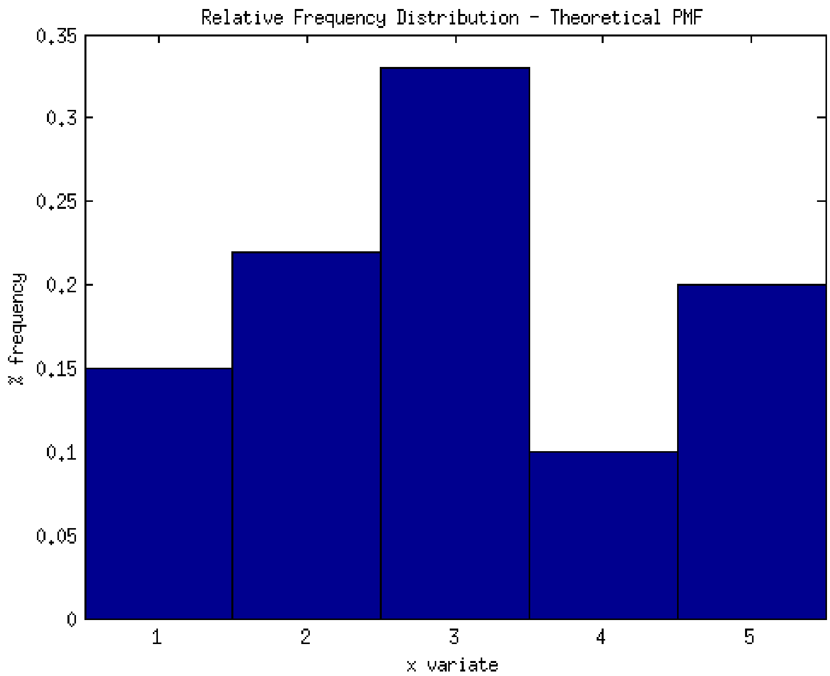


Figure 13: Theoretical frequency of $p(y)$