EE 511 Simulation Methods for Stochastic Systems
Project #2: Rejection and Independence
Chaitra Suresh (USC ID:7434709345)

[Double Rejection]

Goal: Generating X ~ f(x) where X is a random variable of bimodal distribution made up of an equally weighted, convex summation of beta and triangular distribution.

$$f(x) = \begin{cases} 0.5 * Beta(8,5), & 0 < x \leq 1 \\ 0.5 * (x-4), & 4 < x \leq 5 \\ -0.5 * (x-6), & 5 < x \leq 6 \\ 0, & else \end{cases}$$

Given: g(x) – a proposed probability density function(pdf)

Algorithm/ Routine:

Assumptions: (1) f(x) <= c*g(x), for all x

(2) Cumulative distributive function (CDF) of f(x) is flat, there is no unique mapping and thus, inverse CDF method cannot be applied.

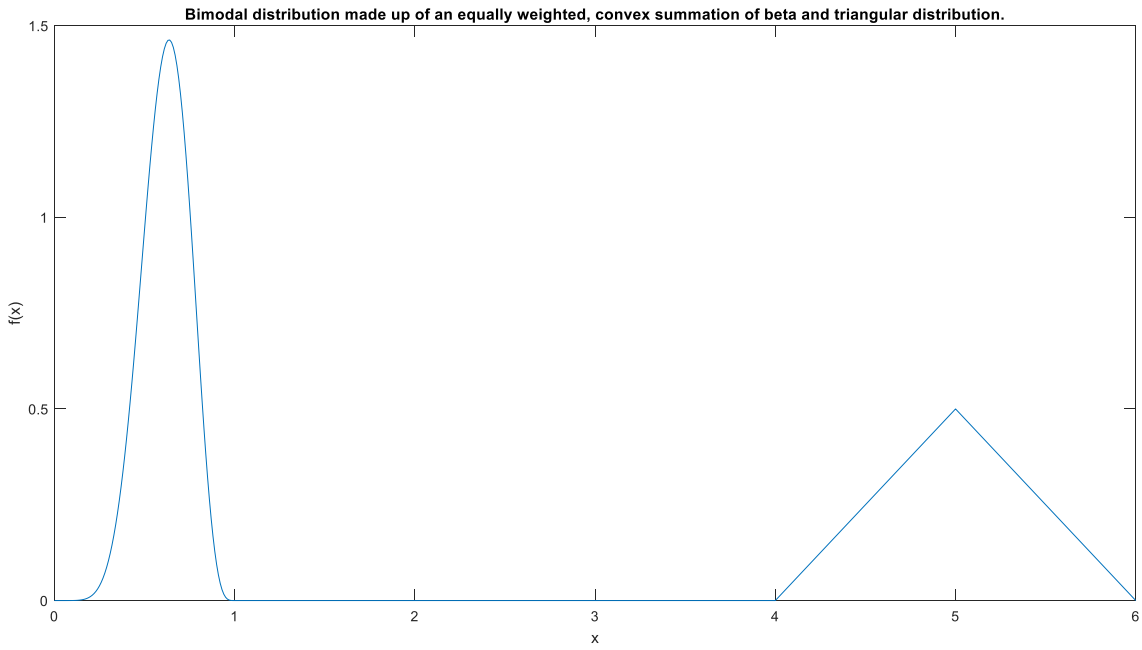
Step1: Pick a uniform random number between a and b (here a=0, b=6) V ~ g(x)=U([a b])

Step 2: Generate a uniform random number between 0 and max: U ~ U([0 max]) : max= c*g(x) , where rejection rate depends on the choice of c.

Step 3: X ← V if U <= $f_x(V)$

The rejection rate is the average number of rejected candidates per sample. This is a measure of the efficiency of Random Number Generator.

Results:



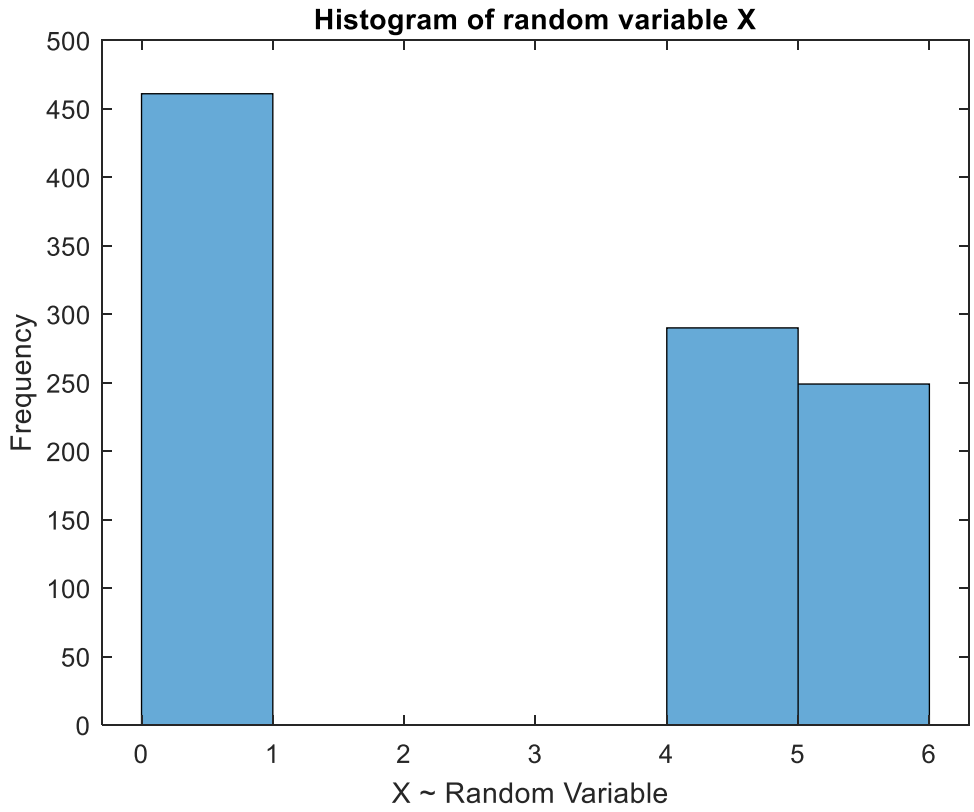


Table 1. Comparison of Rejection rate for different choice of c

max(g(x))=(1/6), max(f(x))= 1.4626

max(c*g(x))	Rejection Rate
1.5	0.8852
1.7	0.9043
2	0.9174

Discussion:

1. The plot of f(x) demonstrates the flat region (f(x)=0) between 1<x<=4. Cumulative Distributive function (CDF) of f(x) is flat for a small region, thereby inverse CDF method cannot be applied to sample the random variable, as it does not have a unique mapping. Hence, Rejection method is adopted.
2. From the histogram of random variable X, it is evident that more samples are obtained between the region 0<x<=1 Since there are less rejections due to very small difference between the c*g(x) envelope and f(x).
3. From Table1, it can be seen that as c*g(x) value increases, since g(x) is constant =1/6 , as c increases the rejection rate as well increases.
4. The choice of c plays a vital role in determining the rejection rate. If c=max(f(x)), then the rejection rate is optimal.
5. Since there are two distributions (beta and triangular), instead of using one g(x) spanned over entire region, if separate rejection method is applied to both distributions (defined over respective regions), then it results in lower rejection rate.

[Independence: Internally and Externally]

Goal: 1. Use the covariance statistic to test the independence between X_k and the lagged version X_{k+5}
2. Use 2-way contingency tables to argue for the independence of X~Beta(8,5) and Y~Beta(4,7)

Algorithm/ Routine:

Part1:

1. X_k are the samples obtained from the bimodal distribution above.
2. If X_k and X_{k+5} are independent, then the off diagonal co-efficient of covariance matrix are equal to 0.
   or Covariance is evaluated as $cov(X,Y)=E((X-\mu_x)(Y-\mu_y))$ where $X=X_k$ and $Y=X_{k+5}$ and X are samples obtained from the rejection method.
3. If X_k and X_{k+5} are independed then $cov(X_k,X_{k+5})=E((X-\mu_{xk})(X_{k+5}-\mu_{xk+5}))=0$

(X_{k+5} is the lagged version of X_k)

Part2:

1. Contingency Table are built to test the independence between random variable X and Y

fxy	j=1	j=2	j=3	
i=1				n1.
i=2				n2.
i=3				n3.
	n.1	n.2	n.3	

f_y approximately

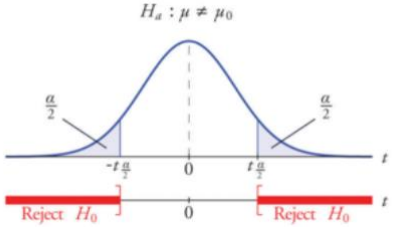
f_x approximately

where the table is filled with samples

2. $B_{i,j}$ refers to the $(i,j)^{th}$ bin
   $O_{i,j}$ = Number of samples ϵ $B_{i,j}$
   $E_{i,j}$ = Expected number of samples in $B_{i,j}$
   $P_{i.}=n_{i.}/n$ where $n_{i.}$ is the sum of sample is i^{th} row and all columns and n is the total number of samples
   $P_{.j}=n_{.j}/n$ where $n_{.j}$ is the sum of sample is j^{th} column and all rows and n is the total number of samples
   $E_{i,j} = n^* P_{i.} * P_{.j}$

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(Oij-Eij)^2}{Eij}$$

3. Hypothesis Testing



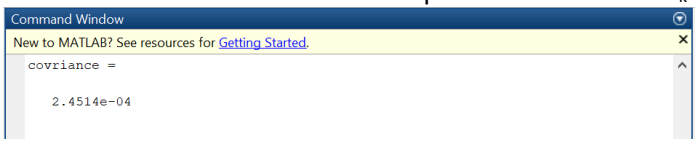
Null Hypothesis (H_0): X and Y are independent if:
$\chi 2 \sim \chi_\alpha$ ((r-1)(c-1)) r and c indicate the total row and column of the contingency table, (r-1)(c-1) is the degree of the chi-square distribution and α – significance level

Alternate Hypothesis(H_{alt}): if the above stated approximation fails, X and Y related

Result:

Part1

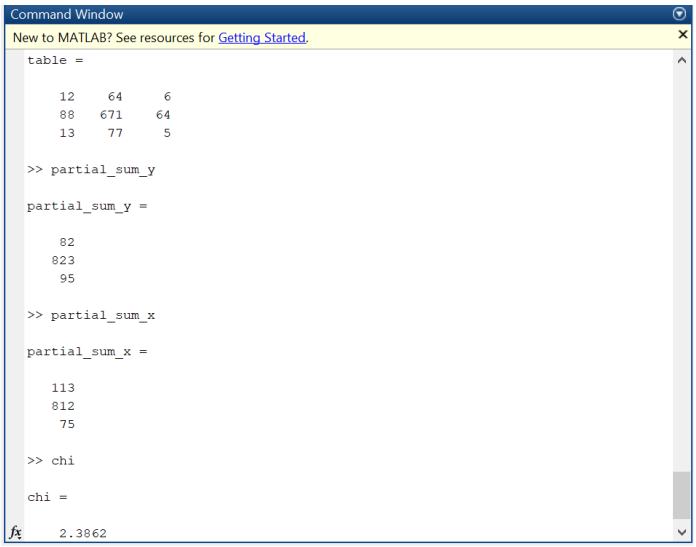
Covariance statistic to test the independence between X_k and the lagged version X_{k+5}



Covariance is approximately equal to 0, thus X_k and X_{k+5} are independent

Part2

2-way contingency tables to argue for the independence of X~Beta(8,5) and Y~Beta(4,7)



Discussion:

1. Covariance= 2.4514e-04 which is approximately equal to 0, thus X_k and X_{k+5} are independent.
2. 3x3 Contingency table is created with r=3 and c=3, degree of freedom =(r-1)(c-1)=4
   Result indicate the table, fy and fx approximately
3. The obtained $\chi 2$ value is 2.3862.
   At $\alpha = 0.05$ $\chi_{0.05}(4) = 9.488$ from the below table
4. Since $\chi 2 = 2.3862 << X_{0.05}(4)$ implying (p-value > α). Do not reject the null Hypothesis.

Table of the chi square distribution – Appendix J, p. 915

	Level of Significance α								
df	0.200	0.100	0.075	0.050	0.025	0.010	0.005	0.001	0.0005
1	1.642	2.706	3.170	3.841	5.024	6.635	7.879	10.828	12.116
2	3.219	4.605	5.181	5.991	7.378	9.210	10.597	13.816	15.202
3	4.642	6.251	6.905	7.815	9.348	11.345	12.838	16.266	17.731
4	5.989	7.779	8.496	9.488	11.143	13.277	14.860	18.467	19.998

5. Since the null hypothesis is accepted, X and Y are independent

[Network Fit]

Goal: 1. To generate and plot three network samples for each value of p=0.03 and p=0.12.
2. To generate a network with (n, p) = (100, 0.06). Count the degree of each node in the network and plot the histogram of degrees. Use goodness-of-fit tests to check how well the network degree distribution fits a binomial (n=100, p=0.06) or a poisson (λ=np=6).

Given: n people in a social network and any given unordered pair of two people are connected at random and independently with probability p

Algorithm/ Routine:
Part1:

1. For given n, p: the total number of edges is ${}^{n}C_{2}$ (n choose 2) in a graph given that any unordered pair of two people are connected at random and independently.
2. for i=1:n and for j=i+1:n a uniform random number is generated [0,1], if the value is ≤ p, an edge is created between node i and j else there is no edge between i and j nodes. (node1 cannot be connected to node1 , therefore the second for loop starts from i+1)
3. Obtained edge matrix is upper triangular. Since if node1 is connected to node2, it implies that node2 is as well connected to node1. Final incidence matrix = Edge matrix + transpose of Edge matrix.
4. Graph is plotted with n nodes and by parsing the incidence matrix as edges.

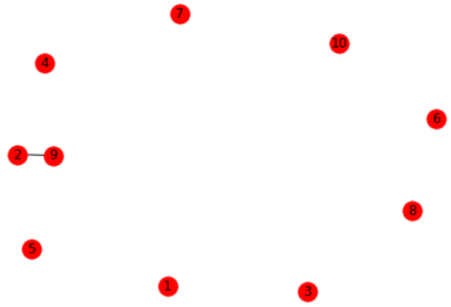
Part2:

1. The above procedure of Part1 is performed.
2. From the incidence matrix, the observed degree of a node can be calculated by adding the corresponding rows of final incidence matrix.
3. For expected binomial network degree distribution, n random binomial distribution with n=100, p=6 is generated
4. For expected poisson network degree distribution, n random poisson distribution with λ=np=6 is generated
5. To evaluate the goodness of fit test, contingency table is created of equal bin count for observed and expected network degree distribution.
6. For bin-count=10 the chi square values for binomial and poisson fit are calculated as
   $B_{i,j}$ refers to the $(i,j)^{th}$ bin
   $O_{i,j}$ = Number of samples ϵ $B_{i,j}$
   $E_{i,j}$ = Expected number of samples in $B_{i,j}$
   $P_{i.}=n_{i.}/n$ where $n_{i.}$ is the sum of sample is i^{th} row and all columns and n is the total number of samples
   $P_{.j}=n_{.j}/n$ where $n_{.j}$ is the sum of sample is j^{th} column and all rows and n is the total number of samples
   $E_{i,j} = n^* P_{i.} * P_{.j}$
   $\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(Oij-Eij)^2}{Eij}$
7. Conclusion from Hypothesis testing

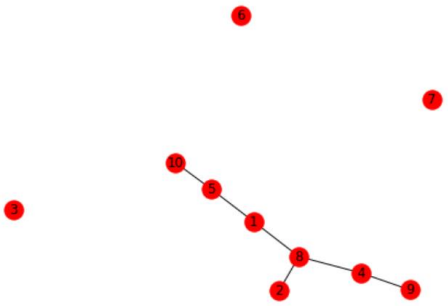
Result:

Part1

Case1: n=10

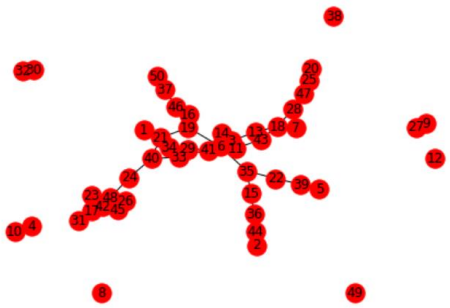


n=10 p= 0.03

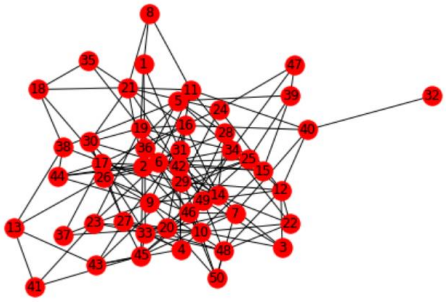


n=10 p= 0.12

Case2: n=50

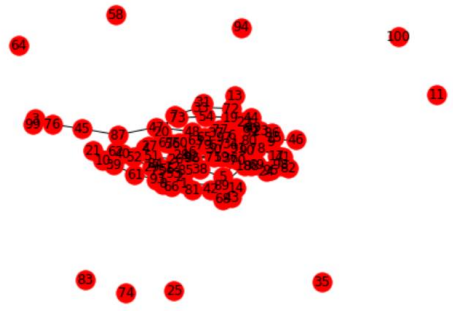


n=50 p= 0.03

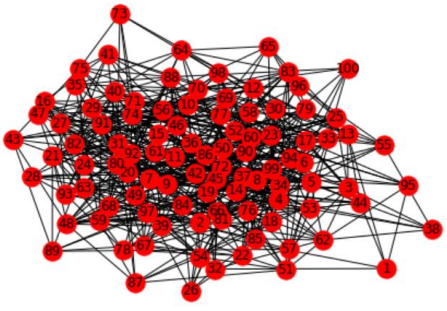


n=50 p= 0.12

Case3: n=100

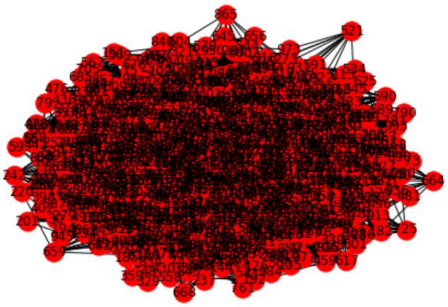


n=100 p= 0.03

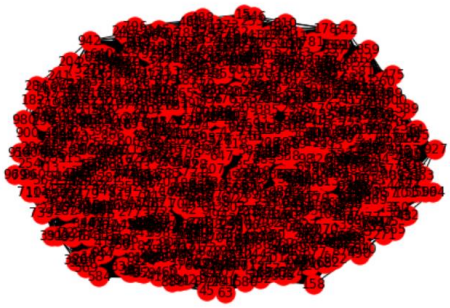


n=100 p= 0.12

Case4: n=1000



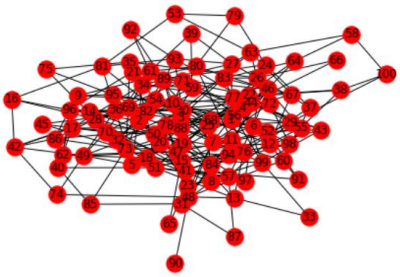
n=1000 p= 0.03



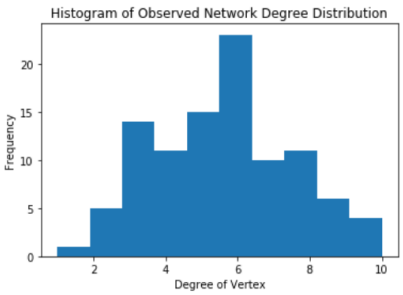
n=1000 p= 0.12

Part2

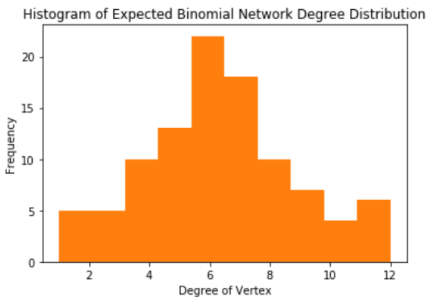
Network graph for n=100,p=0.06



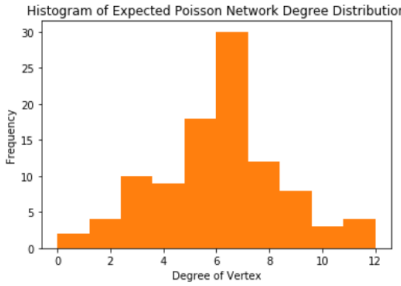
Histogram for Observed Network Degree Distribution



Histogram for Expected Binomial Network Degree Distribution



Histogram for Expected Poisson Network Degree Distribution



Chi-Square Goodness of fit test for binomial and poisson

```
In [219]:
chi=0;
for i in range (0,10):
    chi=chi+(m.pow((observed_bin_hist[i]-expected_bin_hist[i]),2)/expected_bin_hist[i]);

print(chi)

chi=0;
for i in range (0,10):
    chi=chi+(m.pow((observed_bin_hist[i]-expected_poisson_hist[i]),2)/expected_poisson_hist[i]);

print(chi)
```

```
12.676234876234878
9.38611111111111
```

Discussion:

1. In a network with n people, p=0.12 has better/ more connectivity compared to p=0.03
2. As p value increases, the connectivity is better or a greater number of nodes are connected
3. The structure of the graph depends on the number of nodes connected, which indirectly depends on the probability of creating an edge
4. As the number of nodes increases, the graph nodes are oddly placed to suffice the connection
5. The obtained χ2 value for binomial is 12.6762 and the obtained χ2 value for poisson is 9.3861.
6. At α = 0.05 $\chi_{0.05}$ (9) = 16.919 from the below table (number of columns/bin count=10, degree of freedom=(c-1)=9)

Table of the chi square distribution – Appendix J, p. 915

	Level of Significance α								
df	0.200	0.100	0.075	0.050	0.025	0.010	0.005	0.001	0.0005
1	1.642	2.706	3.170	3.841	5.024	6.635	7.879	10.828	12.116
2	3.219	4.605	5.181	5.991	7.378	9.210	10.597	13.816	15.202
3	4.642	6.251	6.905	7.815	9.348	11.345	12.838	16.266	17.731
4	5.989	7.779	8.496	9.488	11.143	13.277	14.860	18.467	19.998
5	7.289	9.236	10.008	11.070	12.833	15.086	16.750	20.516	22.106
6	8.558	10.645	11.466	12.592	14.449	16.812	18.548	22.458	24.104
7	9.803	12.017	12.883	14.067	16.013	18.475	20.278	24.322	26.019
8	11.030	13.362	14.270	15.507	17.535	20.090	21.955	26.125	27.869
9	12.242	14.684	15.631	16.919	19.023	21.666	23.589	27.878	29.667

7. Since χ2 < < $X_{0.05}$ (9) implying (p-value > α). Both binomial and poisson fit well for network degree distribution
8. Since $\chi^2_{\text{poisson}} < \chi^2_{\text{binomial}}$, poisson distribution (λ=np=6) fits better than binomial distribution (n=100, p=0.06)
9. The above conclusion cannot be generalized, since the expected binomial/poisson network degree distribution are Random every time. But since n>>1 and p<<1, poisson fits better.

Code:

[Double Rejection] - MATLAB

```
% %for plotting f(x)
% i=1;
% for x=0:0.01:6
%     if(0<x && x<=1)
%        p(i)=0.5*betapdf(x,8,5);
%     elseif(4<x && x<=5)
%         p(i)=0.5*(x-4);
%     elseif(5<x && x<=6)
%         p(i)=-0.5*(x-6);
%     else
%         p(i)=0;
%     end
%     i=i+1;
% end
%figure(1);
%plot(0:0.01:6,p);
%title('Bimodal distribution made up of an equally weighted, convex summation
%of beta and triangular distribution.');
%xlabel('x');
%ylabel('f(x)');

accepted_sample=0;
rejection=1;
sample=1;
while(accepted_sample<1000)
    r=6*rand;      % Generating V~g(x) ~ U([0 6])
    u=1.5*rand;    % Generating U ~ U([0 max(c*g(x))])
    if(0<r && r<=1)       % Calculating f(V)
       f=0.5*betapdf(r,8,5);
    elseif(4<r && r<=5)
        f=0.5*(r-4);
    elseif(5<r && r<=6)
        f=-0.5*(r-6);
    else
        f=0;
    end
    if u<=f          % Condition for accepting a sample
        x(accepted_sample+1)=r;
        y(accepted_sample+1)=f;
        accepted_sample=accepted_sample+1;
    else
        rejection=rejection+1;  %Rejection count
    end
    sample=sample+1;    % Total sample count
end
figure(2);
histogram(x);
title('Histogram of random variable X');
xlabel('X ~ Random Variable ');
ylabel('Frequency');
Rejection_rate = rejection/sample
```

[Independence: Internally and Externally] - MATLAB

```
% Independence [Internally and Externally]

Yk=zeros(1,1000);
Yk(1,1:995)=y(1,6:1000);
C=cov(y,Yk);
covriance=C(1,2)

 % Contingency Table
X=betarnd(8,5,1,1000);
Y=betarnd(4,7,1,1000);
table=zeros(3,3);
for i=1:1000                % Creation of contingency table
   if(min(X)<=X(i) && X(i)<=mean([min(X) mean(X)]))
       if(min(Y)<=Y(i) && Y(i)<=mean([min(Y) mean(Y)]))
       table(1,1)=table(1,1)+1;
       elseif(mean([max(Y) mean(Y)])<=Y(i) && Y(i)<=max(Y))
        table(1,3)=table(1,3)+1;
       else
        table(1,2)=table(1,2)+1;
       end
   elseif(mean([max(X) mean(X)])<=X(i) && X(i)<=max(X))
       if(min(Y)<=Y(i) && Y(i)<=mean([min(Y) mean(Y)]))
       table(3,1)=table(3,1)+1;
       elseif(mean([max(Y) mean(Y)])<=Y(i) && Y(i)<=max(Y))
        table(3,3)=table(3,3)+1;
       else
        table(3,2)=table(3,2)+1;
       end
   else
       if(min(Y)<=Y(i) && Y(i)<=mean([min(Y) mean(Y)]))
       table(2,1)=table(2,1)+1;
       elseif(mean([max(Y) mean(Y)])<=Y(i) && Y(i)<=max(Y))
        table(2,3)=table(2,3)+1;
       else
        table(2,2)=table(2,2)+1;
       end
   end
end

partial_sum_y(1,1)=sum(table(1,:));
partial_sum_y(2,1)=sum(table(2,:));
partial_sum_y(3,1)=sum(table(3,:))
partial_sum_x(1,1)=sum(table(:,1));
partial_sum_x(2,1)=sum(table(:,2));
partial_sum_x(3,1)=sum(table(:,3))

n=1000;
chi=0;
% Calculation of chi-square value
for i=1:3
    for j=1:3
        e=((partial_sum_y(i,1)*partial_sum_x(j,1))/n);
        chi=chi+((table(i,j)-e).^2)/e;
    end
end
```

[Network Fit] – Python

Part1:

```
import networkx as nx
import numpy as np
import math as m
import matplotlib.pyplot as plt

n=10
p=0.06
Degree=list()
Edge=np.zeros(shape=(n,n))
edgelist=list()
for i in range(0,n):
  for j in range(i+1,n):
    rnd=np.random.uniform(0,1)
    if rnd>=p:
       Edge[i][j]=0
    else:
       Edge[i][j]=1
       e=(i+1,j+1)
       edgelist.append(e)


G =nx.Graph()
G.add_nodes_from(range(1,n+1))
G.add_edges_from(edgelist)
nx.draw(G,with_labels = True)
Edge1=Edge.transpose()
Final_Edge=Edge+Edge1
```

Part2:

```
import networkx as nx
import numpy as np
import math as m
import matplotlib.pyplot as plt

n=10
p=0.06
Degree=list()
Edge=np.zeros(shape=(n,n))
edgelist=list()
for i in range(0,n):
  for j in range(i+1,n):
    rnd=np.random.uniform(0,1)
    if rnd>=p:
       Edge[i][j]=0
    else:
       Edge[i][j]=1
       e=(i+1,j+1)
       edgelist.append(e)
```

```
G =nx.Graph()
G.add_nodes_from(range(1,n+1))
G.add_edges_from(edgelist)
nx.draw(G,with_labels = True)
Edge1=Edge.transpose()
Final_Edge=Edge+Edge1

for i in range(0,n):
   sum1=0
   for j in range(0,n):
    sum1=sum1+Final_Edge[i][j]
   Degree.append(sum1)

print(Degree)
Observed_degree=Degree
observed_bin_hist = plt.hist(Degree, bins=n)[0]

a=np.random.binomial(100,0.06,size=100)
print(a)

expected_bin_hist = plt.hist(a)[0]
print(expected_bin_hist)
plt.hist(a)
plt.xlabel("Degree of Vertex")
plt.ylabel("Frequency")
plt.title("Histogram of Expected Binomial Network Degree Distribution")

a=np.random.poisson(lam=n*p,size=100)
print(a)

expected_poisson_hist = plt.hist(a)[0]
print(expected_poisson_hist)
plt.hist(a)
plt.xlabel("Degree of Vertex")
plt.ylabel("Frequency")
plt.title("Histogram of Expected Poisson Network Degree Distribution")

chi=0;
for i in range (0,10):
   chi=chi+(m.pow((observed_bin_hist[i]-expected_bin_hist[i]),2)/expected_bin_hist[i]);

print(chi)

chi=0;
for i in range (0,10):
   chi=chi+(m.pow((observed_bin_hist[i]-expected_poisson_hist[i]),2)/expected_poisson_hist[i]);

print(chi)
```