DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

ISLAMIC UNIVERSITY, BANGLADESH

ইসলামী বিশ্ববিদ্যালয়

Statistics for Communication Engineering Laboratory

SUBMITTED TO:

Prof. Dr Zahidul Islam

Professor

Information & Communication Technology

Islamic University, Bangladesh

Sadia Afrin Kumu| Roll No.: **2018056**| Reg. No.: **1331**| **10.02.2024**|

Experiment No. – 1

Simulating a Normal Distribution

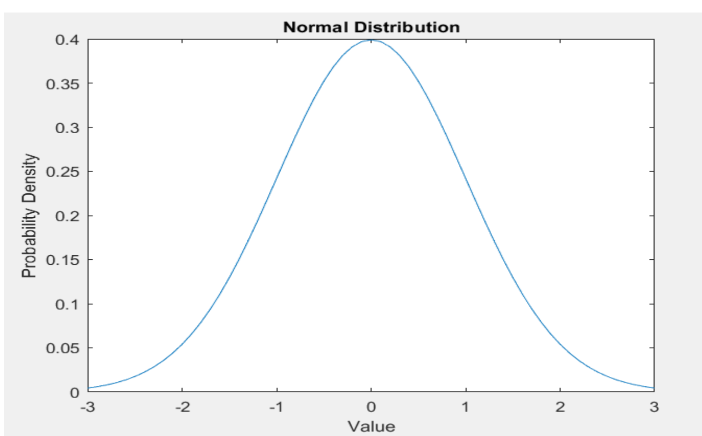
OBJECTIVE:

Generate and plot a normal distribution with specified mean and standard deviation.

INPUT:

```
%Parameters

mu=0;%Mean

sigma=1;%Standard Deviation

%Generate a normal distribution

x=-3*sigma:0.1:3*sigma;

y=normpdf(x,mu,sigma);

%Plot

figure;

plot(x,y);

title('Normal Distribution');

xlabel('Value');

ylabel('Probability Density');
```

OUTPUT:



Experiment No. – 2

Calculating Binomial Probabilities

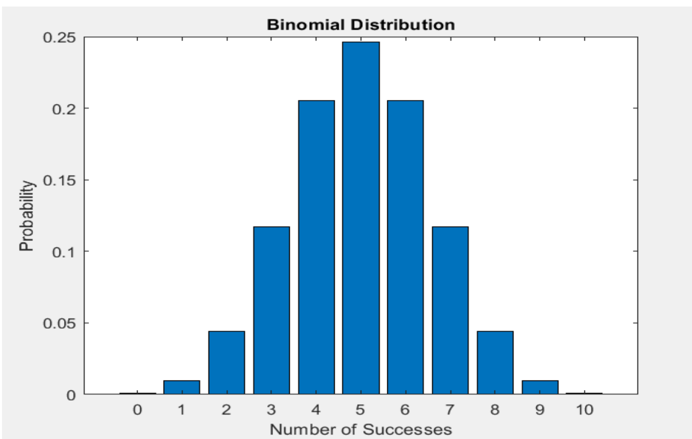
OBJECTIVE:

Calculate and plot the probabilities for a binomial distribution.

INPUT:

```
%Parameters
n=10;%Number of trials
p=0.5;%Probality of success
%Calculate binomial probabilities
x=0:n;
y=binopdf(x,n,p);
%Plot
figure;
bar(x,y);
title('Binomial Distribution');
xlabel('Number of Successes');
ylabel('Probability');
```

OUTPUT:



Experiment No. – 3

Poisson Distribution

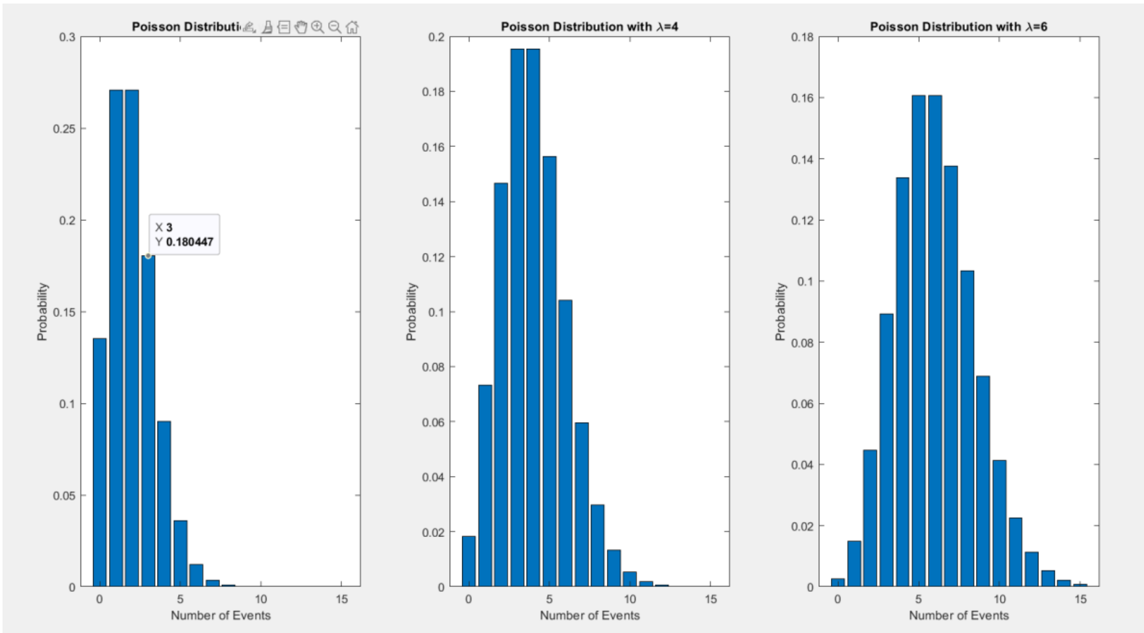
OBJECTIVE:

Plot the Poisson distribution for different mean values.

INPUT:

```
%Parameters
lambda=[2,4,6];%Different mean values
%Generate and plot Poissson distributions
figure;
for i=1:length(lambda)
    x=0:15;
    y=poisspdf(x,lambda(i));
    subplot(1,length(lambda),i);
    bar(x,y);
    title(['Poisson Distribution with \lambda=',num2str(lambda(i))]);
    xlabel('Number of Events');
    ylabel('Probability');
end
```

OUTPUT:



Experiment No. – 4

Exponential Distribution

OBJECTIVE:

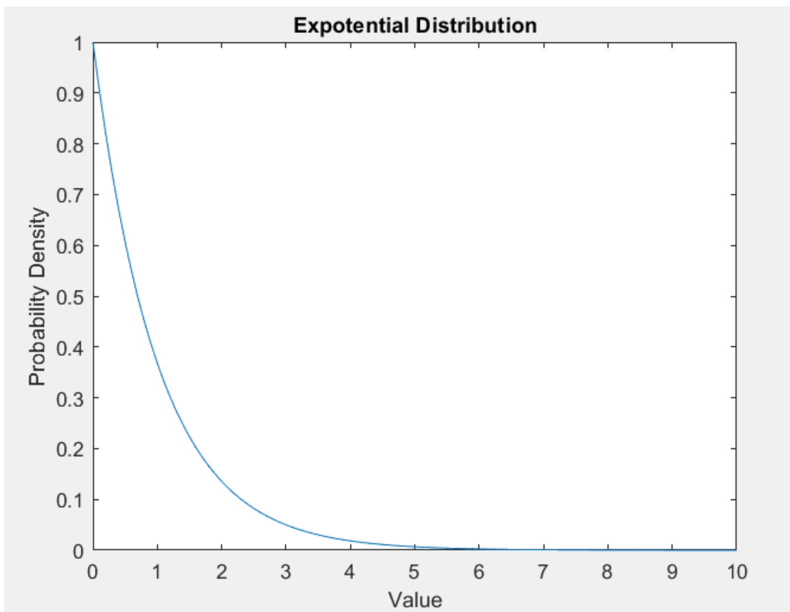
Generate an exponential distribution and plot its probability destiny function.

INPUT:

```
%Parameter

lambda=1;%Rate parameter

%Generate expotential distribution

x=0:0.1:10;

y=exppdf(x,1/lambda);

%Plot

figure;
```

```
plot(x,y);

title('Expotential Distribution');

xlabel('Value');

ylabel('Probability Density');
```

<u>OUTPUT</u>:



Experiment No. – 5

Uniform Distribution

OBJECTIVE:

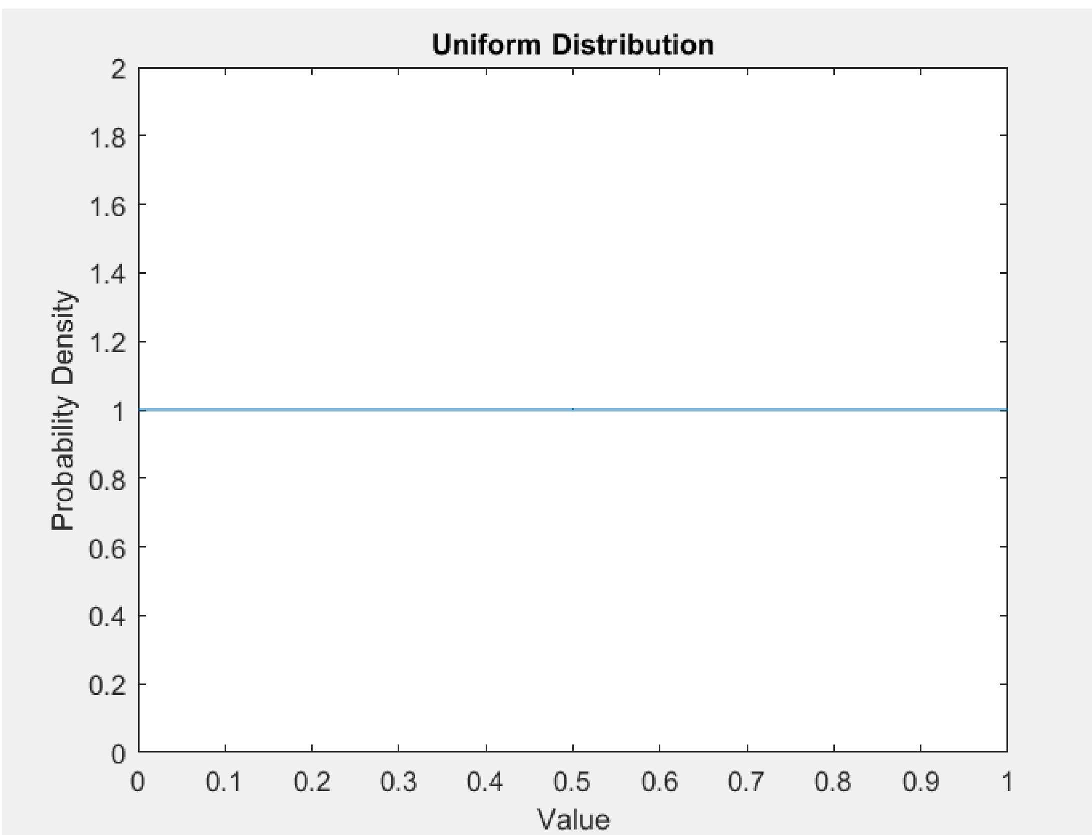
Simulate and visualize a uniform distribution.

<u>INPUT</u>:

```
%Parameters

a=0;%Lower bound

b=1;%Upper bound
```

```
%Generate uniform distribution
x=a:0.01:b;
y=unifpdf(x,a,b);
%plot
figure;
plot(x,y);
title('Uniform Distribution');
xlabel('Value');
ylabel('Probability Density');
```

<u>OUTPUT</u>:



Experiment No. – 6

Cumulative Distribution Function (CDF)

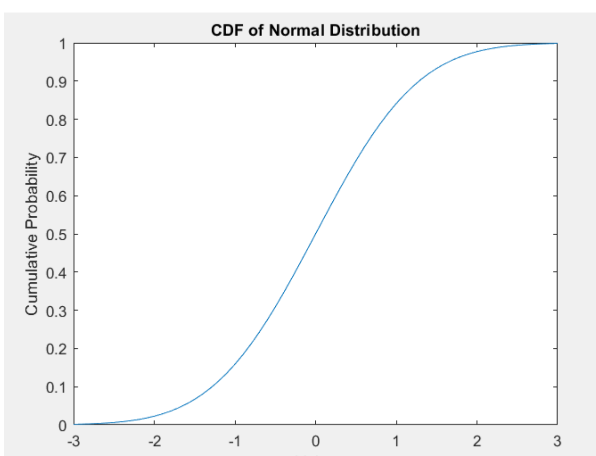
OBJECTIVE:

Plot the CDF of a normal distribution.

INPUT:

```
%Parameters

mu=0;%Mean

sigma=1;%Standard Deviation

%Generate CDF

x=-3*sigma:0.1:3*sigma;

y=normcdf(x,mu,sigma);

%Plot

figure;

plot(x,y);

title('CDF of Normal Distribution');

xlabel('Value');

ylabel('Cumulative Probability');
```

OUTPUT:



Experiment No. – 7

Working with Random Variables

OBJECTIVE:

Generate random numbers from a normal distribution and calculate their statistics.

INPUT:

```
%Generate random numbers
num_samples=1000;
samples=normrnd(0,1,[num_samples,1]);
%Calculate statistics
mean_val=mean(samples);
std_dev=std(samples);
%Display results
fprintf('Mean:%.2f\n',mean_val);
fprintf('Standard Deviation:%.2f\n',std_dev);
```

OUTPUT:

Mean:-0.03

Standard Deviation:1.00

Experiment No. – 8

Calculating Mean, Median and Mode

OBJECTIVE:

Find the mean, median and mode of a given data set.

INPUT:

```
%Calculating Mean,median an mode of a given data set.

%Data

data=[15,9,26,13,14,12,22,19];

%Mean

mean_val=mean(data);

%Median

median_val=median(data);

%Mode

mode_val=mode(data);

%Display results

fprintf('Mean:%.2f\n',mean_val);

fprintf('Median:%.2f\n',median_val);

fprintf('Mode:%.2f\n',mode_val);
```

OUTPUT:

Mean:16.25

Median:14.50

Mode:9.00

Experiment No. – 9

Standard Deviation and Variance

OBJECTIVE:

Compute the standard deviation and variance of a data set.

INPUT:

```
%Standard deviation and variance
%Data
data=[15,9,26,13,14,12,22,19];
%Standard Deviation
std_dev=std(data);
%Variance
variance=var(data);
%Display results
fprintf('Standard Deviation:%.2f\n',std_dev);
fprintf('variance:%.2f\n',variance);
```

OUTPUT:

Standard Deviation:5.65

variance:31.93

Experiment No. – 10

Linear Correlation Coefficient

OBJECTIVE:

Determine the linear correlation coefficient (Pearson's r) between two sets of data.

INPUT:

```
%Linear correlation Coefficient

%Data

data_x=[1,2,3,4,5];

data_y=[2,4,5,4,5];

%correlation Coefficient

corr_coeff=corrcoef(data_x,data_y);

%Display result

fprintf('Correlation Coefficient:%.2f\n',corr_coeff(1,2));
```

OUTPUT:

Correlation Coefficient:0.77

Experiment No. – 11

Histogram Plotting

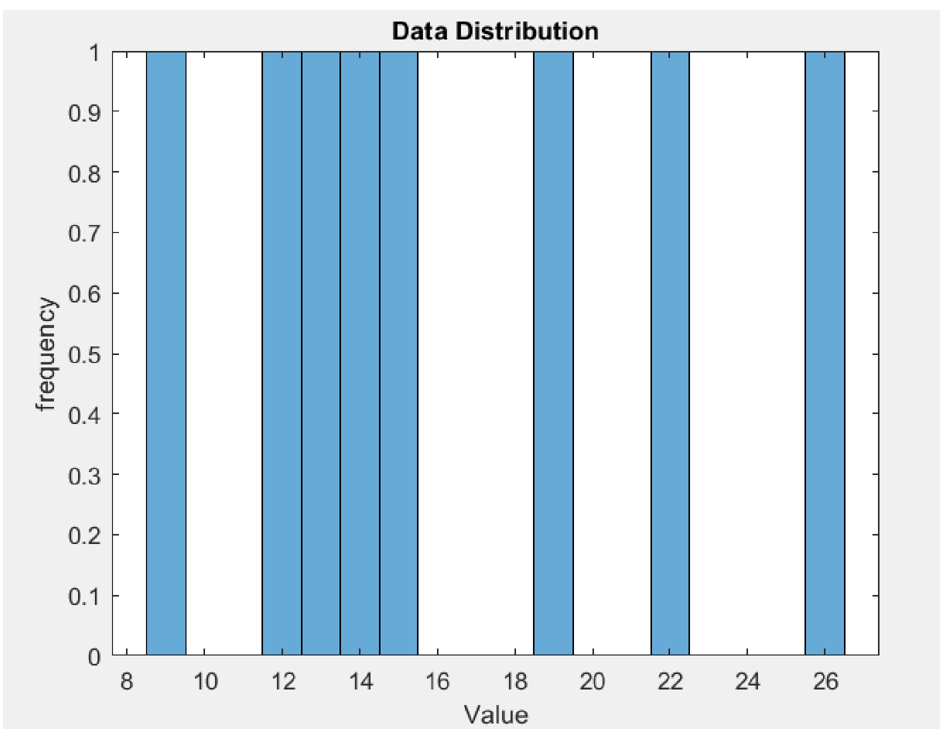
OBJECTIVE:

Create a Histogram to visualize the distribution of a data set.

INPUT:

```
%Histogram Plotting

%Data

data=[15,9,26,13,14,12,22,19];

%Plot

figure;

histogram(data);

title('Data Distribution');

xlabel('Value');

ylabel('frequency');
```

OUTPUT:



Experiment No. – 12

Boxplot for Data Distribution

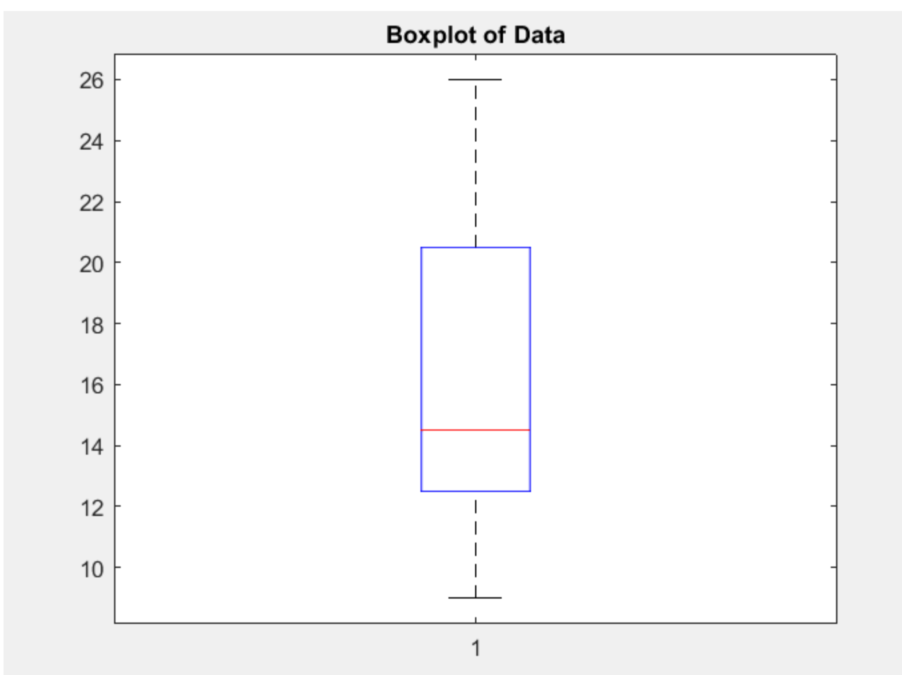
OBJECTIVE:

Generate a boxplot to observe the spread and skewness of data.

INPUT:

```
%Boxplot for Data Distribution

%Data

data=[15,9,26,13,14,12,22,19];

%Boxplot

figure;

boxplot(data);

title('Boxplot of Data');
```

OUTPUT:



Experiment No. – 13

Scatter Plot for Two Variables

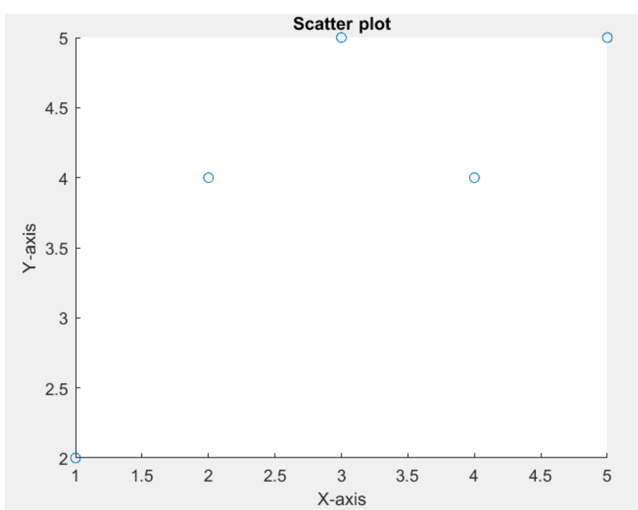
OBJECTIVE:

Create a scatter plot to visualize the relationship between two variables.

INPUT:

```
%Scatter Plot for Two Variables

%Data

data_x=[1,2,3,4,5];

data_y=[2,4,5,4,5];

%Scatter plot

figure;

scatter(data_x,data_y);

title('Scatter plot');

xlabel('X-axis');

ylabel('Y-axis');
```

OUTPUT:



Experiment No. – 14

Generating Random Data and Analyzing

OBJECTIVE:

Generate random data following a normal distribution and analyze it.

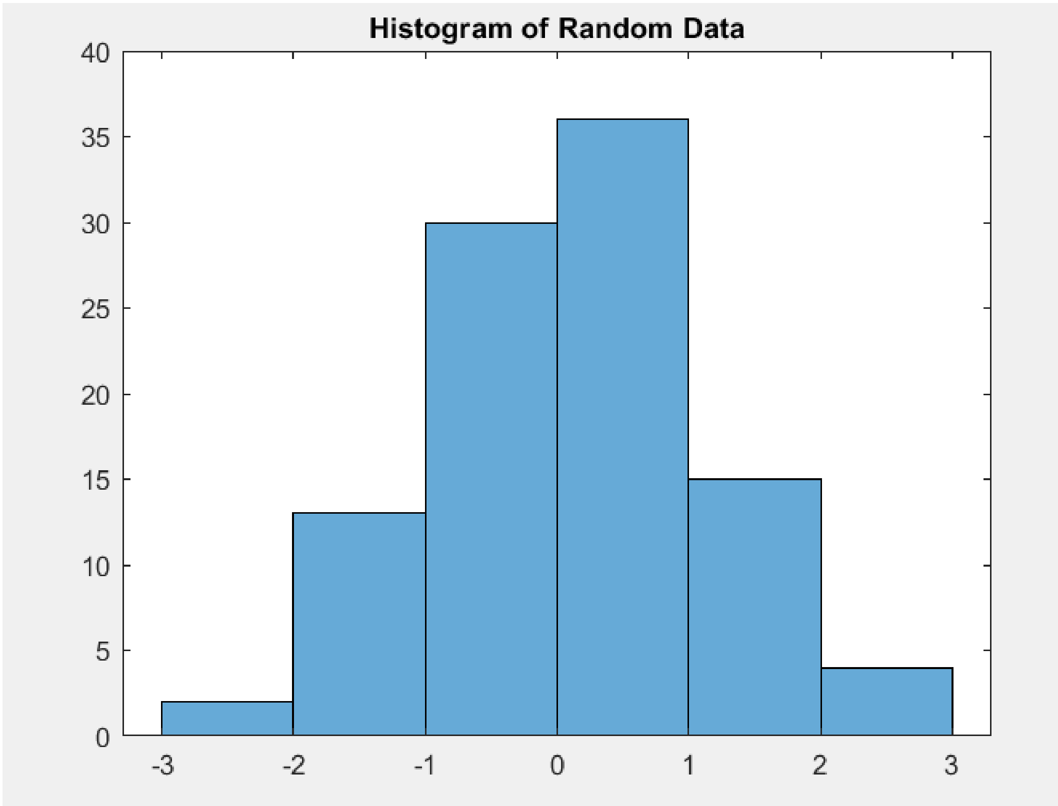
INPUT:

```
%Generating Random Data and Analyzing
%Generate random data
data=normrnd(0,1,[100,1]);%100 random numbers from N(0,1)
%Mean and Standard Deviation
mean_val=mean(data);
std_dev=std(data);
fprintf(' Random Data:%.2f\n',mean_val);
fprintf('Standard Deviation of Random Data:%.2f\n',std_dev);
%Plot Histogram
figure;
histogram(data);
title('Histogram of Random Data');
```

OUTPUT:

Random Data:0.08

Standard Deviation of Random Data:1.07



Experiment No. – 15

Simple Linear Regression

OBJECTIVE:

Simple linear regression involves a single independent variable. Let's say we have some data x and y, and we want to fit a line y = a*x+b.

<u>INPUT</u>:

%Simple Linear Regression

%Sample Data

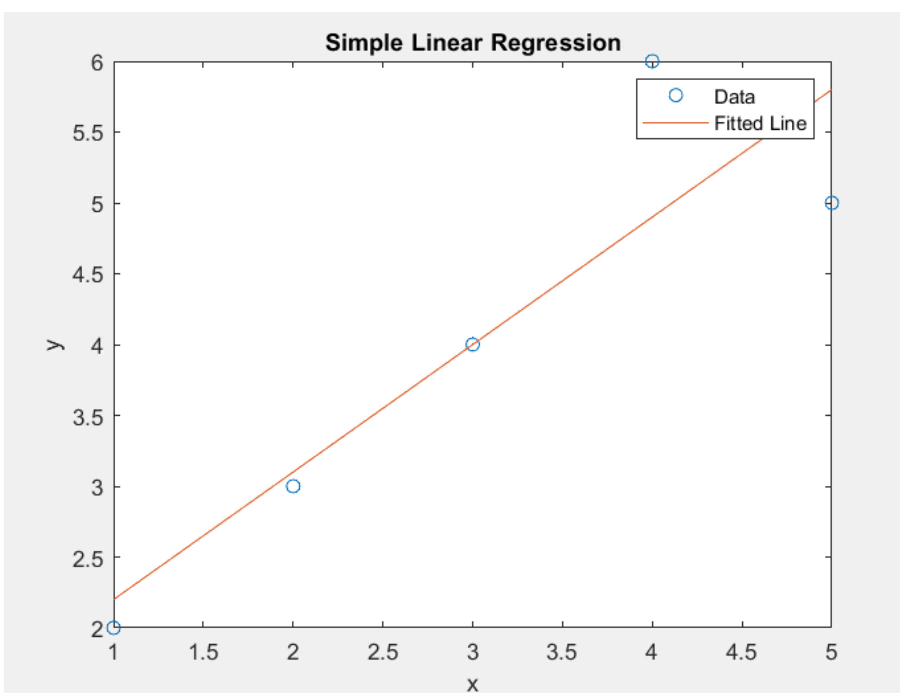
x=[1,2,3,4,5];

y=[2,3,4,6,5];

%Perform Linear Regression

```
p = polyfit(x,y,1); %p(1) is slope, p(2) is intercept

%Create a linear model

y_fit = polyval(p,x);

%Plot

figure;

plot(x,y,'o'); %Original data

hold on;

plot(x,y_fit,'-'); %Fitted Line

title('Simple Linear Regression');

xlabel('x');

ylabel('y');

legend('Data','Fitted Line');
```

<u>OUTPUT</u>:



Experiment No. – 16

Multiple Linear Regression

OBJECTIVE:

In multiple linear regression, we predict a dependent variable based on multiple independent variables.

<u>INPUT</u>:

```
%Multiple Linear Regression

%Sample Data

X=[1 2 3; 2 3 4; 3 4 5; 4 5 6; 5 6 7]; %Each Row is an observation

y=[2;3;4;6;5]; %Dependent variable

%Add a column of ones to X for the intercepts

X=[ones(size(X,1),1),X];

%Perform regression

b = regress(y,X); %Returns the regression coefficients

%Predicted Values

y_pred = X*b;

%Display the coefficients

disp('Coefficients(including intercept:');

disp(b);

%Plot-only practical if you have 1 or 2 independent variables

%For more variables, consider #D plots or partial regression plots
```

OUTPUT:

Coefficients (including intercept:

```
        0
   0.2500
        0
   0.6500
```

Experiment No. – 17

Polynomial Regression

OBJECTIVE:

Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y.

INPUT:

```
%Polynomial Regression

%Sample Data

x=[1,2,3,4,5];

y=[2,4,6,8,10];

%Polynomial Degree

degree =2;

%Perform polynomial regression

p= polyfit(x,y,degree);

%Create a polynomial model

x_fit = linspace(min(x),max(x),100); %100 points for a smoother plot

y_fit = polyval(p,x_fit);

%Plot
```

```
figure;
plot(x,y,'o'); %Original Data
hold on;
plot(x_fit,y_fit,'-'); %Fitted polynomial
title('Polynomial Regression');
xlabel('x');
ylabel('y');
legend('Data','Fitted polynomial');
```

OUTPUT: