Inferential Statistics Assignment

- 19230

Question:

Consider the following distributions

(i) Binomial random variable with parameter p = 0.5 and n = 100.
(ii) Poisson random variable with parameter λ = 3.
(iii) Uniform distribution with parameters a = 1 and b = 5.
(iv) Exponential distribution with parameter λ = 3.
(v) Normal distribution with parameters μ = 5 and σ = 1.
(vi) Standard normal distribution.

Answer the following questions for each distribution:

(1) Generate 1000 random numbers.
(2) Plot the density curve.
(3) Perform exploratory data analysis. i.e,
- Data summary: compute maximum, mean, meadian, minimum, standard deviation, variation, IQR (inter quantile range).
- Data visualization: Dot plot, Histogram, Box plot and Q-Q plot.

(4) Identify outliers.

Answer:

The application consists of two files, main.py and distributions.py.
distributions.py contains methods to find details about any distribution. It is a library that comtains methods to plot density curve,perform explonatory data analysis, data visualization and indetify outliers.
main.py generates various random numbers of different distributions. This file imports distributions.py file and calls various functions in the library to accomplish the task.

main.py

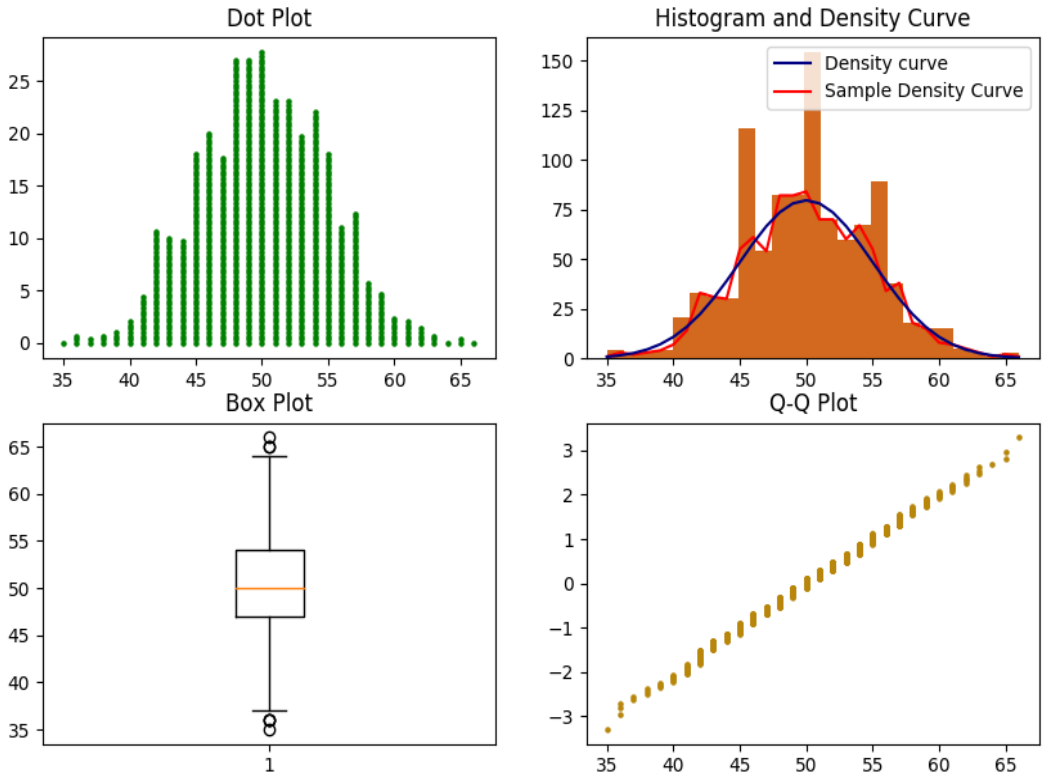
```
6  binomial = np.random.binomial(100,0.5,1000)
7  exponential = np.random.exponential(3,1000)
8  exponential -= exponential%1  ##truncating data
9  normal = np.random.normal(5,1,1000)
10 normal -= normal%0.1  ##truncating data
11 poisson = np.random.poisson(3,1000)
12 std_nrm = np.random.normal(0,1,1000)
13 std_nrm -= std_nrm%0.1  ##truncating data
14 uniform = np.random.uniform(1,5,1000)
15 uniform -= uniform%0.1  ##truncating data
```

Generated all dstributions using numpy.

Binomial Distribution Code with p = 0.5 and n = 100 :

```
##########Binomial##############
binomial = np.sort(binomial)
fig,axs = plt.subplots(2,2,figsize = (10,10))
print("++++++++++++BINOMIAL++++++++++++")
print("Maximum            :",db.maximum(binomial))
print("Minimum            :",db.minimum(binomial))
print("Mean               :",db.mean(binomial))
print("Median             :",db.median(binomial))
print("Variance           :",db.var(binomial,db.mean(binomial)))
print("Standard Deviation :",db.std_dev(binomial,db.mean(binomial)))
print("IQR                :",db.iqr(binomial))
db.dotplot(binomial,axs[0,0])
y = stats.binom.pmf(binomial,100,0.5)*1000 ##get pdf for each point in x
db.histogram(binomial,y,axs[0,1])
out = db.boxplot(binomial,axs[1,0])
db.qq_plot(binomial,axs[1,1])
outliers = list(item.get_ydata() for item in out['fliers'])
outliers = list(outliers[0])
outliers,y = db.freq(outliers)
print("Outliers           :",outliers)
plt.show()
```

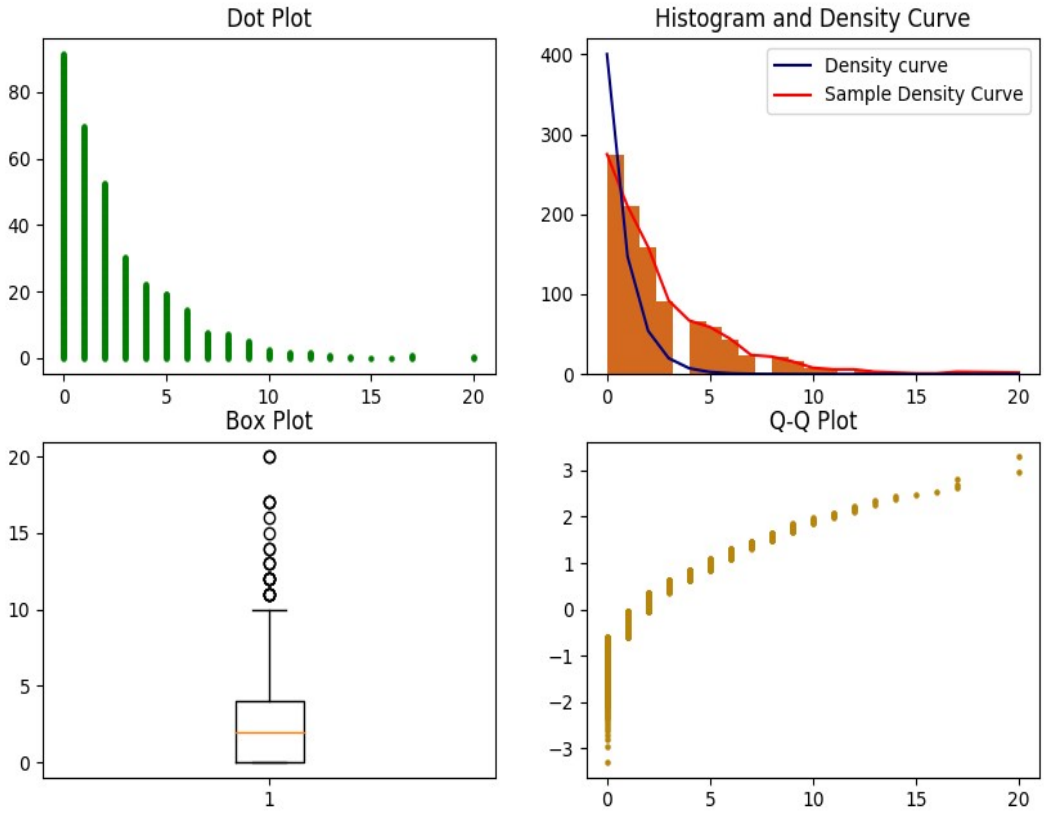
Binomial Distribution Output :



Exponential Distribution Code with λ = 3 :

```
#############Exponential##################
exponential = np.sort(exponential)
fig,axs = plt.subplots(2,2,figsize = (10,10))
print("++++++++++++EXPONENTIAL++++++++++++")
print("Maximum            :",db.maximum(exponential))
print("Minimum            :",db.minimum(exponential))
print("Mean               :",db.mean(exponential))
print("Median             :",db.median(exponential))
print("Variance           :",db.var(exponential,db.mean(exponential)))
print("Standard Deviation :",db.std_dev(exponential,db.mean(exponential)))
print("IQR                :",db.iqr(exponential))
db.dotplot(exponential,axs[0,0])
y = stats.expon.pdf(exponential,0)*400  ##get pdf for each point in x
db.histogram(exponential,y,axs[0,1])
out = db.boxplot(exponential,axs[1,0])
db.qq_plot(exponential,axs[1,1])
outliers = list(item.get_ydata() for item in out['fliers'])
outliers = list(outliers[0])
outliers,y = db.freq(outliers)
print("Outliers           :",outliers)
plt.show()
```

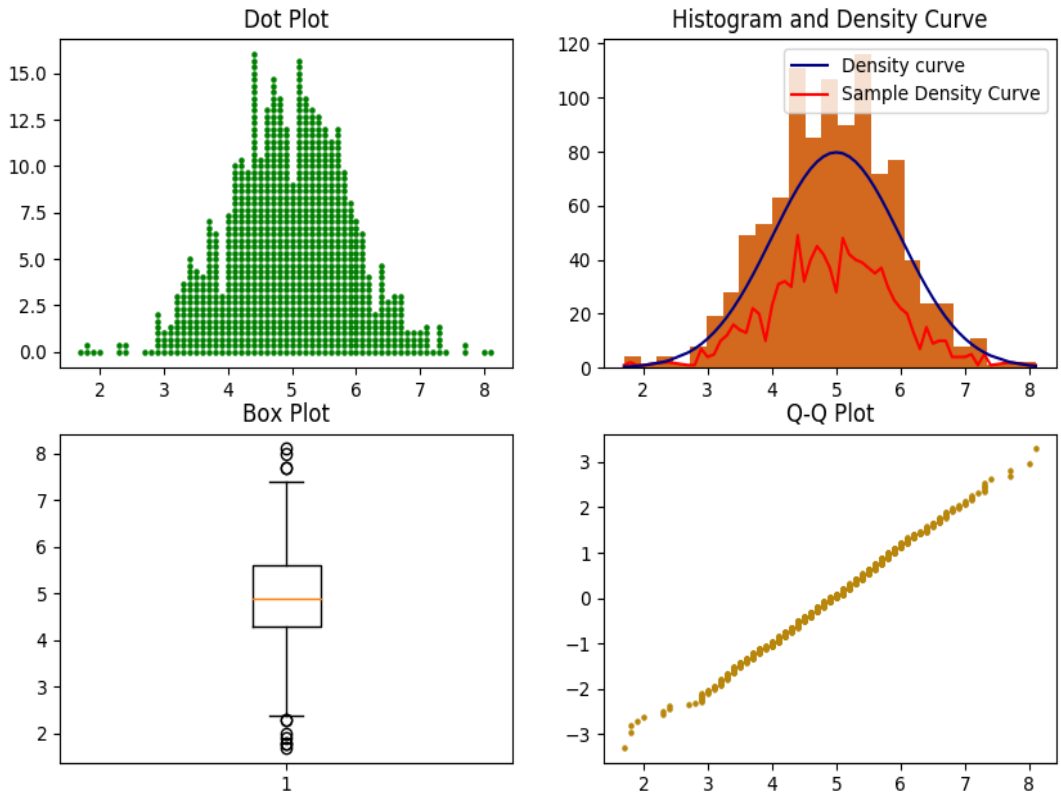
Exponential Distribution Output :



Normal Distribution Code with μ = 5 and σ = 1 :

```
##########Normal##############
normal = np.sort(normal)
fig,axs = plt.subplots(2,2,figsize = (10,10))
print("++++++++++++NORMAL++++++++++++")
print("Maximum             :",db.maximum(normal))
print("Minimum             :",db.minimum(normal))
print("Mean                :",db.mean(normal))
print("Median              :",db.median(normal))
print("Variance            :",db.var(normal,db.mean(normal)))
print("Standard Deviation  :",db.std_dev(normal,db.mean(normal)))
print("IQR                 :",db.iqr(normal))
db.dotplot(normal,axs[0,0])
y = stats.norm.pdf(normal,5,1)*200
db.histogram(normal,y,axs[0,1])
out = db.boxplot(normal,axs[1,0])
db.qq_plot(normal,axs[1,1])
outliers = list(item.get_ydata() for item in out['fliers'])
outliers = list(outliers[0])
outliers,y = db.freq(outliers)
print("Outliers            :",outliers)
plt.show()
```

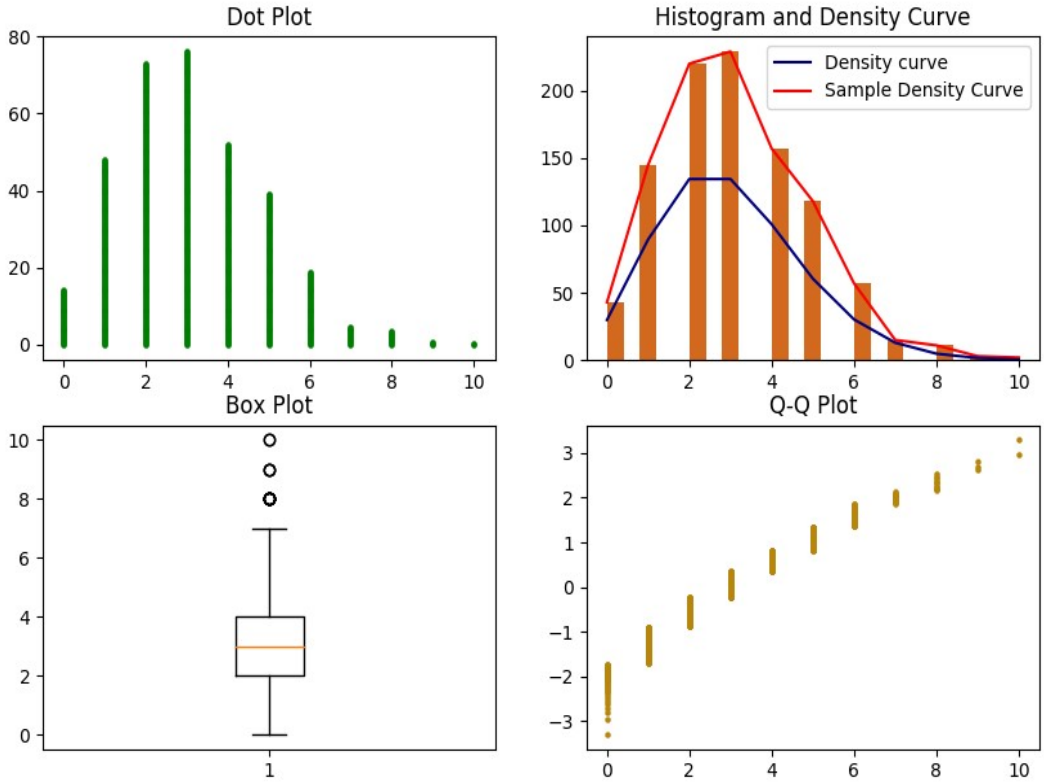
Normal Distribution Output :



Piosson Distribution Code with λ = 3 :

```
##########Poisson##############
poisson = np.sort(poisson)
fig,axs = plt.subplots(2,2,figsize = (10,10))
print("++++++++++++POISSON++++++++++++")
print("Maximum            :",db.maximum(poisson))
print("Minimum            :",db.minimum(poisson))
print("Mean               :",db.mean(poisson))
print("Median             :",db.median(poisson))
print("Variance           :",db.var(poisson,db.mean(poisson)))
print("Standard Deviation :",db.std_dev(poisson,db.mean(poisson)))
print("IQR                :",db.iqr(poisson))
db.dotplot(poisson,axs[0,0])
y = stats.poisson.pmf(poisson,3)*600
db.histogram(poisson,y,axs[0,1])
out = db.boxplot(poisson,axs[1,0])
db.qq_plot(poisson,axs[1,1])
outliers = list(item.get_ydata() for item in out['fliers'])
outliers = list(outliers[0])
outliers,y = db.freq(outliers)
print("Outliers           :",outliers)
plt.show()
```

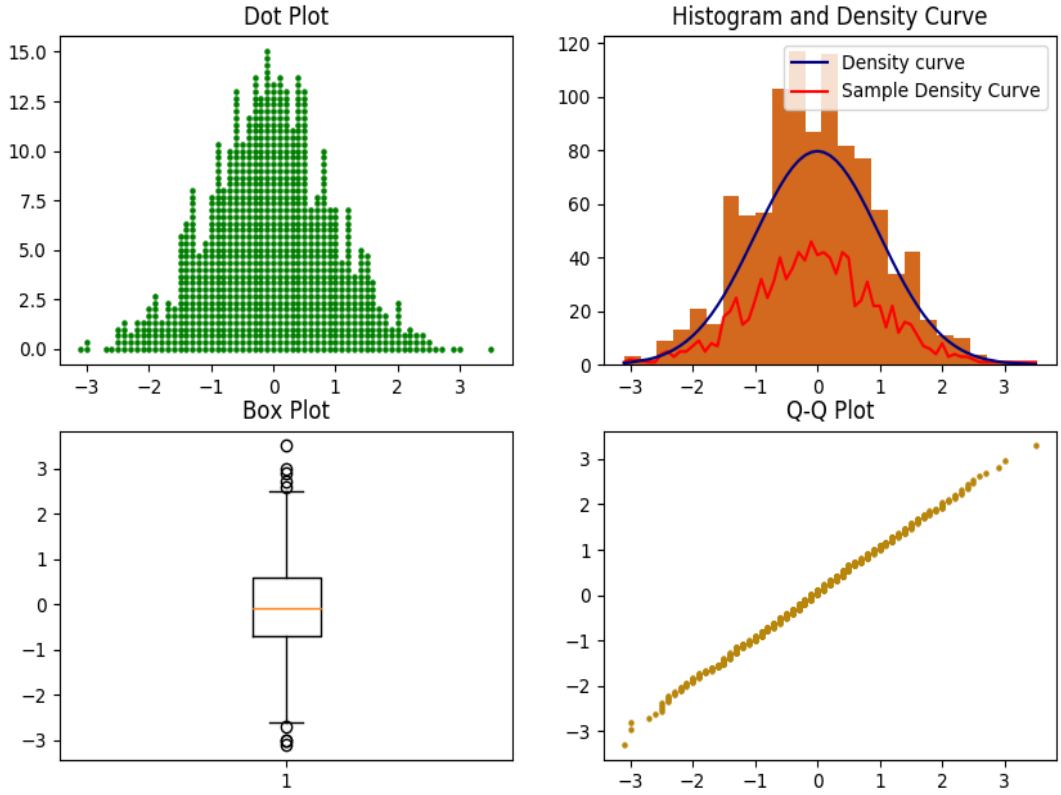
Poisson Distribution Output :



Standard Normal Distribution Code :

```
##########Standard Normal###############
std_nrm = np.sort(std_nrm)
fig,axs = plt.subplots(2,2,figsize = (10,10))
print("++++++++++++STANDARD NORMAL++++++++++++")
print("Maximum            :",db.maximum(std_nrm))
print("Minimum            :",db.minimum(std_nrm))
print("Mean               :",db.mean(std_nrm))
print("Median             :",db.median(std_nrm))
print("Variance           :",db.var(std_nrm,db.mean(std_nrm)))
print("Standard Deviation :",db.std_dev(std_nrm,db.mean(std_nrm)))
print("IQR                :",db.iqr(std_nrm))
db.dotplot(std_nrm,axs[0,0])
y = stats.norm.pdf(std_nrm)*200
db.histogram(std_nrm,y,axs[0,1])
out = db.boxplot(std_nrm,axs[1,0])
db.qq_plot(std_nrm,axs[1,1])
outliers = list(item.get_ydata() for item in out['fliers'])
outliers = list(outliers[0])
outliers,y = db.freq(outliers)
print("Outliers           :",outliers)
plt.show()
```

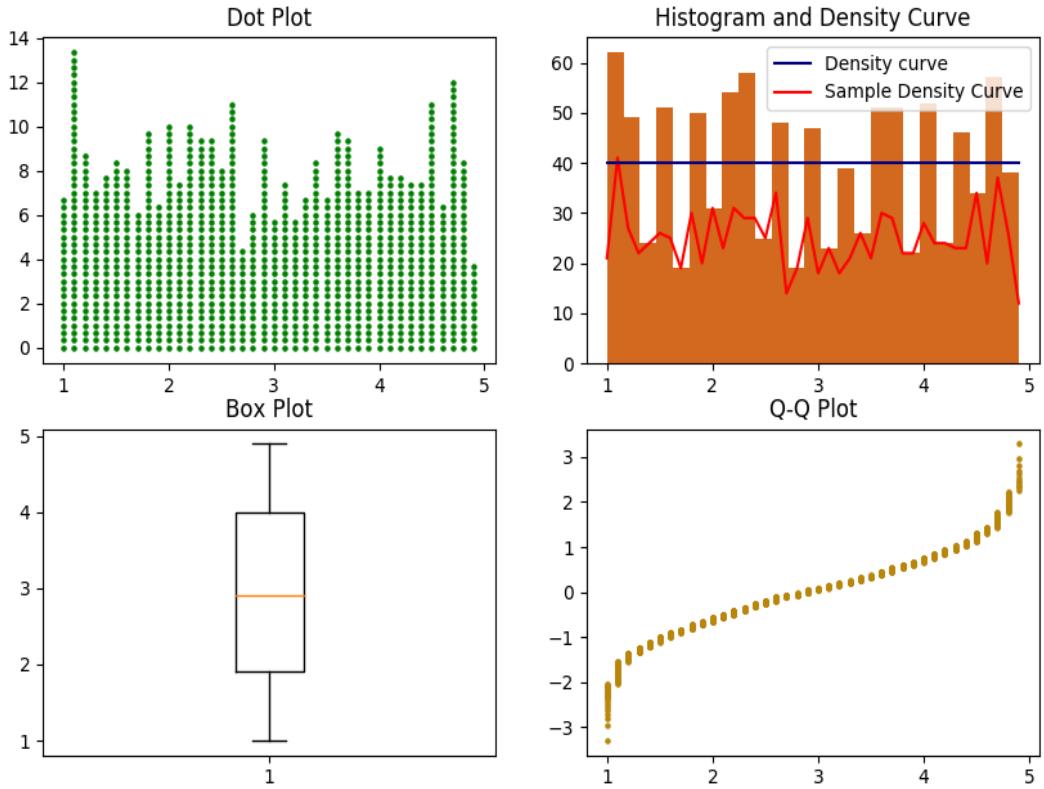
Standard Normal Output :



Uniform Distribution Code with a = 1 and b = 5 :

```
##########Uniform##############
uniform = np.sort(uniform)
fig,axs = plt.subplots(2,2,figsize = (10,10))
print("++++++++++++UNIFORM++++++++++++")
print("Maximum             :",db.maximum(uniform))
print("Minimum             :",db.minimum(uniform))
print("Mean                :",db.mean(uniform))
print("Median              :",db.median(uniform))
print("Variance            :",db.var(uniform,db.mean(uniform)))
print("Standard Deviation :",db.std_dev(uniform,db.mean(uniform)))
print("IQR                 :",db.iqr(uniform))
db.dotplot(uniform,axs[0,0])
y = stats.uniform.pdf(uniform,1,5)*200
db.histogram(uniform,y,axs[0,1])
out = db.boxplot(uniform,axs[1,0])
db.qq_plot(uniform,axs[1,1])
outliers = list(item.get_ydata() for item in out['fliers'])
outliers = list(outliers[0])
outliers,y = db.freq(outliers)
print("Outliers            :",outliers)
plt.show()
```

Uniform Distribution Output :



Output on the Terminal :

```
++++++++++BINOMIAL++++++++++++
Maximum            : 66
Minimum            : 35
Mean               : 50.051
Median             : 50.0
Variance           : 24.446845845845925
Standard Deviation : 4.94437517244049
IQR                : 7
Outliers           : [35, 36, 65, 66]
++++++++++EXPONENTIAL++++++++++++
Maximum            : 20.0
Minimum            : 0.0
Mean               : 2.526
Median             : 2.0
Variance           : 8.750074074074067
Standard Deviation : 2.9580524123270817
IQR                : 4.0
Outliers           : [11.0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 20.0]
++++++++++NORMAL++++++++++++
Maximum            : 8.1
Minimum            : 1.7000000000000002
Mean               : 4.935400000000005
Median             : 4.9
Variance           : 0.922429269269266
Standard Deviation : 0.9604318139614421
IQR                : 1.3000000000000007
Outliers           : [1.7000000000000002, 1.8, 1.9000000000000001, 2.0,
2.3000000000000003, 7.7, 8.0, 8.1]
++++++++++POISSON++++++++++++
Maximum            : 10
Minimum            : 0
Mean               : 3.072
Median             : 3.0
Variance           : 3.047863863863931
Standard Deviation : 1.7458132385406897
IQR                : 2
Outliers           : [8, 9, 10]
++++++++++STANDARD NORMAL++++++++++++
Maximum            : 3.5
Minimum            : -3.1
Mean               : -0.05690000000000027
Median             : -0.1
Variance           : 1.02365604604605
Standard Deviation : 1.0117588873076677
IQR                : 1.3000000000000003
Outliers           : [-3.1, -3.0, -2.7, 2.6, 2.7, 2.9000000000000004, 3.0, 3.5]
++++++++++UNIFORM++++++++++++
Maximum            : 4.9
Minimum            : 1.0
Mean               : 2.9173999999999976
Median             : 2.9000000000000004
Variance           : 1.3610783183183102
Standard Deviation : 1.1666526125279582
IQR                : 2.0999999999999996
Outliers           : []
```

The Library Functions :

```
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

def freq(data):  ##returns data nad it's frequency
    frequency = {}
    for i in data:
        if(i in frequency.keys()):
            frequency[i]+=1
        else:
            frequency[i]=1
    return list(frequency.keys()),list(frequency.values())

def maximum(data):  ##returns maximum of sorted data
    return data[-1]

def minimum(data):  ##returns minimum of sorted data
    return data[0]

def iqr(data):  ##returns iqr of sorted data
    q1 = int(len(data)/4)
    q3 = int(3*len(data)/4)
    return data[q3]-data[q1]  ##iqr is the difference between q1 and q3

def mean(data):  ##returns mean of the data
    sum = 0
    for i in data:
        sum+=i
    avg = sum/len(data)
    return avg

def median(data):  ##returns median of the sorted data
    size = len(data)
    if(size%2 == 1):
        med = data[int(size/2)]
    else:
        med = (data[int(size/2)] + data[int(size/2)-1])/2
    return med

def std_dev(data,avg):  ##returns standard deviation of the data
    vari = var(data,avg)  ##uses variance method which is available in the library
    return np.sqrt(vari)

def var(data,avg):  ##returns variance of the data
    diff = 0
    for i in data:
        diff += (i-avg)**2
    vari = diff/(len(data)-1)  ## formula for sample variance
    return vari

def dotplot(data,ax):  ##attaches scatterplot of data  as dotplot to the axes
    ax.set_title("Dot Plot")
    x,y = freq(data)
    for i in range(len(x)):
        for j in range(y[i]):
            ax.scatter(x[i],j/3,color='green',s=5)

def histogram(data,pd,ax):  ##attaches histogram of the data to the axes
    ax.hist(data,bins=25,color='chocolate')
    ax.set_title("Histogram and Density Curve")
    x,y = freq(data)
    ax.plot(x,y,color='red')  ##plot the density curve of the data
    ax.plot(data,pd,color='navy')
    ax.legend([plt.Line2D([0],[0],color='navy'),(plt.Line2D([0],[0],color='red'))],
['Density curve','Sample Density Curve'])
```

```
def boxplot(data,ax):  ##returns boxplot dictionary
    ax.set_title("Box Plot")
    return(ax.boxplot(data,vert = True))

def qq_plot(data,ax):  ##attaches qq_plot to the axes
    q = []
    for i in range(len(data)):
        q.append(stats.norm.ppf((i+0.5)/len(data)))
    ax.scatter(data,q,s=5,color='darkgoldenrod')
    #ax.plot(data,q,color='green')
    ax.set_title("Q-Q Plot")
```