CSE 411 ASSIGNMENT 3

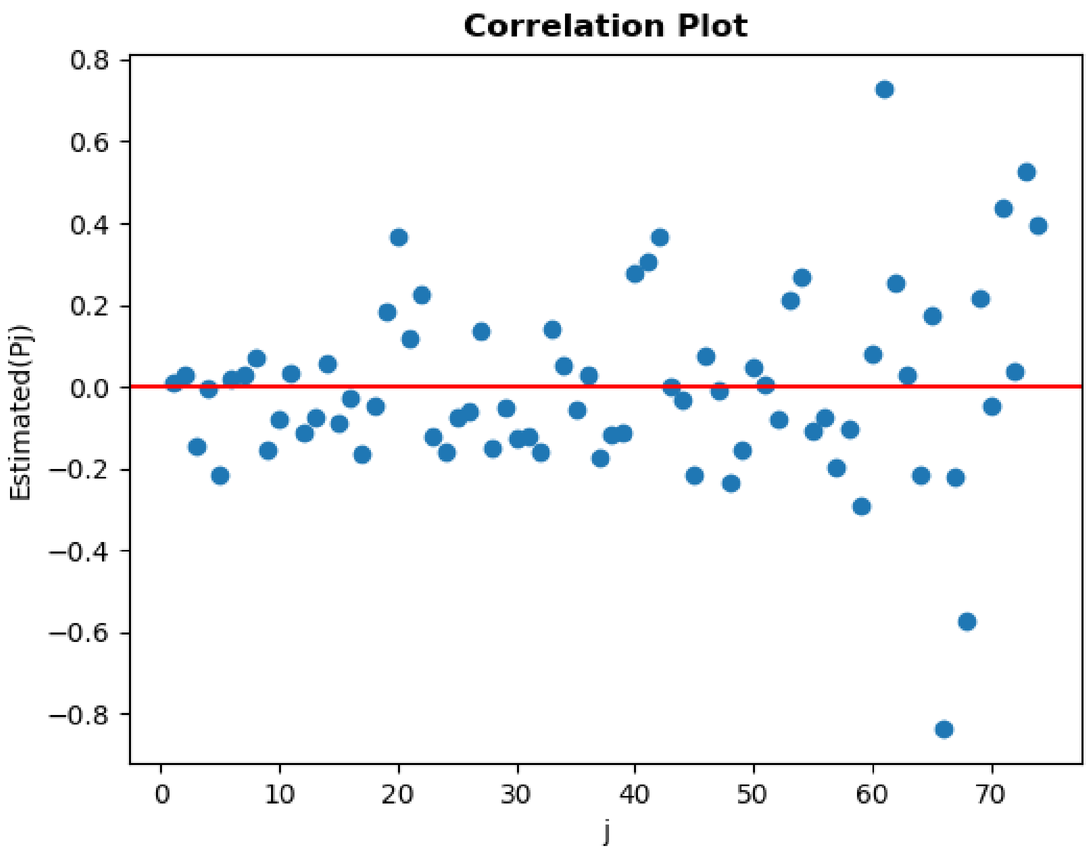
Submitted by:
Md. Tawkat Islam Khondaker
Student Id: 1405036
Md. Hasin Abrar
Student Id: 1405048

ASUS

We are using the data of student id: 1405036

Sample Independence Assessment:

- <u>Correlation Plot:</u>

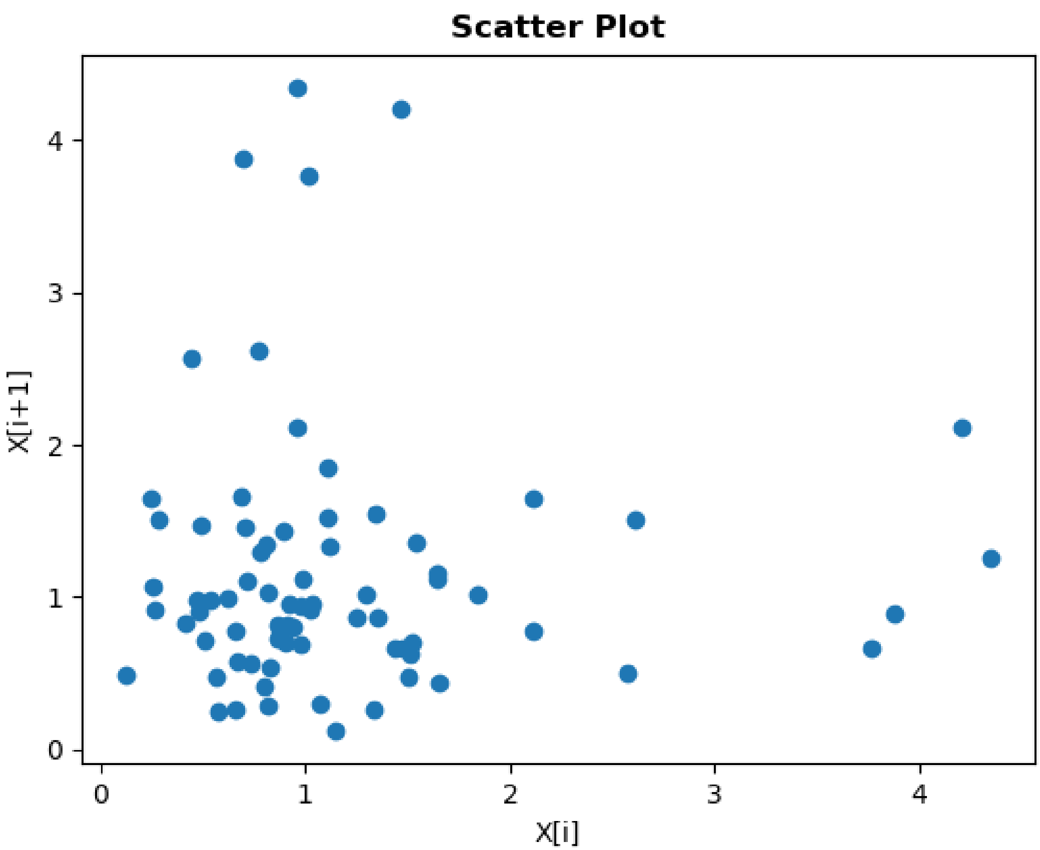



```
from statistics import mean
def S2(data):
    n=len(data)
    mean_value=mean(data)
    s2=0
    for x in data:
        s2+=(x-mean_value)**2
    s2/=(n-1)
    return s2

def Cj(data,j):
    n=len(data)
```

```python
    mean_value=mean(data)

    cj=0
    for i in range(n-j):
        cj+=(data[i]-mean_value)*(data[i+j]-mean_value)
    cj/=(n-j)
    return cj

def Pj(data,j):
    return Cj(data,j)/S2(data)


def Corr(data,j=None):
    if(j==None):
        j=len(data)-1

    X = [i for i in range(1, j+1)]
    Y = []
    for x in X:
        Y.append(Pj(data, x))
    return X,Y

def Corr_Plot(X,Y):
    plt.axhline(0, color='red')
    plt.scatter(X,Y)
    plt.xlabel('j')
    plt.ylabel('Estimated(Pj)')
    plt.title('Correlation Plot',fontweight='bold')
    plt.savefig('Correlation_Plot j= '+str(len(X))+'.png')
```

Most of the plotted points are near the horizontal line regardless of some outliers that is situated far from the horizontal line. So, we can say observations are independent.

- <u>Scatter Diagram:</u>




```
def Scatter(data):
   X=[]
   Y = []
   for i in range(len(data)-1):
      X.append(data[i])
      Y.append(data[i+1])
   return X,Y

def Scatter_Plot(X,Y):
   plt.scatter(X,Y)
   plt.xlabel('X[i]')
   plt.ylabel('X[i+1]')
   plt.title('Scatter Plot', fontweight='bold')
   #plt.show()
   plt.savefig('Scatter_Plot.png')
```

Here we can see this graph is sparse and does not show any positive or negative slop. So, we can say observations are independent.

Activity I: Hypothesizing Families of Distributions:

- <u>Summary Statistic:</u>

```
from statistics import median,mean,variance
import math
from scipy.stats import skew

def CV(data):
   var=variance(data)
   mean_data=mean(data)
   return (math.sqrt(var)/mean_data)

def Lexis_Ratio(data):
   var=variance(data)
   mean_data = mean(data)
   return (var/mean_data)

min_value=min(data)
max_value = max(data)
mean_data=mean(data)
median_data = median(data)
var = variance(data)
skewness=skew(data)
```

Minimum: 0.126

Maximum: 4.345

Mean: 1.1599466666666667

Median: 0.943

Variance: 0.7328737809009009

Coefficient of Variance: 0.7380343427128377

Lexis Ratio: 0.6318167912038205

Skewness: 2.1265552574071966

Result:

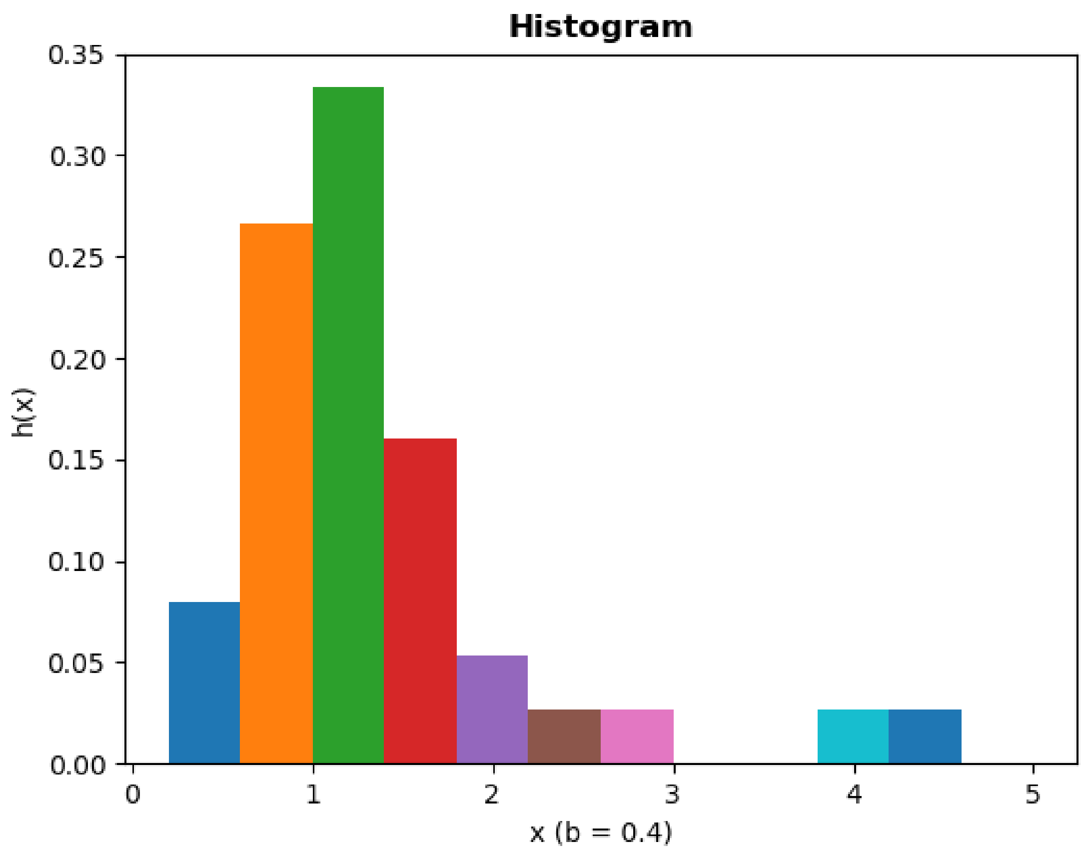
There is no specific domain for generated data. So, sample dataset is continuous.

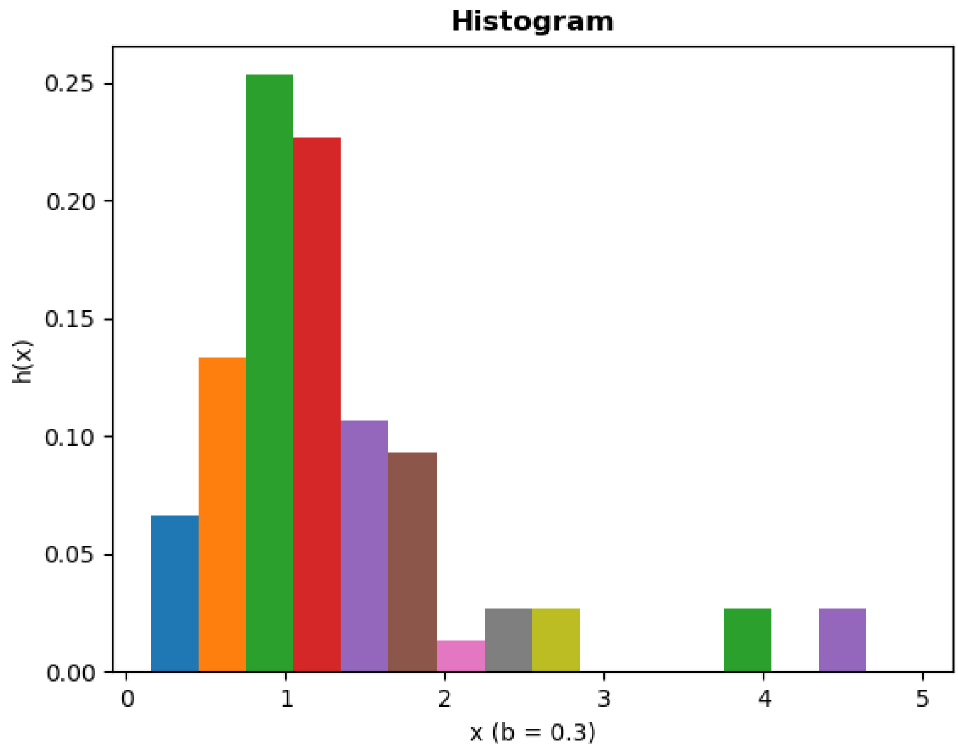
Mean > Median: Not Symmetric.

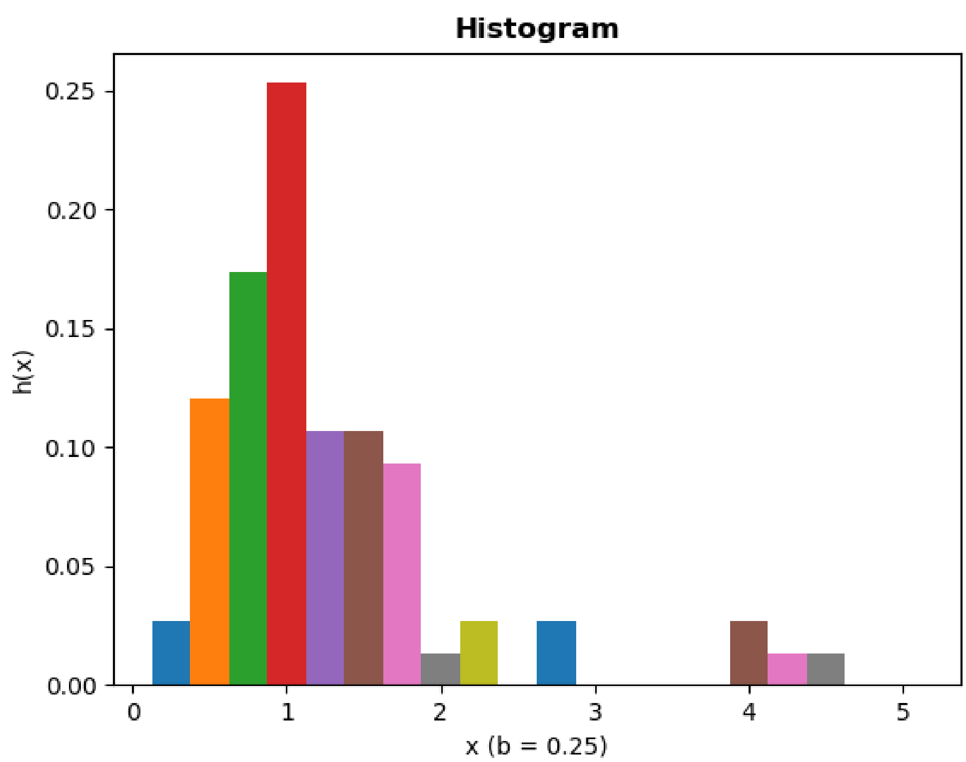
Skewness > 0: Skewed to Right.

Coefficient of variation < 0: Weibull or Gamma.

- Histogram:




Histogram
h(x)
0.25
0.20
0.15
0.10
0.05
0.00
0
1
2
3
4
5
x (b = 0.3)


Histogram
h(x)
0.25
0.20
0.15
0.10
0.05
0.00
0
1
2
3
4
5
x (b = 0.25)

```
def Data_Interval(data,b=0.1):
   n=len(data)
   max_range=math.ceil(max(data))
   X=[]
   Y=[]
   low=0
   high=b
   while(high<=max_range):
      X.append(high)
      Y.append((len([d for d in data if low<=d<high]))/n)
      low=high
      high+=b

   return X,Y

def Plot_Histogram(X,Y,b):
   for i,x in enumerate(X):
      plt.bar(x, Y[i], width=b)

   plt.xlabel('x (b = '+str(b)+')')
   plt.ylabel('h(x)')
   plt.title('Histogram', fontweight='bold')
   plt.savefig('Histogram Plot for b='+str(b)+'.png')


def Diff_Interval_Plot(data,b=0.1):
   X,Y=Data_Interval(data,b)
   Plot_Histogram(X,Y,b)
```

We can see relatively smooth-looking shape occurs at Del(b)= 0.3, with k=17 number of intervals.

Moreover, the shape of the Histogram resembles with that of a Weibull density.

- <u>Quantile Summaries and Box Plot:</u>

```
from statistics import mean,median
import matplotlib.pyplot as plt

def Quantile(data):
    data.sort()
    n=len(data)
    i=(n+1)//2
    median_data = median(data)
    j=(i+1)//2
    quartile0=data[j-1]
    quartile1=data[(n-j+1)-1]
    quartile=(quartile0+quartile1)/2
    k=(j+1)//2
    octile0=data[k-1]
    octile1=data[(n-k+1)-1]
    octile=(octile0+octile1)/2
    extreme=(data[0]+data[n-1])/2

    return median_data,quartile,octile,extreme


def Box_Plot(data,median,quartile,octile,extreme):
    plt.boxplot(data, showmeans=True, whis=max(data)+10, vert=False)

    plt.axvline(median, color='cyan', label='Median_Midpoint')
    plt.axvline(quartile, color='blue', label='Quartile_Midpoint')
    plt.axvline(octile, color='green', label='Octile_Midpoint')
    plt.axvline(extreme, color='red', label='Extreme_Midpoint')

    plt.legend()
    plt.title('Box Plot', fontweight='bold')
    plt.savefig('Box Plot.png')
```

Median: 0.943

Quantile[0]: 0.664

Quantile[1]: 1.436

Quantile Midpoint: 1.05

Octile[0]: 0.477

Octile[1]: 1.657

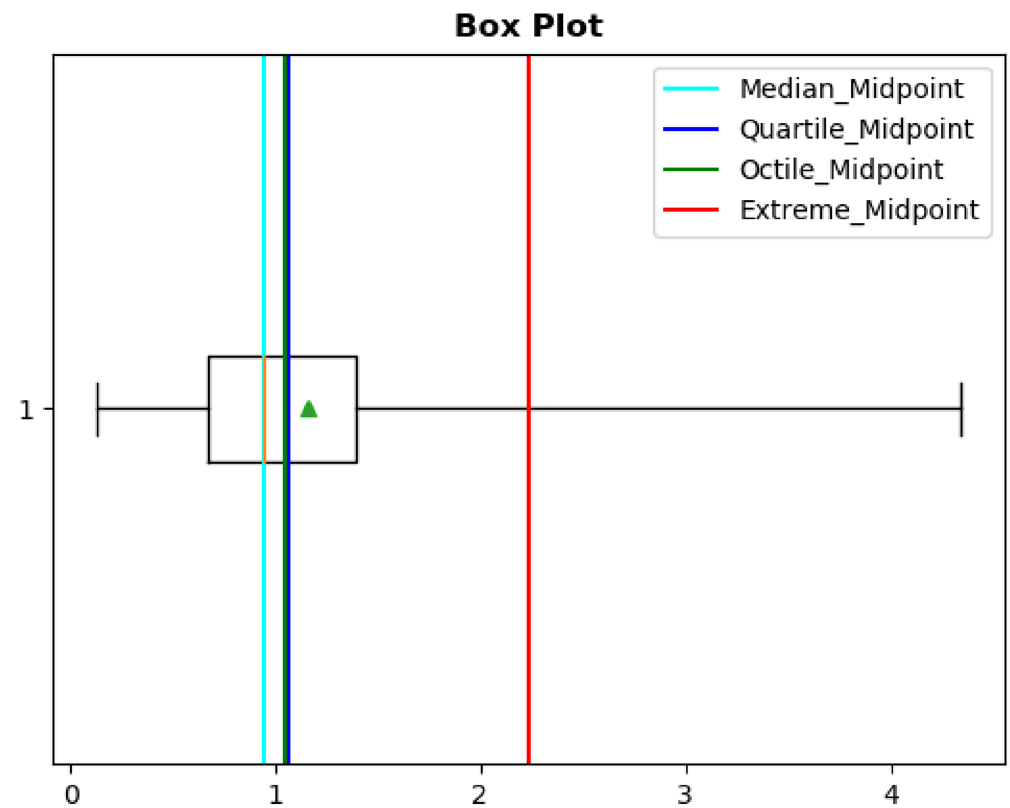
Octile Midpoint: 1.067

Extreme[0]: 0.126

Extreme[1]: 4.345

Extreme Midpoint: 2.2355

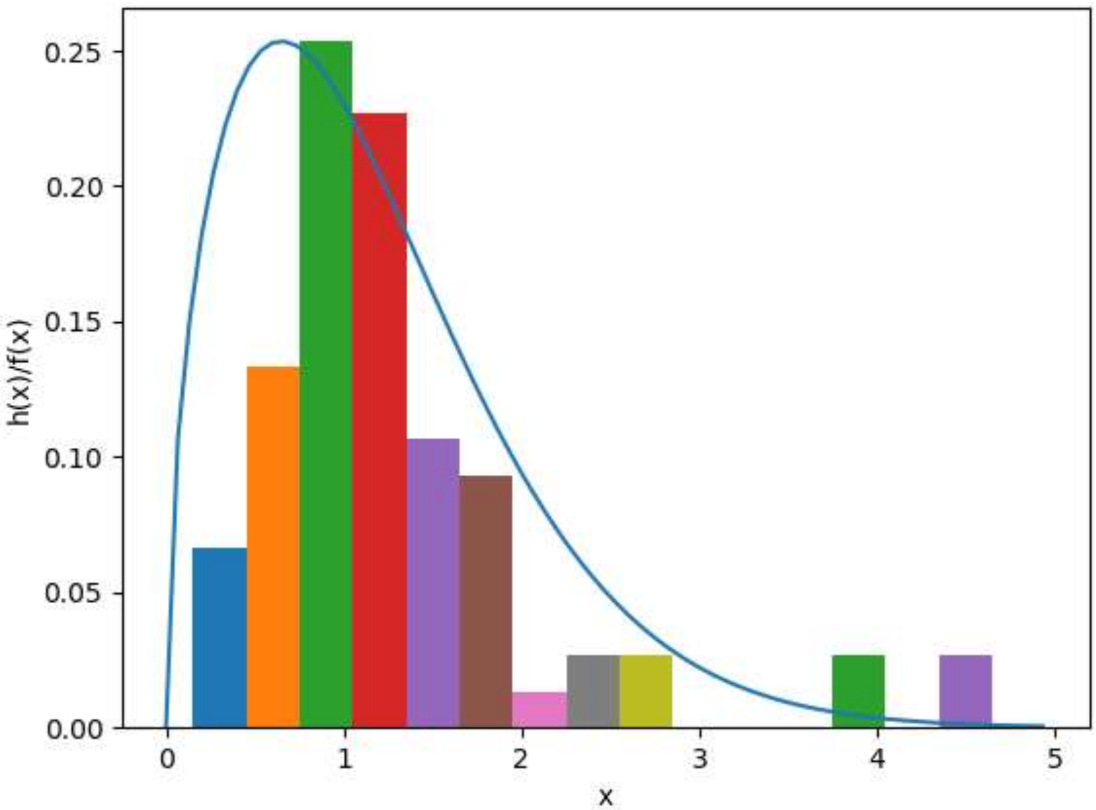
We can see midpoints are gradually increasing. So, underlying Distribution is Right Skewed.



The elongated nature of the right side of the box plot reaffirms our hypothesis that it is right skewed.

Activity II: Estimation of Parameters:

- <u>Estimation of Parameter:</u>



As our histogram resembles mostly with the density function of Weibull distribution, we will estimate parameters of Weibull(α, β), using Maximum-Likelihood Estimators(MLEs).

MLE for Weibull distribution:

Density function for Weibull distribution;

$$f(x) = \begin{cases} \alpha\beta^{-\alpha} x^{\alpha-1} e^{-(x/\beta)^\alpha}; & \text{if } x>0 \\ 0 & ; \text{otherwise} \end{cases}$$

Here, α = shape parameter ($\alpha>0$)

β = scale parameter ($\beta>0$)

So, the likelihood function is,

$$L(\alpha,\beta) = \prod_{i=1}^{n} \alpha\beta^{-\alpha} x_i^{\alpha-1} e^{-(x_i/\beta)^\alpha}$$

$$= \prod_{i=1}^{n} \frac{\alpha}{\beta}\left(\frac{x_i}{\beta}\right)^{\alpha-1} e^{-(x_i/\beta)^\alpha}$$

Maximizing $L(\alpha,\beta)$ is equivalent to maximizing $LL(\alpha,\beta) = \ln L(\alpha,\beta)$, which is the log-likelihood function.

$$\therefore LL(\alpha,\beta) = \sum_{i=1}^{n} \ln\left[\frac{\alpha}{\beta}\left(\frac{x_i}{\beta}\right)^{\alpha-1} e^{-\left(\frac{x_i}{\beta}\right)^\alpha}\right]$$

$$= \sum_{i=1}^{n} \left[\ln\alpha - \ln\beta + (\alpha-1)\ln x_i - (\alpha-1)\ln\beta - \left(\frac{x_i}{\beta}\right)^{\alpha} \right]$$

$$= n\left[\ln\alpha - \alpha\ln\beta\right] + (\alpha-1)\sum_{i=1}^{n}\ln x_i - \sum_{i=1}^{n}\left(\frac{x_i}{\beta}\right)^{\alpha}$$

As, the logarithm function is strictly increasing, maximizing $L(\alpha,\beta)$ is equivalent to maximizing $LL(\alpha,\beta)$ $\Rightarrow$ (α,β) maximizes $L(\alpha,\beta)$ if and only if (α,β)

maximizes $LL(\alpha,\beta)$. $LL(\alpha,\beta)$ will be maximized when

$$\frac{dLL(\alpha,\beta)}{d(\alpha,\beta)} = 0$$

Solving this, we get:

$$\frac{\sum_{i=1}^{n} x_i^{\hat{\alpha}} \ln x_i}{\sum_{i=1}^{n} x_i^{\hat{\alpha}}} - \frac{1}{\hat{\alpha}} = \frac{\sum_{i=1}^{n} \ln x_i}{n}$$

$$\hat{\beta} = \left(\frac{\sum_{i=1}^{n} x_i^{\hat{\alpha}}}{n}\right)^{1/\hat{\alpha}}$$

Using recursion for Newton iteration:

$$\hat{\alpha}_{k+1} = \hat{\alpha}_k + \frac{A + 1/\hat{\alpha}_k - C_k/B_k}{1/\hat{\alpha}_k + (B_k H_k - C_k^2)/B_k^2}$$

Where, $A = \frac{\sum_{i=1}^{n} \ln x_i}{n}$

$$B_k = \sum_{i=1}^{n} x_i^{\hat{\alpha}_k}$$

$$C_k = \sum_{i=1}^{n} x_i^{\hat{\alpha}_k} \ln x_i$$

$$H_k = \sum_{i=1}^{n} x_i^{\hat{\alpha}_k} (\ln x_i)^2$$

And initial value for $\hat{\alpha}$,

$$\hat{\alpha}_0 = \left\{ \frac{(6/\pi^2)\left[\sum_{i=1}^{n}(\ln x_i) - \left(\sum_{i=1}^{n} \ln x_i\right)^2 / n\right]}{n-1} \right\}^{-1/2}$$

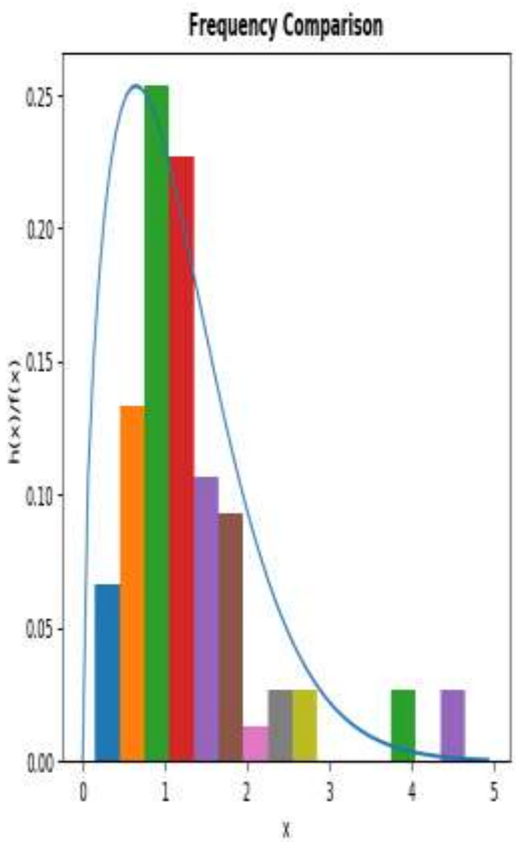
Then we can solve these equations using Excel solver.

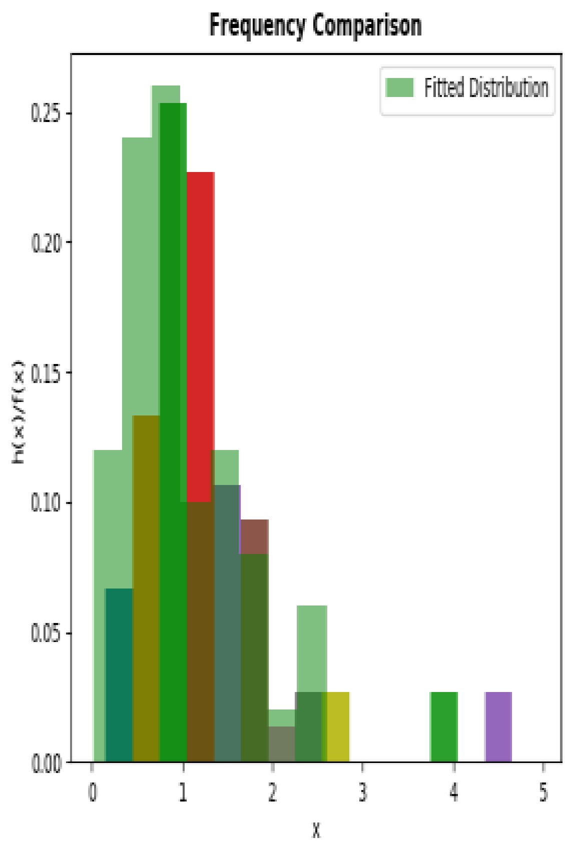
```
data=getData()


from xlwt import Workbook


wb = Workbook()

sheet1 = wb.add_sheet('Sheet 1')

row=4
col=1
for x in data:
    sheet1.write(row, col, x)
    row+=1


wb.save('Alpha Beta.xls')
```

From this we estimate, Alpha= 1.5278996202501096, Beta= 1.2999425079066453

Activity III: Determining How Representative the Fitted Distributions are:

- <u>Frequency Comparisons:</u>

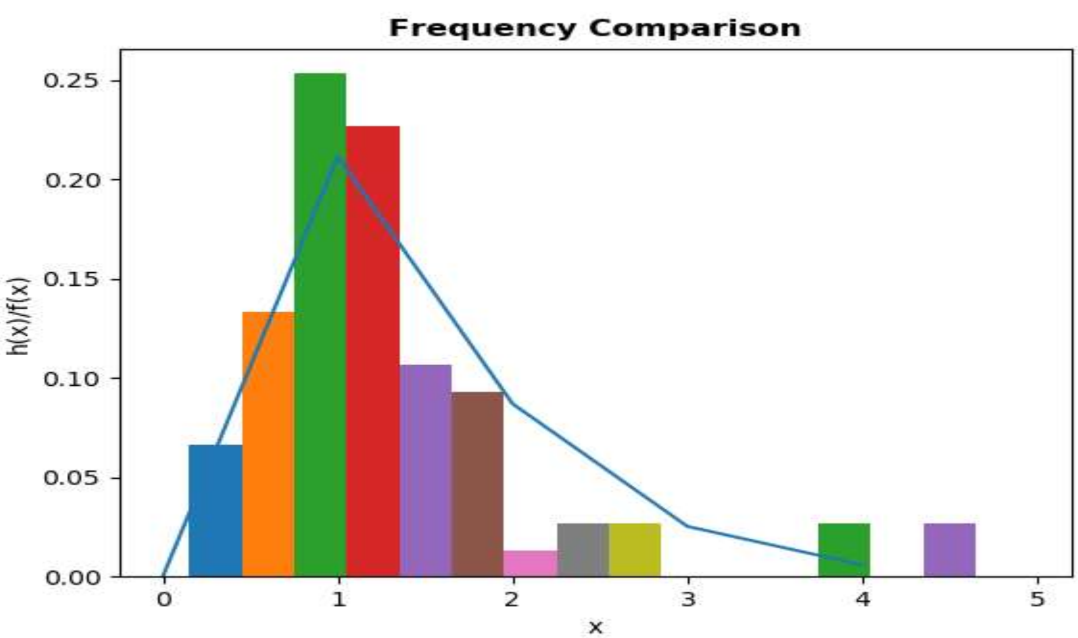





```
from Activity_1.Histogram import Data_Interval

def weibull(x,a,b):
   return (a * (b**(-a)) * (x**(a - 1)) * np.exp(-(x / b)**a))

def Plot_Histogram(X,Y,b):

   for i,x in enumerate(X):
      plt.bar(x,Y[i],width=b)

   genX=np.arange(0,75)/15
```

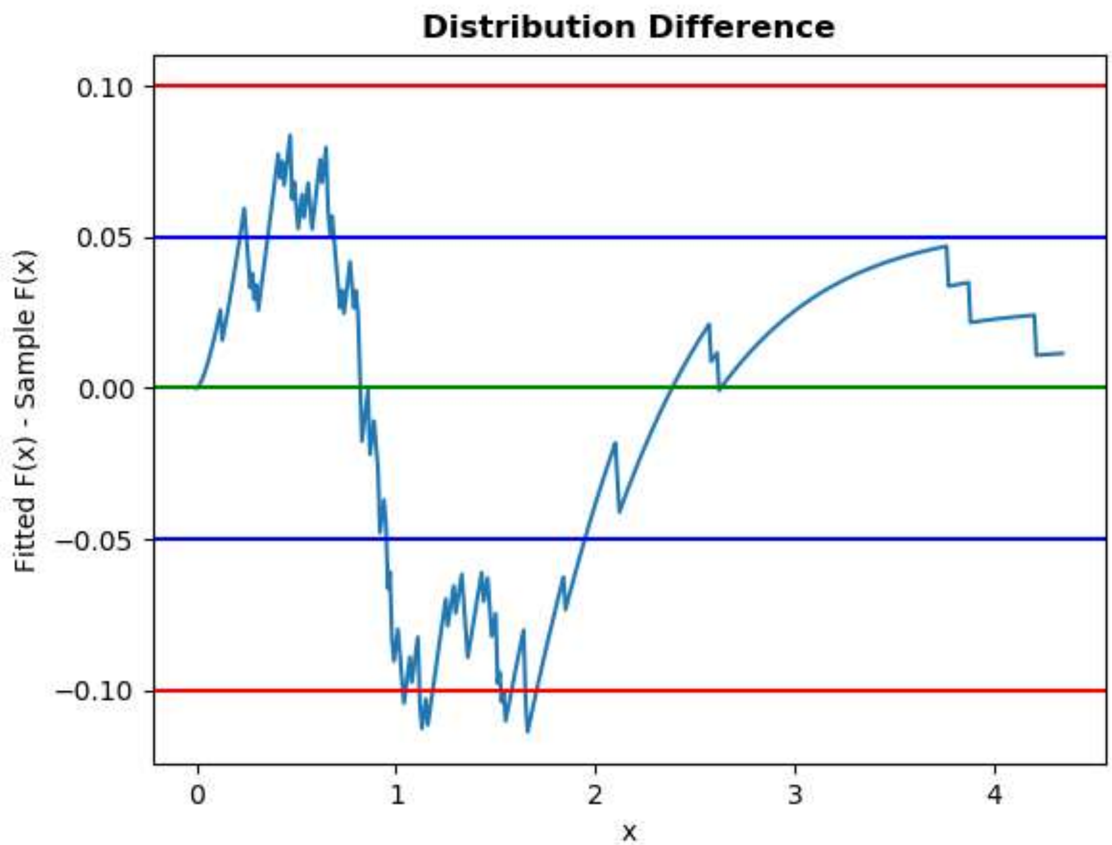
```
    plt.plot(genX, weibull(genX,
a=1.5278996202501096,b=1.2999425079066453)*scaling)

    plt.xlabel('x')
    plt.ylabel('h(x)/f(x)')
    plt.title('Frequency Comparison', fontweight='bold')
    plt.savefig('Frequency Comparison.png')


def Diff_Interval_Plot(data,b=0.1):
    X,Y=Data_Interval(data,b)
    Plot_Histogram(X,Y,b)
```

For Frequency Comparison, the fitted distribution is not a good representation of the true underlying distribution of the data because sample size(=75) is not sufficiently large, on the other hand fitted distribution is generated for sufficiently huge data.



But if we take the fitted distribution for small size of data we can see it almost follows the underlying distribution. So we can say, Frequency Comparisons almost represents the underlying distribution.

- <u>Distribution –Function –Difference:</u>

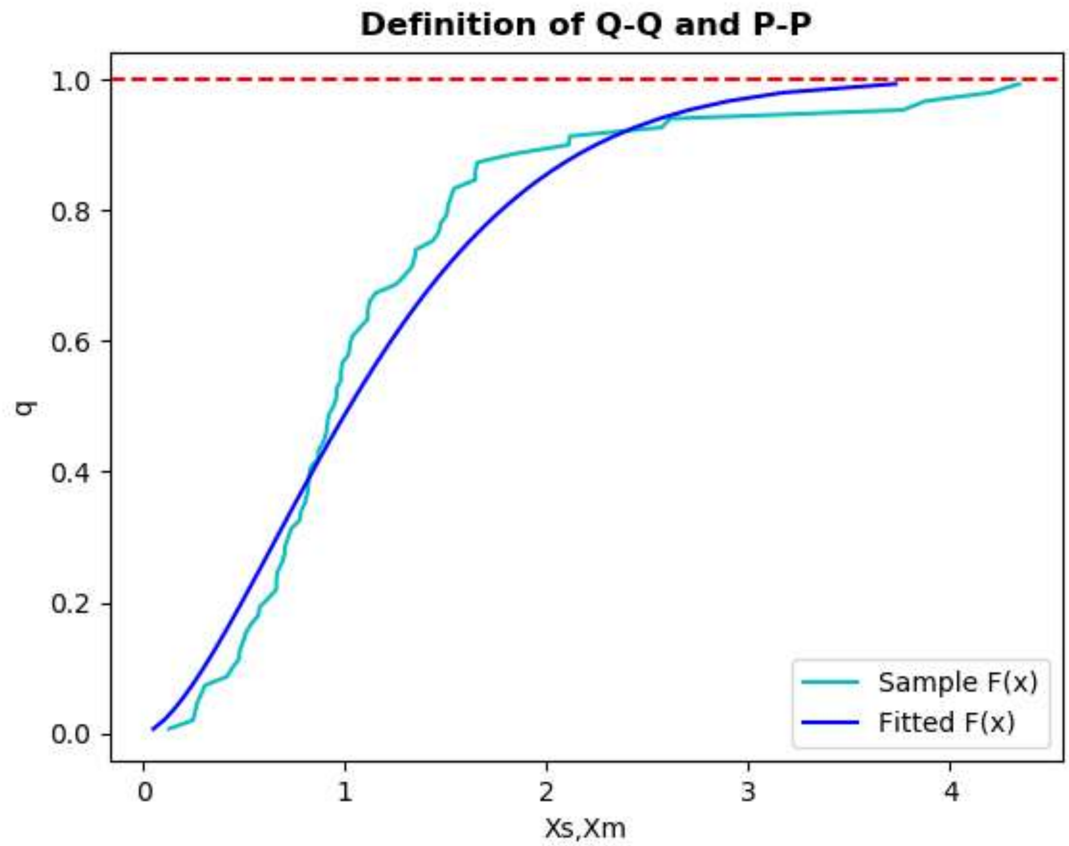



```
def Fitted_Distr(x,a=1.5278996202501096,b=1.2999425079066453):
   return (1-np.exp((-(x / b)**a)))

def Sample_Distr(data,x):
   n=len(data)
   cnt=0
   for d in data:
      if d<=x:
         cnt+=1
   return cnt/n

def Distr_Diff(data,x=0.1):
   incr=0
   max_range=max(data)
   X=[]
```

```
    diff=[]
    while(incr<=max_range):
        X.append(incr)
        diff.append(Fitted_Distr(incr)-Sample_Distr(data,incr))
        incr+=x
    return X,diff

def Plot_Diff(X,Y):
    plt.plot(X,Y)
    plt.axhline(0, color='green')
    plt.axhline(0.05, color='blue')
    plt.axhline(0.1, color='red')
    plt.axhline(-0.05, color='blue')
    plt.axhline(-0.1, color='red')
    plt.xlabel('x')
    plt.ylabel('Fitted F(x) - Sample F(x)')
    plt.title('Distribution Difference', fontweight='bold')
    plt.savefig('Distribution Diffrernce.png')
```

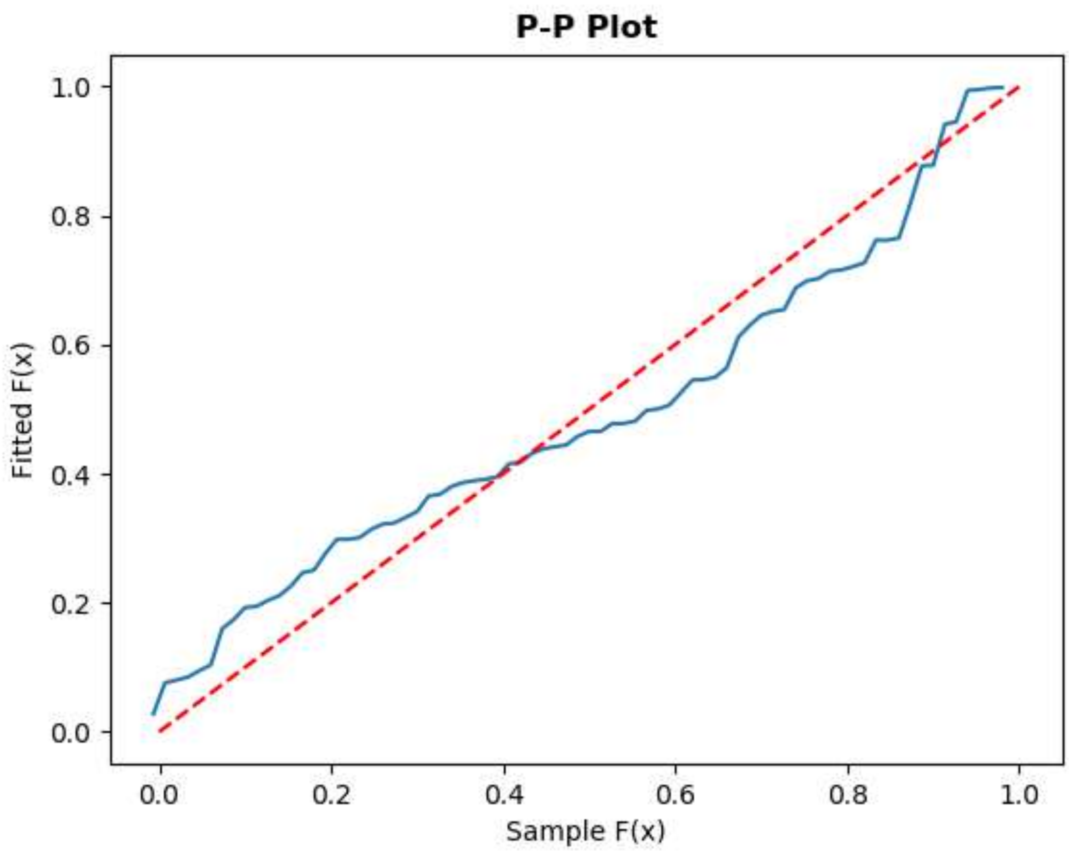
We can see the difference plot crosses blue and red lines in several cases. It does remain stable near 0.0. So we can say it is not good fitted in this fitted model due to small sample size(=75). However if sample size were large enough we would expect a good fit.

P-P and Q-Q Plot:



Here we have plotted the a graph for quantile vs sample X_s, fitted X_m to see how sample distribution and fitted distribution behave.

- P-P Plot:



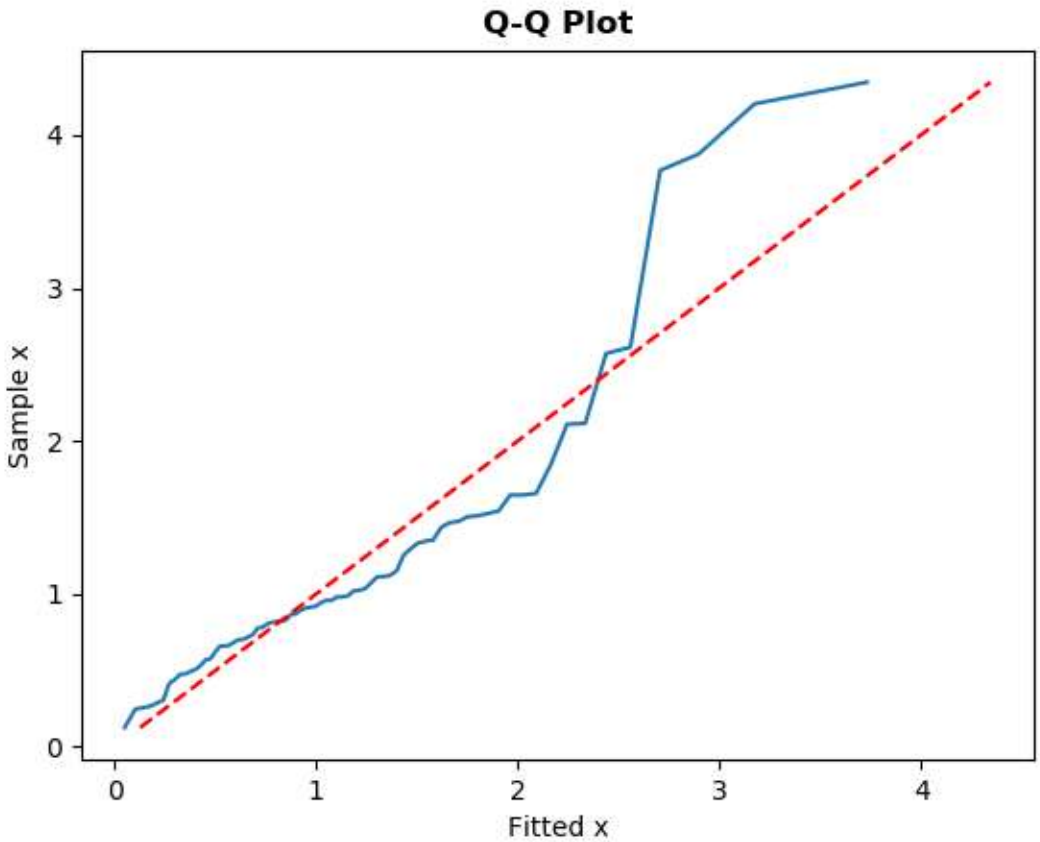

```
def Fitted_Distr(x,a=1.5278996202501096,b=1.2999425079066453):
   return (1-np.exp((-(x / b)**a)))

def Sample_Distr(data,x):
   n=len(data)
   cnt=0
   for d in data:
      if d<=x:
         cnt+=1
   return cnt/n

def Distr_Diff(data,x=0.1):
   incr=0
   max_range=max(data)
   X=[]
```

```
    Y=[]
    for i,d in enumerate(data):
        X.append((i-.5)/len(data))
        Y.append(Fitted_Distr(d))
    return X,Y

def Plot_Diff(X,Y):
    plt.plot(X,Y)
    plt.plot([0,1],[0,1],color='r',ls='--')
    plt.xlabel('Sample F(x)')
    plt.ylabel('Fitted F(x)')
    plt.title('P-P Plot', fontweight='bold')
    plt.savefig('P-P Plot.png')
```

We know P-P plot amplifies difference between the middle of the fitted $\hat{F}$(x) and sample F_n(x) but fails to amplify difference between the right tails.

Here, Sample distribution almost follows Fitted distribution with some deviation in the middle due to small dataset(=75). So, we can say this fitted model is a good representation of the sample distribution.

- <u>Q-Q Plot:</u>




```
def Fitted_Distr(y,a=1.5278996202501096,b=1.2999425079066453):
    result = b * (-(math.log(1 - y))) ** (1 / a)
    return result

def getFitted_Xq(data,qi):
    return np.percentile(data,qi)

def Distr_Diff(data,fitted_data,x=0.1):
    incr=0
    n=len(data)
    X=[]
    Y=[]
    for i,d in enumerate(data,1):
        X.append((Fitted_Distr((i-0.5)/n)))
        Y.append((d))
```

```
    return X,Y

def Plot_Diff(X,Y):
    plt.plot(X,Y)
    plt.xlabel('Fitted x')
    plt.ylabel('Sample x')

    plt.title('Q-Q Plot', fontweight='bold')
    plt.plot(Y,Y,color='r',ls='--')
    plt.savefig('Q-Q Plot.png')
```

Q-Q plot amplifies difference between the right tails of the fitted distribution $\hat{F}$(x) and our sample distribution F_n(x).

Here, Sample distribution almost follows Fitted distribution with some deviation in the right tails due to small dataset(=75). So, we can say this fitted model is a good representation of the sample distribution.

- Chi-Square Test:

```
def Inv_Distr(y,a=1.5278996202501096,b=1.2999425079066453):

    result = b * (-(math.log(1 - y))) ** (1 / a)
    return result

def Indiv_Chi(Nj,nPj):
    x=(Nj-nPj)**2
    x/=nPj
    return x

def Calc_Chi_Square(data,k):
    n=len(data)
    Pj = 1 / k
    nPj = n * Pj
    chi=0
    a0=0.0
    a1=0.0
    for i in range(1,k):
        a1=Inv_Distr(i/k)
        cnt=0
        for d in data:
            if a0<=d<a1:
                cnt+=1
        chi+=Indiv_Chi(cnt,nPj)
        a0=a1


    #last interval
    a0=a1
    a1=max(data)+0.1
    cnt = 0
    for d in data:
        if a0 <= d < a1:
            cnt += 1
    chi += Indiv_Chi(cnt, nPj)
    return chi
```

We choose No. of Intervals k=15.

So $n*P_j = 75*(1/15)$

= 5

So the conditions of equiprobable intervals k>=3 and $n*P_j$>=5 for all j is satisfied.

Simulation Result:

j: 1 Interval= [0.000000, 0.225907) nPj= 5.000000 Nj= 1 $((N_j - n P_j)^2)/n P_j$ = 3.200000

j: 2 Interval= [0.225907, 0.364165) nPj= 5.000000 Nj= 5 $((N_j - n P_j)^2)/n P_j$ = 0.000000

j: 3 Interval= [0.364165, 0.487054) nPj= 5.000000 Nj= 4 $((N_j - n P_j)^2)/n P_j$ = 0.200000

j: 4 Interval= [0.487054, 0.604179) nPj= 5.000000 Nj= 5 $((N_j - n P_j)^2)/n P_j$ = 0.000000

j: 5 Interval= [0.604179, 0.720000) nPj= 5.000000 Nj= 8 $((N_j - n P_j)^2)/n P_j$ = 1.800000

j: 6 Interval= [0.720000, 0.837511) nPj= 5.000000 Nj= 8 $((N_j - n P_j)^2)/n P_j$ = 1.800000

j: 7 Interval= [0.837511, 0.959324) nPj= 5.000000 Nj= 9 $((N_j - n P_j)^2)/n P_j$ = 3.200000

j: 8 Interval= [0.959324, 1.088220) nPj= 5.000000 Nj= 7 $((N_j - n P_j)^2)/n P_j$ = 0.800000

j: 9 Interval= [1.088220, 1.227652) nPj= 5.000000 Nj= 4 $((N_j - n P_j)^2)/n P_j$ = 0.200000

j: 10 Interval= [1.227652, 1.382473) nPj= 5.000000 Nj= 5 $((N_j - n P_j)^2)/n P_j$ = 0.000000

j: 11 Interval= [1.382473, 1.560331) nPj= 5.000000 Nj= 7 $((N_j - n P_j)^2)/n P_j$ = 0.800000 j: 12 Interval= [1.560331, 1.774970) nPj= 5.000000 Nj= 3 ((Nj-n P_j)^2)/n P_j = 0.800000

j: 13 Interval= [1.774970, 2.056158) nPj= 5.000000 Nj= 1 ((Nj-n P_j)^2)/n P_j = 3.200000

j: 14 Interval= [2.056158, 2.495141) nPj= 5.000000 Nj= 2 ((Nj-n P_j)^2)/n P_j = 1.800000

j: 15 Interval= [2.495141, 4.445000) nPj= 5.000000 Nj= 6 ((Nj-n P_j)^2)/n P_j = 0.200000

No. of Intervals k: 15

X2: 18.0

X2(15-1,1-0.05): 23.685

X2(15-1,1-0.10): 21.064

Cannot Reject the Hypothesis at alpha=0.05

Cannot Reject the Hypothesis at alpha=0.10

So, Chi-Square Test gives us no reason to conclude that our sample data is poorly fitted by Weibull(α=1.5278996202501096, β=1.2999425079066453)distribution.