Example A4.2: Data Analysis Concrete n=40

In this notebook we are evaluating the optimal distribution fitting the data. At first, we import a set of libraries

In [1]:
```
import OpenAIUQ as auq
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import pearsonr
```

Problem Definition

We have a dataset of 40 data of concrete density (Kg/m^3) and strength (MPa). We want:

1) to evaluate the statistics of the data, including their correlation;

2) to find the optimal distributions of the marginals of the data;

3) to determine the joint distribution of the data

4) generate samples and checking the model accuracy

Step 0: Import Data

First, we import the dataset represent 40 data of the couples of data collected in the file 'concrete40.dat'. d1 and d2 are vectors collecting the marginal data.

In [2]:
```
#=======================================
#CONCRETE 40
#=======================================
dataset=np.loadtxt('concrete40.dat')
print(dataset)
```

```
[[2437.    60.5]
 [2437.    60.9]
 [2425.    59.8]
 [2427.    53.4]
 [2428.    56.9]
 [2448.    67.3]
 [2456.    68.9]
 [2436.    49.9]
 [2435.    57.8]
 [2446.    60.9]
 [2441.    61.9]
 [2456.    67.2]
 [2444.    64.9]
 [2447.    63.4]
 [2433.    60.5]
 [2429.    68.1]
 [2435.    68.3]
 [2471.    65.7]
 [2472.    61.5]
 [2445.    60. ]
 [2436.    59.6]
 [2450.    60.5]
 [2454.    59.8]
 [2449.    56.7]
```

```
[2441.    57.9]
[2457.    60.2]
[2447.    55.8]
[2436.    53.2]
[2458.    61.1]
[2415.    50.7]
[2448.    59. ]
[2445.    63.3]
[2436.    52.5]
[2469.    54.6]
[2455.    56.3]
[2473.    64.9]
[2488.    69.5]
[2454.    58.9]
[2427.    54.4]
[2411.    58.8]]
```

In [3]:
```
#Define the 2 sets of data-objects
d1=auq.data1(dataset[:,0])
d2=auq.data1(dataset[:,1])
print('d1= ',d1.data)
print()
print('d2= ',d2.data)
```

```
d1=  [2437. 2437. 2425. 2427. 2428. 2448. 2456. 2436. 2435. 2446. 2441. 2456.
 2444. 2447. 2433. 2429. 2435. 2471. 2472. 2445. 2436. 2450. 2454. 2449.
 2441. 2457. 2447. 2436. 2458. 2415. 2448. 2445. 2436. 2469. 2455. 2473.
 2488. 2454. 2427. 2411.]

d2=  [60.5 60.9 59.8 53.4 56.9 67.3 68.9 49.9 57.8 60.9 61.9 67.2 64.9 63.4
 60.5 68.1 68.3 65.7 61.5 60.  59.6 60.5 59.8 56.7 57.9 60.2 55.8 53.2
 61.1 50.7 59.  63.3 52.5 54.6 56.3 64.9 69.5 58.9 54.4 58.8]
```

Step 1: Evaluate the statistics of the data d1 and d2

Variable d1: Concrete density (kg/m3)

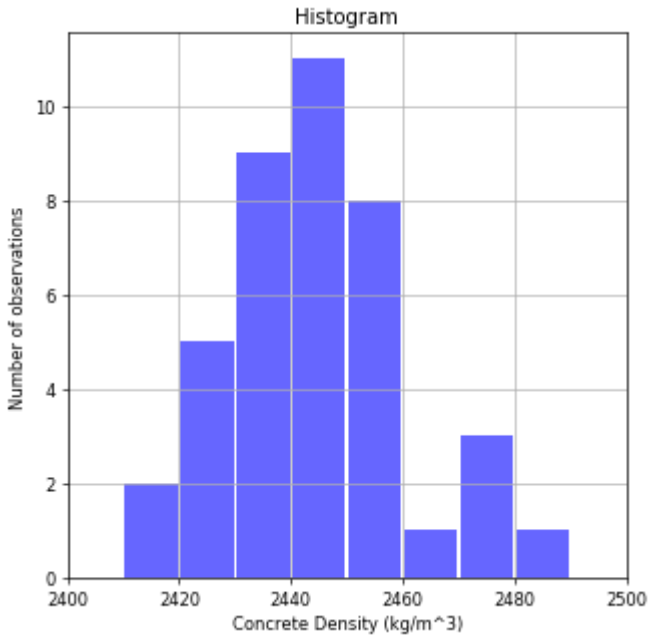
In [4]:
```
print('Data analysis')
print('min value: {:.4}'.format(d1.datamin))
print('max value: {:.4}'.format(d1.datamax))
print('range: {:.3}'.format(d1.r))
```

```
Data analysis
min value: 2.411e+03
max value: 2.488e+03
range: 77.0
```

Histogram

In [5]:
```
#Choose binning
bin1=np.arange(2400,2510,10)

fig1=plt.figure(num=1,figsize=(6,6),dpi=60)
#num: figure number
#figsize: fugure dimension
#dpi

ax=fig1.add_subplot(1,1,1)
#plot=1 row, 1 column

n,bin,patches=ax.hist(d1.data, bins=bin1, histtype='bar',facecolor='blue', density=F
#bins: limits of the bins
#facecolor: color of the bar
#density: False, True
#alpha: transparency
#rwidth: distance between histograms
```

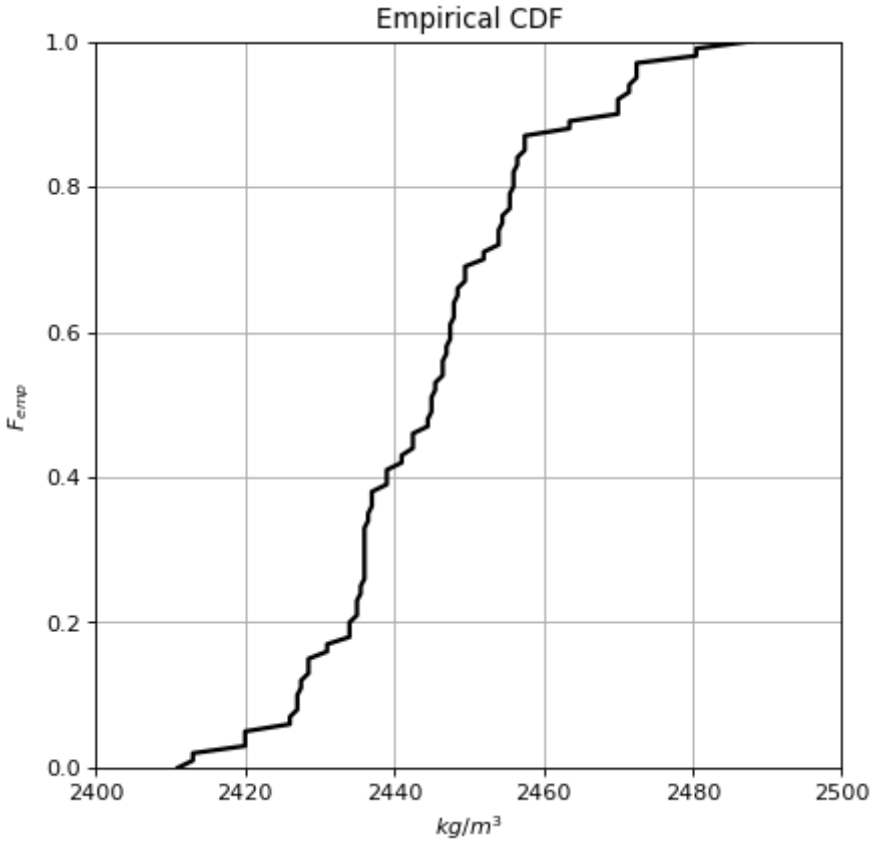
```
ax.set_title('Histogram')
ax.set_xlabel('Concrete Density (kg/m^3)')
ax.set_ylabel('Number of observations')
ax.grid()
ax.set_xlim(2400,2500)
ax.set_xticks(np.arange(2400,2520,20));
```



Cumulative Relative frequency

In [6]:

```
d1.set_range(xiniz=2400,xfin=2500)
#set the range of interest of the variable

d1.get_cdf_emp()
#evaluate the empirical cdf
#d.xemp: x-axis of the empirical cdf
#d.Femp: empirical cdf

fig2=plt.figure(num=2,figsize=(6,6),dpi=80)
ax=fig2.add_subplot(1,1,1)
ax.plot(d1.xemp,d1.Femp,'k',lw=2)
ax.grid(True)
ax.set_title('Empirical CDF')
ax.set_xlabel('$kg/m^3$')
ax.set_ylabel('$F_{emp}$')
ax.set_xlim(2400,2500)
ax.set_ylim(0,1);
#ax.set_xticks([2400,2420,2440,2460,2500]);
#ax.set_yticks([0,0.2,1]);

#d1.plot_cdf_emp(fignum=2,figsize=(6,6),figdpi=60)
#plot the empirical cdf
```



Metrics

In [7]:
```
print('Data analysis')
print('Mean: {:.4} MPa'.format(d1.mean))
print('Standard deviation: {:.4} MPa'.format(d1.std))
print('Coefficient of variation: {:.4}'.format(d1.cov))
print('Skewness: {:.4}'.format(d1.g1))
print('Kurtosis: {:.4}'.format(d1.g2))
```

```
Data analysis
Mean: 2.445e+03 MPa
Standard deviation: 15.79 MPa
Coefficient of variation: 0.00646
Skewness: 0.3892
Kurtosis: 3.302
```

The coefficient of variation ν shows a very low value of spreading around the mean value.

The low skewness γ_1 shows a tail dominant on the right. But the number is data is small to have a reliable estimate of skewness.

The kurtosis value γ_2 shows that the tails are higher than the Gaussian. But the number is data is too small to have a reliable estimate of kurtosis.

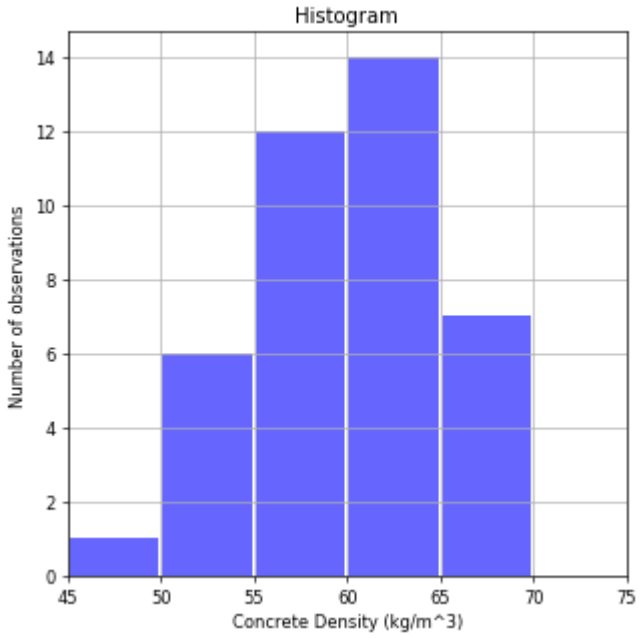
Variable d2: Concrete strength (MPa)

In [8]:
```
print('Data analysis')
print('min value: {:.3}'.format(d2.datamin))
print('max value: {:.3}'.format(d2.datamax))
print('range: {:.3}'.format(d2.r))
```

```
Data analysis
min value: 49.9
max value: 69.5
range: 19.6
```

Histogram

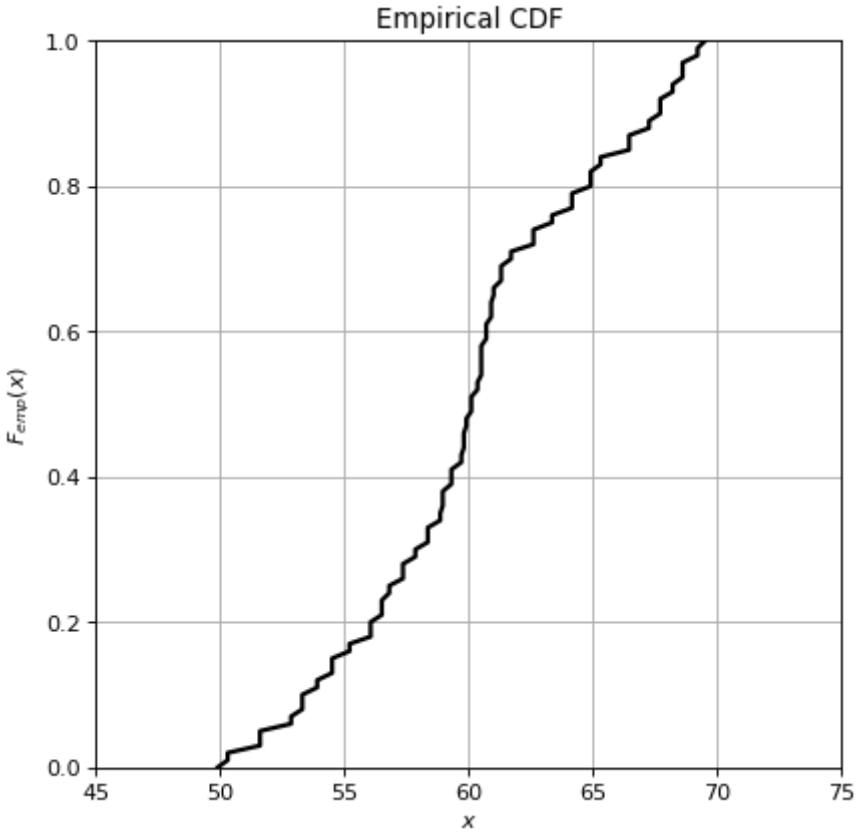
In [9]:
```
#Choose binning
bin1=np.arange(45,80,5)

fig3=plt.figure(num=3,figsize=(6,6),dpi=60)
#num: figure number
#figsize: fugure dimension
#dpi

ax=fig3.add_subplot(1,1,1)
#plot=1 row, 1 column

n,bin,patches=ax.hist(d2.data, bins=bin1, histtype='bar',facecolor='blue', density=F
#bins: limits of the bins
#facecolor: color of the bar
#density: False, True
#alpha: transparency
#rwidth: distance between histograms

ax.set_title('Histogram')
ax.set_xlabel('Concrete Density (kg/m^3)')
ax.set_ylabel('Number of observations')
ax.grid(True)
ax.set_xlim(45,75)
ax.set_xticks(bin);
```



Cumulative Relative frequency

In [10]:
```
d2.set_range(xiniz=45,xfin=75)
#set the range of interest of the variable

d2.get_cdf_emp()
#evaluate the empirical cdf

#fig=plt.figure(num=4,figsize=(6,6),dpi=80)
#ax=fig.add_subplot(1,1,1)
#ax.plot(d2.xemp,d2.Femp,'k',lw=2)
#ax.grid(True)
#ax.set_title('Empirical CDF')
#ax.set_xlabel('$N/mm^2$')
#ax.set_ylabel('$F_{emp}$')
#ax.set_xlim(45,75)
#ax.set_ylim(0,1);
```

```
#ax.set_xticks([45,50,55,60,65,70,75]);
##ax.set_yticks([0,0.2,1]);

d2.plot_cdf_emp(fignum=4,figsize=(6,6),figdpi=80)
#plot the empirical cdf
```



Metrics

In [11]:
```
print('Data analysis')
print('Mean: {:.3} MPa'.format(d2.mean))
print('Standard deviation: {:.3} MPa'.format(d2.std))
print('Coefficient of variation: {:.3}'.format(d2.cov))
print('Skewness: {:.3}'.format(d2.g1))
print('Kurtosis: {:.3}'.format(d2.g2))
```

```
Data analysis
Mean: 60.1 MPa
Standard deviation: 4.95 MPa
Coefficient of variation: 0.0823
Skewness: 0.026
Kurtosis: 2.45
```

The coefficient of variation ν shows a low value of spreading around the mean value. This is a typical value for concrete

The low skewness γ_1 shows high degree of symmetry of the data around the mean value. But the number is data is small to have a reliable estimate of skewness.

The kurtosis value γ_2 shows that the tails are lower than the Gaussian. But the number is data is too small to have a reliable estimate of kurtosis.

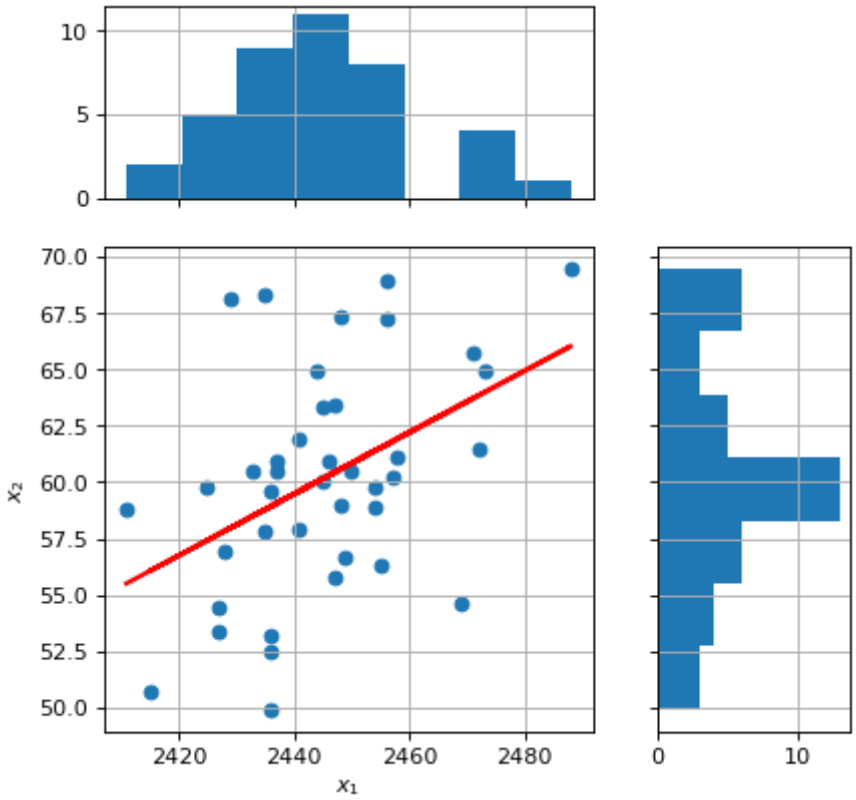
Sample Correlation and Scatter Plot

In [12]:
```
D=auq.data2(dataset)
```

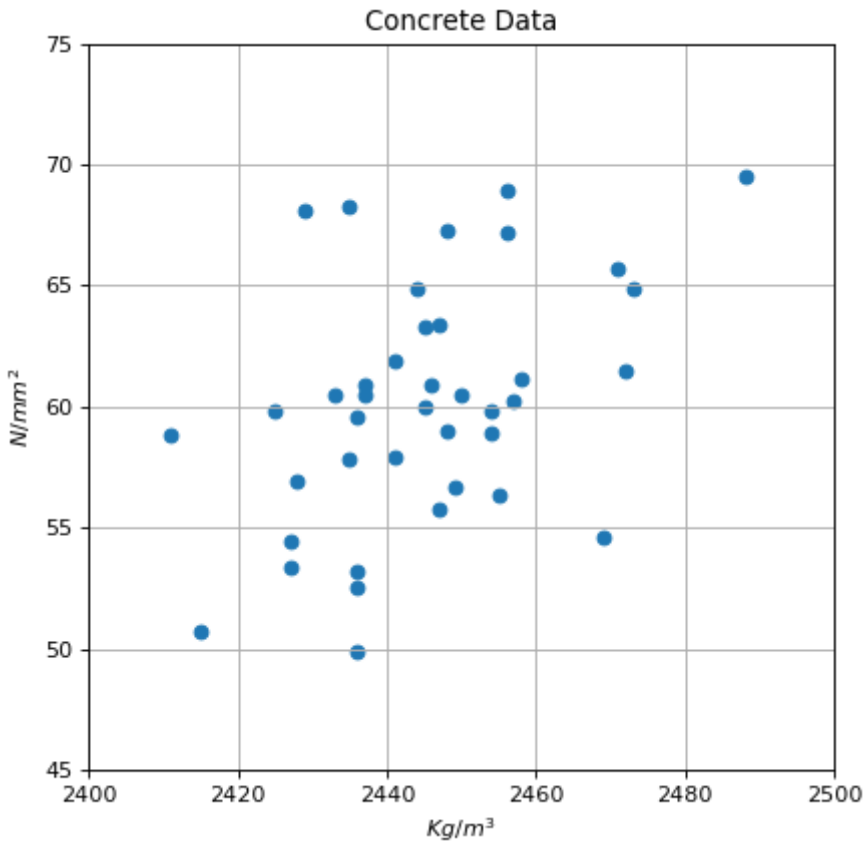
In [13]:
```
D.get_corr()
print(D.R)
```

```
[[1.         0.43642728]
 [0.43642728 1.        ]]
```

In [14]:
```
D.plot(numfig=5,figsize=(6,6), dpi=80, sec=[1,2], hist='yes',regression='yes')
#sec=[1,2] variables to be considered, in the example: d1,d2
#hist='yes','no': scatter plot with histograms
#regression='yes','no': plot the linear regression over the points
```



In [15]:
```
fig6=plt.figure(num=6,figsize=(6,6),dpi=80)
ax=fig6.add_subplot(1,1,1)
ax.scatter(d1.data,d2.data)
ax.grid(True)
ax.set_title('Concrete Data')
ax.set_xlabel('$Kg/m^3$')
ax.set_ylabel('$N/mm^2$')
ax.set_xlim(2400,2500)
ax.set_ylim(45,75);
ax.set_xticks([2400,2420,2440,2460,2480,2500]);
##ax.set_yticks([0,0.2,1]);
```



Step 2: Probability distributions of the marginals

x1: Concrete Density

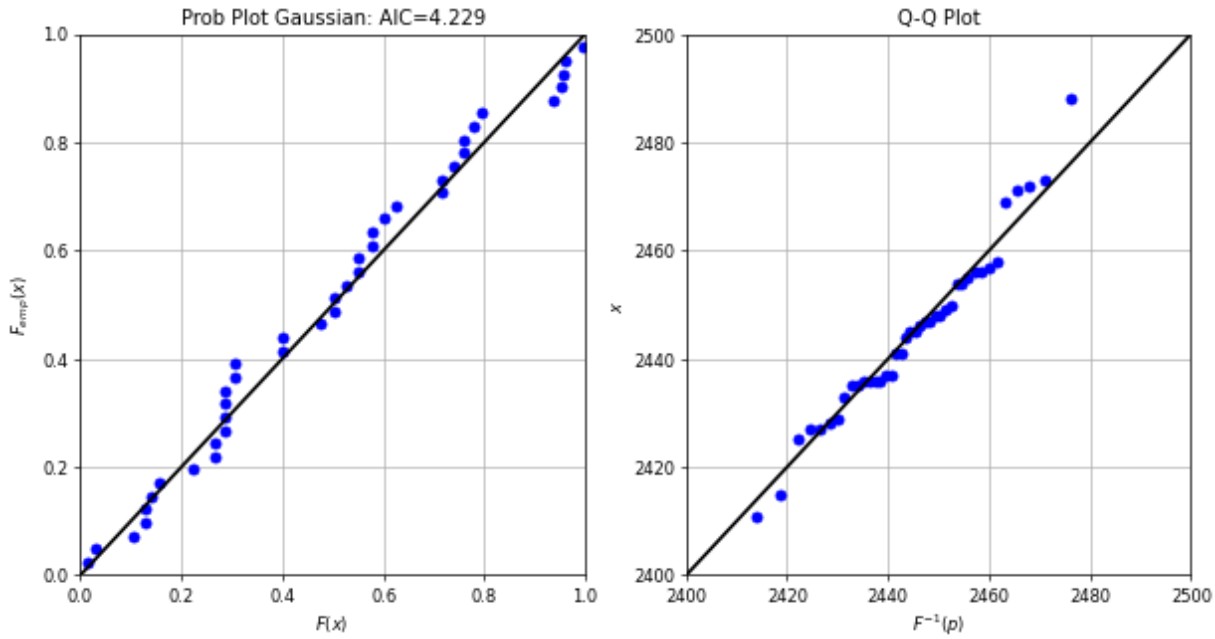
Let's find now a distribution model fitting well the data. We explore

- Gaussian
- Lognormal
- Weibull

Gaussian fit

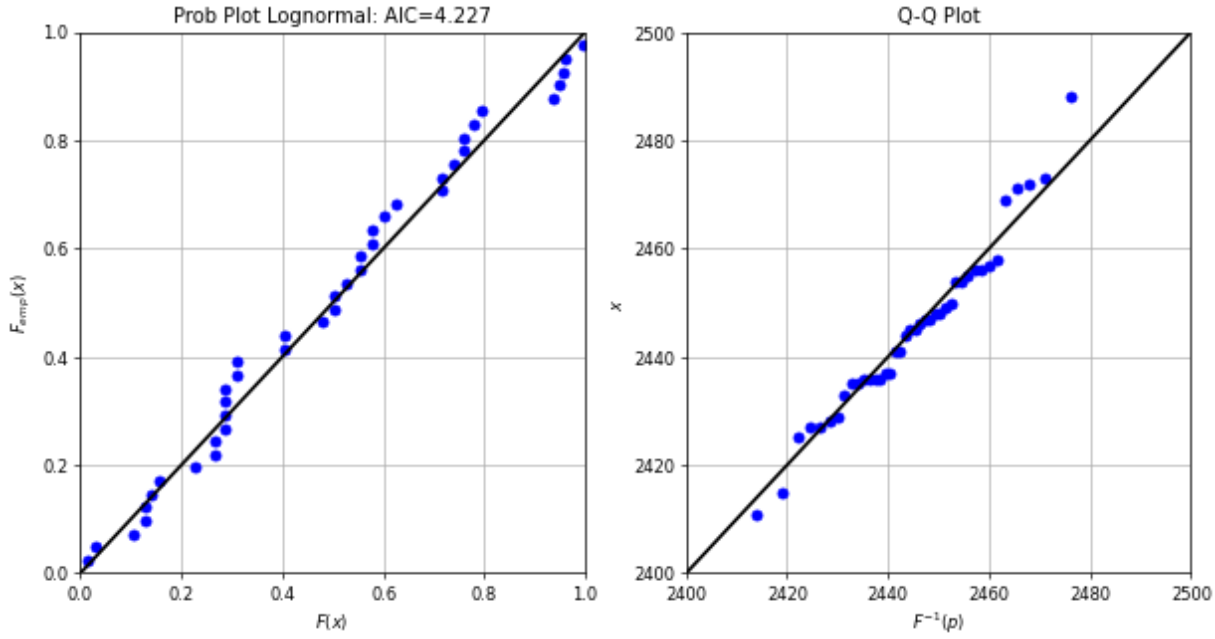
In [16]:
```
x1_a1=auq.dist('Gaussian')
#Gaussian, Lognormal, Weibull, Uniform

x1_a1.set_range(xiniz=2400,xfin=2500,size=100)
#xiniz: Lower bound
#xfin: Upper bound
#size: number of points

x1_a1.MLEfit(d1.data,ML='yes')
#Evaluate the parameters of x through MLE
```

In [17]:
```
fig11=auq.distplot(num=11,figsize=(12,6),dpi=60)
#fig is the object associated to the plot

fig11.prob_plot_w2(d1.data,x1_a1)
#include prob plot and q-q plot
```



Lognormal fit

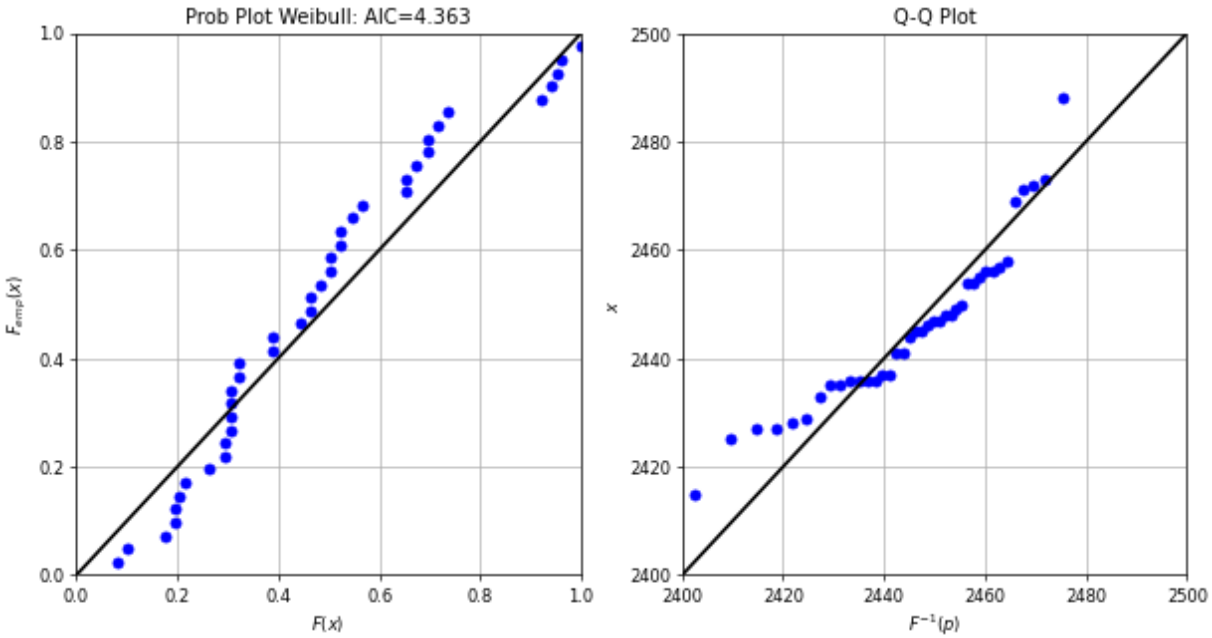
In [18]:
```
x1_a2=auq.dist('Lognormal')
#Gaussian, Lognormal, Weibull, Uniform

x1_a2.set_range(xiniz=2400,xfin=2500,size=100)
#xiniz: lower bound
#xfin: Upper bound
#size: number of points

x1_a2.MLEfit(d1.data,ML='yes')
#Evaluate the parameters of x through MLE

fig12=auq.distplot(num=12,figsize=(12,6),dpi=60)
#fig is the object associated to the distribution plot
fig12.prob_plot_w2(d1.data,x1_a2)
```



Weibull fit

In [19]:
```
x1_a3=auq.dist('Weibull')
#Gaussian, Lognormal, Weibull, Uniform
```

```
x1_a3.set_range(xiniz=2400,xfin=2500,size=100)
#xiniz: Lower bound
#xfin: Upper bound
#size: number of points

x1_a3.MLEfit(d1.data,ML='yes')
#Evaluate the parameters of x through MLE

fig13=auq.distplot(num=13,figsize=(12,6),dpi=60)
#fig is the object associated to the distribution plot
fig13.prob_plot_w2(d1.data,x1_a3)
```



Summary variable x1

The comparison of the AIC of the three distribution is:

In [20]:
```
print('Model Selection AIC')
print('Gaussian: {:.4}'.format(x1_a1.AIC))
print('Lognormal: {:.4}'.format(x1_a2.AIC))
print('Weibull: {:.4}'.format(x1_a3.AIC))
```

```
Model Selection AIC
Gaussian: 4.229
Lognormal: 4.227
Weibull: 4.363
```

Gaussian and Lognormal have approximately the same AIC.

This is also confirmed by the Q-Q plot.

Machine Learning Model Selection

In [21]:
```
print('Model Selection Machine Learning')
print('----------------------------------')
print('Gaussian: {:.4}'.format(x1_a1.DIV_tr))
print('Lognormal: {:.4}'.format(x1_a2.DIV_tr))
print('Weibull: {:.4}'.format(x1_a3.DIV_tr))
print('----------------------------------')
print('Gaussian: {:.4}'.format(x1_a1.DIV_ts))
print('Lognormal: {:.4}'.format(x1_a2.DIV_ts))
print('Weibull: {:.4}'.format(x1_a3.DIV_ts))
```

```
Model Selection Machine Learning
----------------------------------
Gaussian: 3.885
```

```
Lognormal: 3.884
Weibull: 4.043
---------------------------------
Gaussian: 5.239
Lognormal: 5.237
Weibull: 5.668
```

Distribution of x1

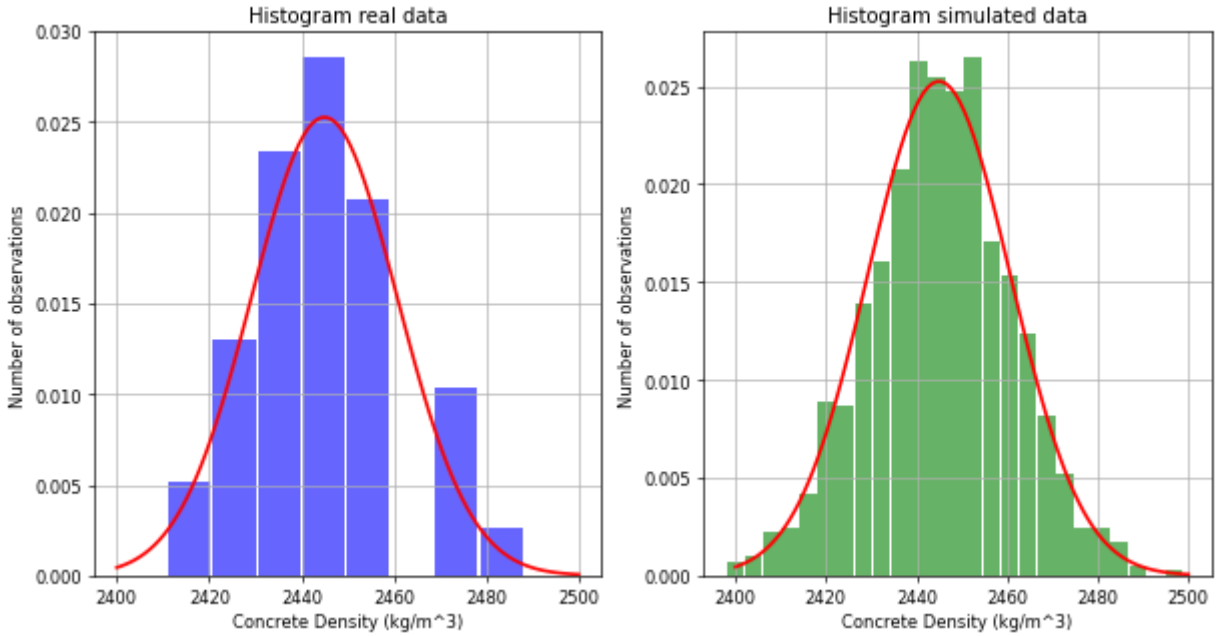
In [22]:
```python
x1=auq.dist('Gaussian')
#Gaussian, Lognormal, Weibull, Uniform

x1.set_range(xiniz=2400,xfin=2500,size=100)
#xiniz: lower bound
#xfin: Upper bound
#size: number of points

x1.MLEfit(d1.data,ML='yes')
#Evaluate the parameters of x through MLE
```

In [23]:
```python
#Generate samples
x1.gen_samples(num=1000)
#x1.samples
```

In [24]:
```python
#Evaluate pdf and cdf
x1.pdf()
x1.cdf()
```

In [25]:
```python
#Choose binning
bin1='auto'

fig301=plt.figure(num=301,figsize=(12,6),dpi=60)
#num: figure number
#figsize: fugure dimension
#dpi

ax1=fig301.add_subplot(1,2,1)
ax2=fig301.add_subplot(1,2,2)
#plot=1 row, 2 columns

n,bin,patches=ax1.hist(d1.data, bins=bin1, histtype='bar',facecolor='blue', density=
#bins: limits of the bins
#facecolor: color of the bar
#density: False, True
#alpha: transparency
#rwidth: distance between histograms
ax1.plot(x1.xx,x1.f,'r',lw=2)

ax1.set_title('Histogram real data')
ax1.set_xlabel('Concrete Density (kg/m^3)')
ax1.set_ylabel('Number of observations')
ax1.grid(True)

#================================================
n,bin,patches=ax2.hist(x1.samples, bins=bin1, histtype='bar',facecolor='green', dens
#bins: limits of the bins
#facecolor: color of the bar
#density: False, True
#alpha: transparency
#rwidth: distance between histograms
ax2.plot(x1.xx,x1.f,'r',lw=2)

ax2.set_title('Histogram simulated data')
ax2.set_xlabel('Concrete Density (kg/m^3)')
```

```
ax2.set_ylabel('Number of observations')
ax2.grid(True)
```



x2: Concrete Strength

Let's find now a distribution model fitting well the data. We explore

- Gaussian
- Lognormal
- Weibull

In [26]:
```
x2_a1=auq.dist('Gaussian')
#Gaussian, Lognormal, Weibull, Uniform

x2_a1.set_range(xiniz=45,xfin=75,size=100)
#xiniz: Lower bound
#xfin: Upper bound
#size: number of points

x2_a1.MLEfit(d2.data,ML='yes')
#Evaluate the parameters of x through MLE
```

In [27]:
```
x2_a2=auq.dist('Lognormal')
#Gaussian, Lognormal, Weibull, Uniform

x2_a2.set_range(xiniz=45,xfin=75,size=100)
#xiniz: Lower bound
#xfin: Upper bound
#size: number of points

x2_a2.MLEfit(d2.data,ML='yes')
#Evaluate the parameters of x through MLE
```

In [28]:
```
x2_a3=auq.dist('Weibull')
#Gaussian, Lognormal, Weibull, Uniform

x2_a3.set_range(xiniz=45,xfin=75,size=100)
#xiniz: Lower bound
#xfin: Upper bound
#size: number of points
```

```
x2_a3.MLEfit(d2.data,ML='yes')
#Evaluate the parameters of x through MLE
```

In [29]:
```
print('Model Selection AIC')
print('--------------------')
print('Gaussian: {:.4}'.format(x2_a1.AIC))
print('Lognormal: {:.4}'.format(x2_a2.AIC))
print('Weibull: {:.4}'.format(x2_a3.AIC))
```

```
Model Selection AIC
--------------------
Gaussian: 3.069
Lognormal: 3.071
Weibull: 3.103
```

In [30]:
```
print('Model Selection Machine Learning')
print('---------------------------------')
print('Gaussian: {:.4}'.format(x2_a1.DIV_tr))
print('Lognormal: {:.4}'.format(x2_a2.DIV_tr))
print('Weibull: {:.4}'.format(x2_a3.DIV_tr))
print('---------------------------------')
print('Gaussian: {:.4}'.format(x2_a1.DIV_ts))
print('Lognormal: {:.4}'.format(x2_a2.DIV_ts))
print('Weibull: {:.4}'.format(x2_a3.DIV_ts))
```

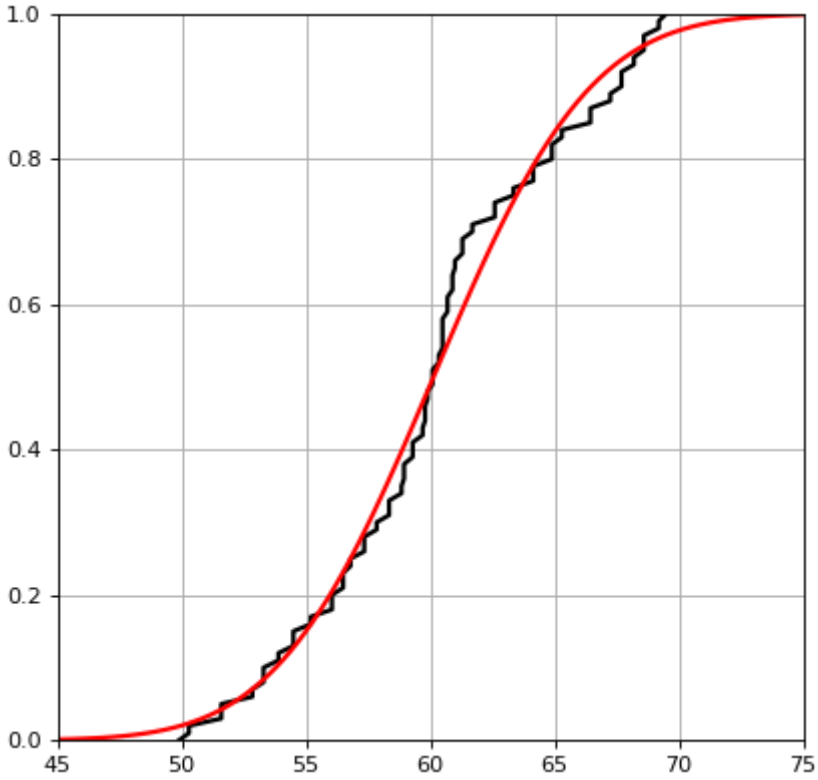
```
Model Selection Machine Learning
---------------------------------
Gaussian: 2.966
Lognormal: 2.974
Weibull: 2.989
---------------------------------
Gaussian: 3.181
Lognormal: 3.174
Weibull: 3.231
```

Distribution of x2

In [31]:
```
x2=auq.dist('Gaussian')
#Gaussian, Lognormal, Weibull, Uniform

x2.set_range(xiniz=45,xfin=75,size=100)
#xiniz: lower bound
#xfin: Upper bound
#size: number of points

x2.MLEfit(d2.data,ML='yes')
#Evaluate the parameters of x through MLE
```

In [32]:
```
#Evaluate pdf and cdf
x2.pdf()
x2.cdf()
```

In [33]:
```
fig=plt.figure(num=301,figsize=(6,6),dpi=80)
ax=fig.add_subplot(1,1,1)
ax.plot(d2.xemp,d2.Femp,'k',lw=2)
#ax.plot(x1_sim.xemp,x1_sim.Femp,'b',lw=2)
ax.plot(x2.xx,x2.F,'r',lw=2)
ax.grid(True)
ax.set_xlim(45,75)
ax.set_ylim(0,1);
#ax.plot(x1.xx,x1.F,'r',lw=4)
```



Step 3:Joint Distribution

In [34]:
```
D.R
```

Out[34]:
```
array([[1.        , 0.43642728],
       [0.43642728, 1.        ]])
```

In [35]:
```
#Include in x all the basic random variables
x=[x1,x2]

#Define the matrix of correlation
corr=[[1,D.R[0,1]],[D.R[0,1],1]]
corr=np.array(corr)

#Define the multivariate distribution model: 'Independent','Gaussian','Nataf'
X=auq.dist2(x,corr,'Gaussian')
```

In [36]:
```
X.Sigma
```

Out[36]:
```
array([[249.419375  ,  34.1303125 ],
       [ 34.1303125 ,  24.52034375]])
```

Simulate samples

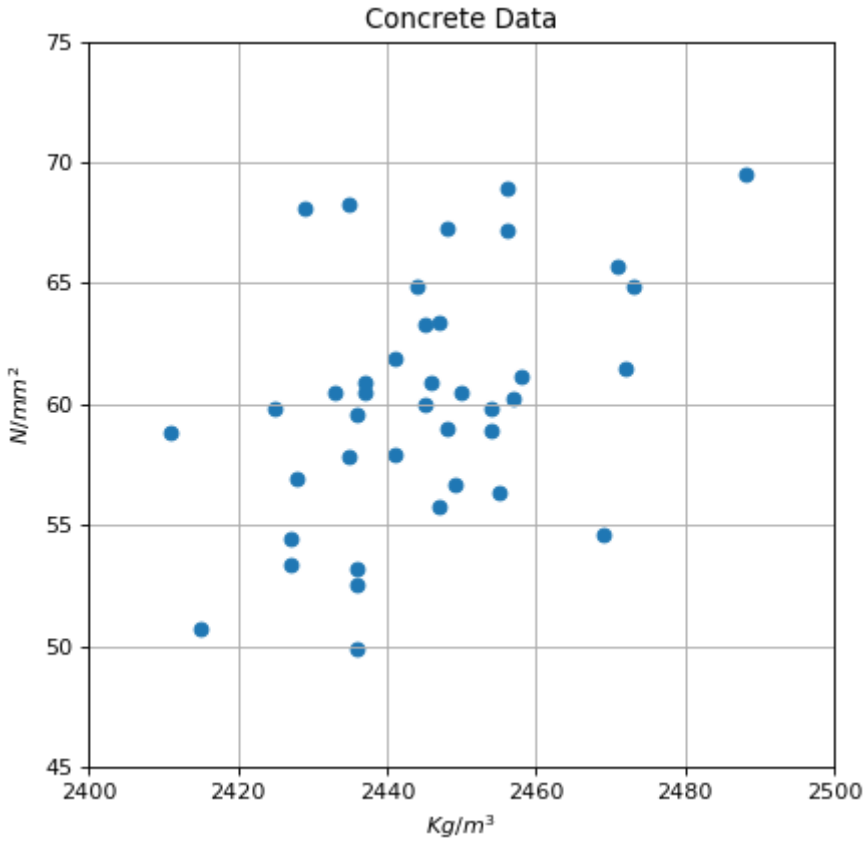
In [37]:
```
#to check the joint distribution, generate samples (num=number of samples)
X.gen_samples(num=1000) #num=number of samples
```

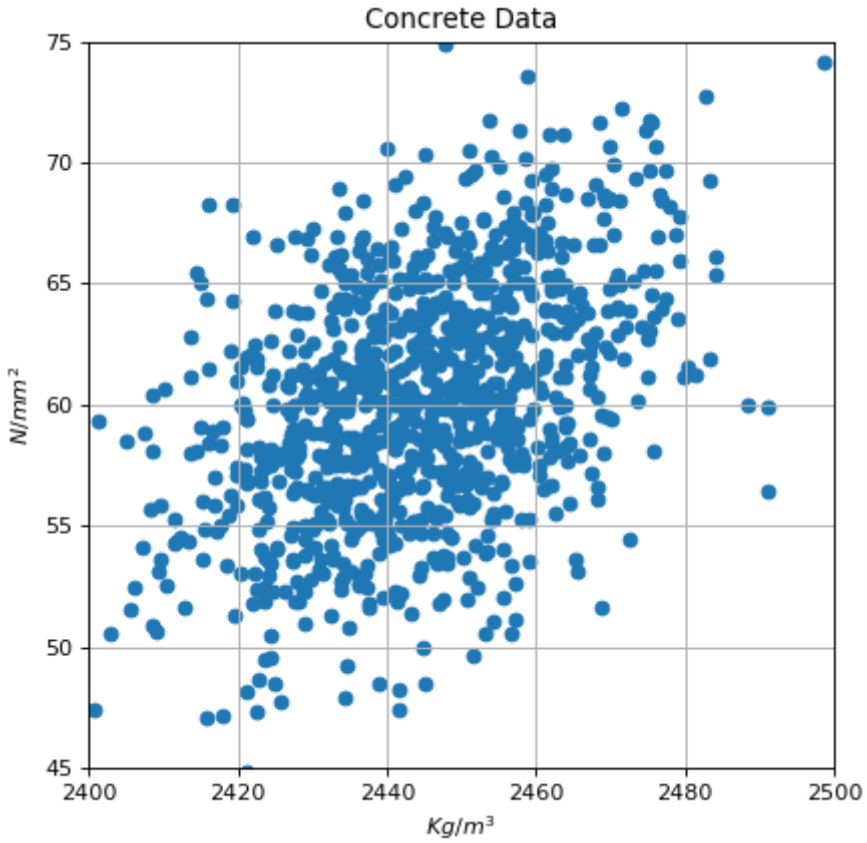
In [38]:
```
X.samples
```

Out[38]:
```
array([[2434.0108043 ,   60.31923675],
       [2438.77169964,   65.78531862],
       [2454.15243203,   55.25022909],
       ...,
       [2452.17152767,   61.39293238],
       [2446.02036148,   59.6281979 ],
       [2469.88916637,   64.81012809]])
```

Plot simulated samples

In [39]:
```
fig501=plt.figure(num=501,figsize=(6,6),dpi=80)
ax=fig501.add_subplot(1,1,1)
ax.scatter(d1.data,d2.data)
ax.grid(True)
ax.set_title('Concrete Data')
ax.set_xlabel('$Kg/m^3$')
ax.set_ylabel('$N/mm^2$')
ax.set_xlim(2400,2500)
ax.set_ylim(45,75);
ax.set_xticks([2400,2420,2440,2460,2480,2500]);
##ax.set_yticks([0,0.2,1]);

fig502=plt.figure(num=502,figsize=(6,6),dpi=80)
ax=fig502.add_subplot(1,1,1)
ax.scatter(X.samples[:,0],X.samples[:,1])
ax.grid(True)
ax.set_title('Concrete Data')
ax.set_xlabel('$Kg/m^3$')
ax.set_ylabel('$N/mm^2$')
ax.set_xlim(2400,2500)
ax.set_ylim(45,75);
ax.set_xticks([2400,2420,2440,2460,2480,2500]);
##ax.set_yticks([0,0.2,1]);
```





In [40]:
```
Dsim=auq.data2(X.samples)
Dsim.get_corr()
Dsim.R
```

Out[40]:
```
array([[1.        , 0.44225128],
       [0.44225128, 1.        ]])
```

In [41]:
```
D.R
```

Out[41]:
```
array([[1.        , 0.43642728],
       [0.43642728, 1.        ]])
```

In [ ]: