Probabilistic Numerical Methods

Homework-1

I) 1. Random Number Generator

In [1]:
```
# Simulating some realizations of random generator
# Generating 20 random numbers between 10 and 30
import random as rnd
random_sample_1 = []
for i in range(20):
  n = rnd.uniform(10,30)
  random_sample_1.append(n)
random_sample_1
```

Out[1]:
```
[21.906249882596285,
 20.50614376571922,
 10.30812841208671,
 25.358483490923447,
 25.466759320694788,
 24.318258861177448,
 17.26787971853736,
 25.147708532945963,
 10.619646059484896,
 20.97706545356424,
 11.78356251570115,
 15.014885056099029,
 18.582971534106175,
 15.660003934227314,
 15.277905583402704,
 16.698574994762804,
 17.245228442399302,
 12.463812329904837,
 28.81244534090522,
 23.104011336247062]
```

In [2]:
```
# Again simulating different set of random numbers using random generator
# Generating 20 random numbers between 10 and 30
random_sample_2 = []
for i in range(20):
  n = rnd.uniform(10,30)
  random_sample_2.append(n)
random_sample_2
```

Out[2]:
```
[21.518853324986324,
 11.473937046401161,
 19.508119866657847,
 29.488707503223715,
 10.594972581309413,
 15.254661499851164,
 19.854759470710732,
 17.923050904945562,
 12.59708104005796,
 23.088790414631255,
 11.260005038799765,
 23.52018344488799,
```

```
 19.581200649812544,
 12.652498431353402,
 15.834714826259972,
 22.90710165543807,
 10.902486970418996,
 12.57367387376744,
 18.510001955443183,
 28.80827181946009]
```

> **COMMENT-** It is evident that there is no co-relation between above generated two sample sets (random_sample_1, random_sample_2) and they appear to be completely random.

In [3]:

```python
# Simulating some realizations of random generator by fixing the seed value as 1000
# Generating 20 random numbers between 10 and 30
random_sample_3 = []
rnd.seed(1000)
for i in range(20):
  n = rnd.uniform(10,30)
  random_sample_3.append(n)
random_sample_3
```

Out[3]:

```
[25.54713285401128,
 23.396511191184995,
 11.98279207849634,
 17.059410223802907,
 19.35815485801684,
 20.69367482941755,
 29.566181218247948,
 12.60630700317318,
 23.424869364605325,
 17.284588318947513,
 19.776714143239715,
 14.060244214681074,
 23.323967511427185,
 14.553260624138641,
 19.161281165935264,
 10.81444795109915,
 29.48579590755657,
 19.749521485378132,
 19.232277272747194,
 24.282943116164006]
```

In [4]:

```python
# Simulating again some realizations of random generator by fixing the seed value as 10
# Generating 20 random numbers between 10 and 30
random_sample_4 = []
rnd.seed(1000)
for i in range(20):
  n = rnd.uniform(10,30)
  random_sample_4.append(n)
random_sample_4
```

Out[4]:

```
[25.54713285401128,
 23.396511191184995,
 11.98279207849634,
 17.059410223802907,
 19.35815485801684,
 20.69367482941755,
 29.566181218247948,
```

```
 12.60630700317318,
 23.424869364605325,
 17.284588318947513,
 19.776714143239715,
 14.060244214681074,
 23.323967511427185,
 14.553260624138641,
 19.161281165935264,
 10.81444795109915,
 29.48579590755657,
 19.749521485378132,
 19.232277272747194,
 24.282943116164006]
```

In [5]:

```
# Simulating some realizations of random generator by fixing the seed value as 2000
# Generating 20 random numbers between 10 and 30
random_sample_5 = []
rnd.seed(2000)
for i in range(20):
  n = rnd.uniform(10,30)
  random_sample_5.append(n)
random_sample_5
```

Out[5]:

```
[18.96914035821057,
 28.94327428468175,
 19.489466540151703,
 25.36465955298137,
 25.81804569999036,
 28.682023115766356,
 15.081154660056557,
 25.918416854846306,
 21.49232680153613,
 12.779539939714255,
 25.107094854101213,
 13.757990423017867,
 26.39688069394593,
 12.920816024311923,
 23.156436177640764,
 24.2715193832855,
 16.876339096231685,
 18.470473413811263,
 29.308699523064156,
 14.127877463713634]
```

> **COMMENT-** From above three samples it is evident that Random Number Generator generates the same set of random numbers if we use same seed value and different set of random numbers for different seed values.

I) 2. Quality of Random Number Generator

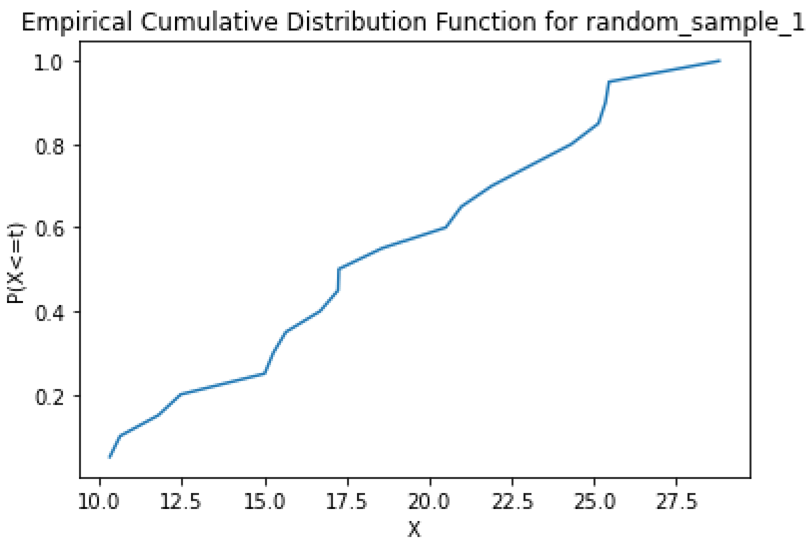
i) Plotting the Empirical Cumulative Distribution Function for random_sample_1

In [6]:

```
import numpy as np
from statsmodels.distributions.empirical_distribution import ECDF
from matplotlib import pyplot
ecdf = ECDF(random_sample_1)
# Printing intermediate cdf's
print('P(x<=15): %.3f' % ecdf(15))
```

```
print('P(x<=20): %.3f' % ecdf(20))
print('P(x<=25): %.3f' % ecdf(25))
print('P(x<=28): %.3f' % ecdf(28))
pyplot.plot(ecdf.x, ecdf.y)
pyplot.xlabel('X')
pyplot.ylabel('P(X<=t)')
pyplot.title("Empirical Cumulative Distribution Function for random_sample_1")
pyplot.show()
```

```
P(x<=15): 0.200
P(x<=20): 0.550
P(x<=25): 0.800
P(x<=28): 0.950
```



ii) Plotting the Inverse Empirical Cumulative Distribution Function for random_sample_1

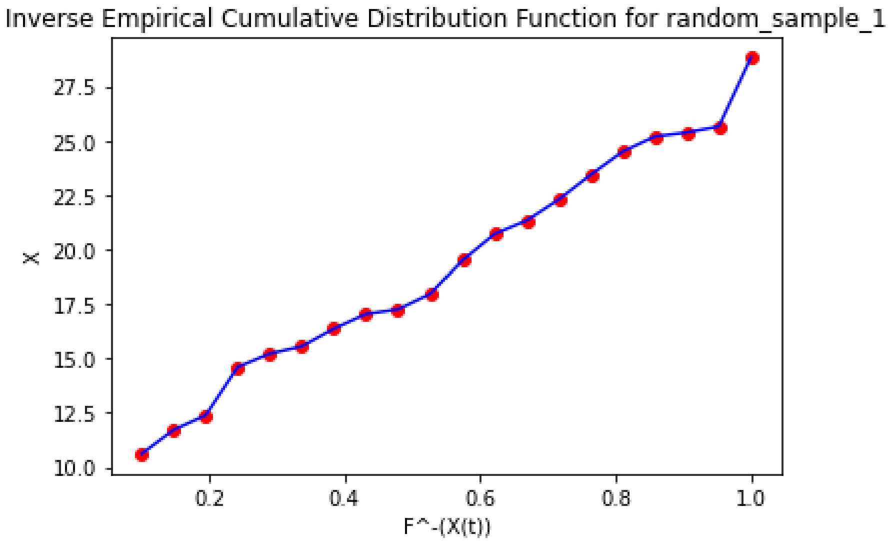
In [7]:

```
from scipy.interpolate import interp1d
# Ordering the Sample
Ordered_Sample = sorted(set(random_sample_1))
# Assigning Cumulative Probability Distribution to the Ordered Sample (P<=X)
Ecdf_Value = [ ecdf(item) for item in Ordered_Sample]
inverted_edf = interp1d(Ecdf_Value, Ordered_Sample)
x = np.linspace(0.1, 1, num=20)
y = inverted_edf(x)
# Finding the respective value of random number for given percentile of cdf for compari
print ('30 percentile:', inverted_edf(0.3))
print ('60 percentile:', inverted_edf(0.6))
print ('75 percentile:', inverted_edf(0.75))
print ('80 percentile:', inverted_edf(0.8))
pyplot.plot(x, y, 'ro', x, y, 'b-')
pyplot.xlabel("F^-(X(t))")
pyplot.ylabel('X')
pyplot.title("Inverse Empirical Cumulative Distribution Function for random_sample_1")
```

```
30 percentile: 15.277905583402704
60 percentile: 20.50614376571922
75 percentile: 23.104011336247062
80 percentile: 24.31825886117745
```

Out[7]:

```
Text(0.5, 1.0, 'Inverse Empirical Cumulative Distribution Function for random_sample_1')
```

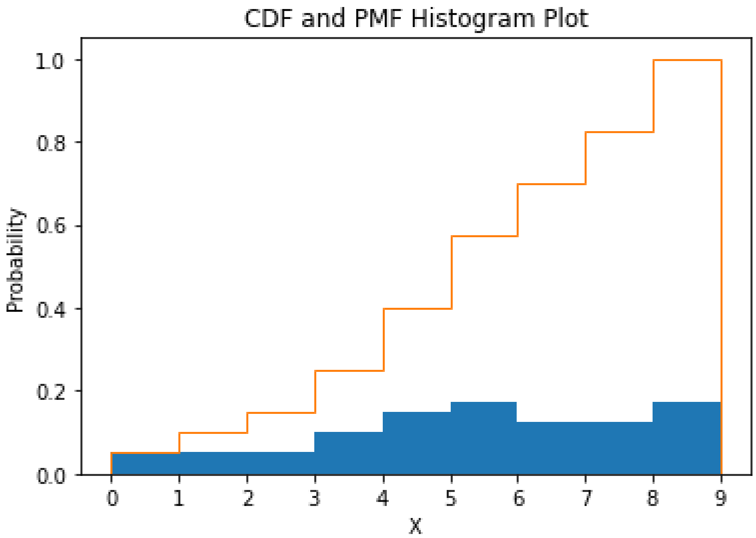


> **COMMENT** It appears from the plots that both the functions are satisfying the properties of a CDF and Inverse CDF functions like Ranging from 0 to 1, Non-Decreasing and Right Continuous. Also the corresponding values "P(x<=15): 0.150, P(x<=20): 0.600, P(x<=25): 0.850, P(x<=28): 1.000" seems to match with there inverse "30 percentile: 16.881795822402125, 60 percentile: 19.974238562412452, 75 percentile: 21.660666306034074, 80 percentile: 22.890655546445117" approximately.

II) Simulating Discrete Random Variable

Here we take a random sample of a Discrete Integer Random Variable ranging from 0 to 9 and plot it's PMF and CDF

In [8]:
```
data = [3,4,2,3,4,5,4,7,8,5,4,6,2,1,0,9,7,6,6,5,4,1,3,5,7,6,8,9,0,5,7,8,5,6,4,3,7,8,5,9
pyplot.hist(data,bins=9,density=True)
pyplot.hist(data,bins=9, density=True, cumulative=True, histtype='step')
pyplot.xlabel("X")
pyplot.ylabel("Probability")
pyplot.xticks(np.arange(0,10))
pyplot.title("CDF and PMF Histogram Plot")
pyplot.show()
```



> **COMMENT** Like mentioned previously the CDF maintains all the properties and the Histogram Plot is like a step function which is typical for a Discrete Random Variable