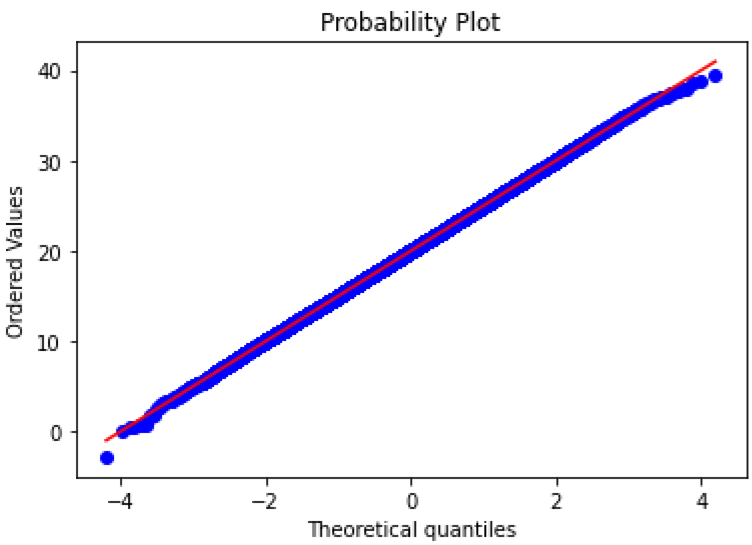
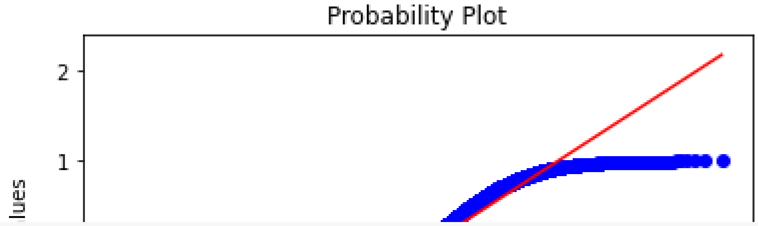
```
import numpy as np
import pylab
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
import random
from sklearn import datasets
from sklearn.utils import resample
from sklearn.metrics import accuracy_score
```

QQ Plot

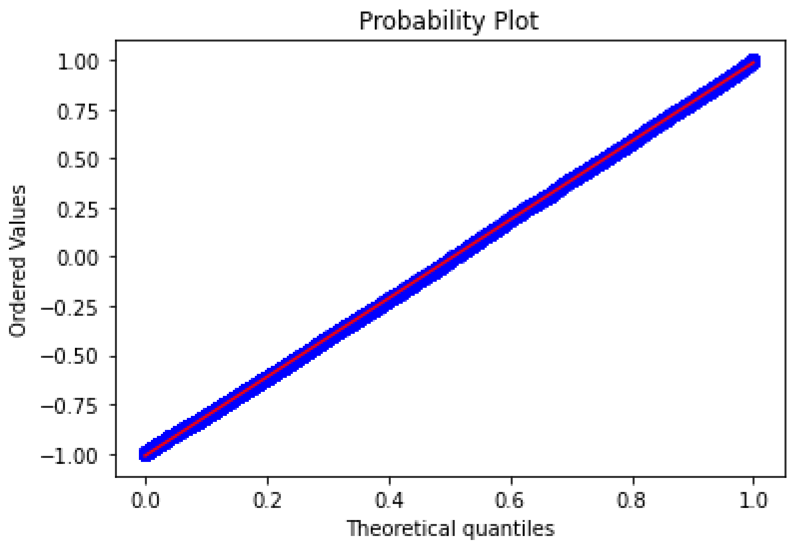
```
# generate 100 sanples from N(20,5)
measurements = np.random.normal(loc = 20, scale = 5, size=50000)
#try size=1000

stats.probplot(measurements, dist="norm", plot=pylab)
pylab.show()
```




```
# generate 100 sanples from N(20,5)
measurements = np.random.uniform(low=-1, high=1, size=10000)
#try size=1000

stats.probplot(measurements, dist="norm", plot=pylab)
pylab.show()
```



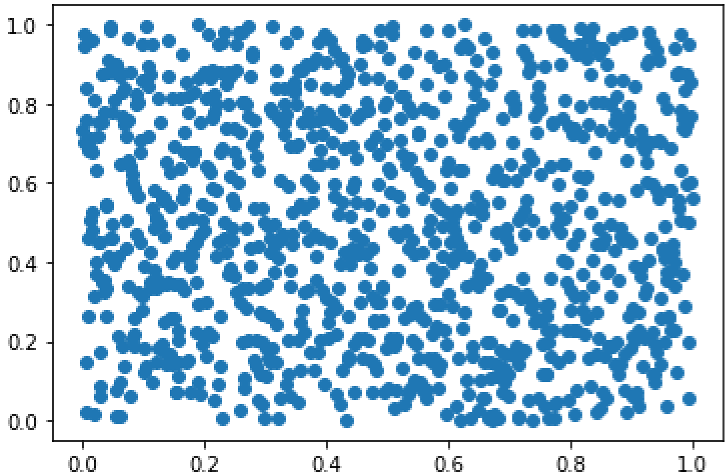

```
# generate 100 sanples from N(20,5)
measurements = np.random.uniform(low=-1, high=1, size=10000)
#try size=1000

stats.probplot(measurements, dist="uniform", plot=pylab)
pylab.show()
```



Random Numbers Generator (random distribution)

```
print(random.random()) #0 to 1, uniform distribution
rands = [random.random() for i in range(1000)]
plt.scatter(np.linspace(0,1, num=1000),rands)
plt.show()

#CDF
cnt, edges = np.histogram(rands, bins=20)
plt.plot(edges[1:], np.cumsum(cnt/sum(cnt)))
```

```
0.3209522535255618
```



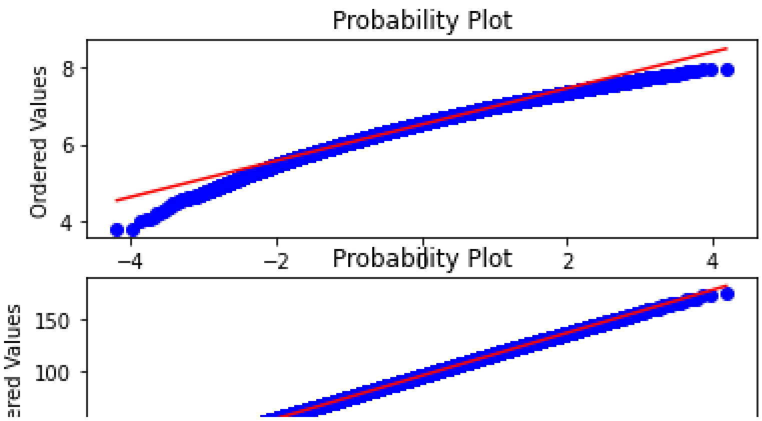
```
[<matplotlib.lines.Line2D at 0x7f7b713f8550>]
```

```
#n - datapoints, sample them of size m uniformly
df = datasets.load_iris().data;
n = df.shape[0]
m = 30

p = m/n
for j in range(1,5):
  sampled_data = []
  for i in range(n):
    if random.random() <= p:
      sampled_data.append(i)
  print(len(sampled_data)) #need not be 30 always, roughly around 30
```

```
28
33
27
33
```

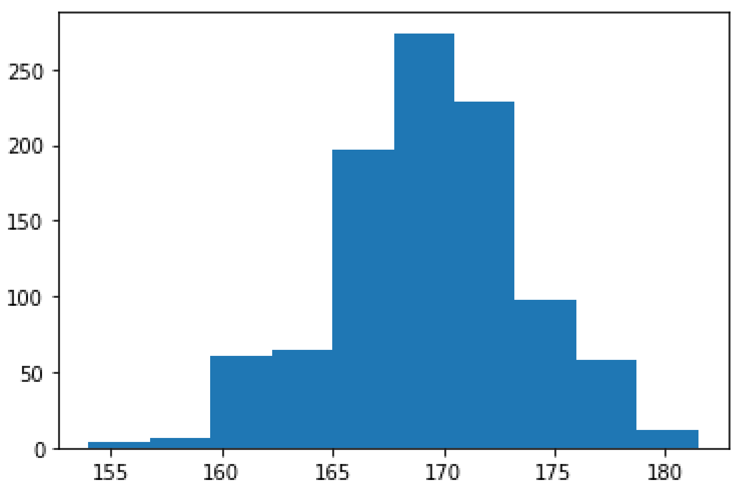
Box Cox Transformation

```
#power law distribution
fig, axes = plt.subplots(nrows = 2, ncols=1)
ax1 = axes[0]
X = stats.loggamma.rvs(5, size=50000) + 5
stats.probplot(X, dist="norm", plot=ax1)

Y, _ = stats.boxcox(X)
stats.probplot(Y, dist="norm", plot=axes[1])

plt.show()
```



Empirical bootstrap for CI

```
X = np.array([180,162,158,172,168,150,171,183,165,176])
k = 1000
m = int((len(X)*0.8))
md = []
for i in range(k):
  sample = resample(X, n_samples = m)
  md.append(np.median(sample))

plt.hist(md)
plt.show()

alpha = 0.95 #95% CI
range_n = np.array([0, ((1-alpha)/2)*100, 100*((1+alpha)/2),100])
md = np.sort(md)
np.percentile(md, range_n)
```

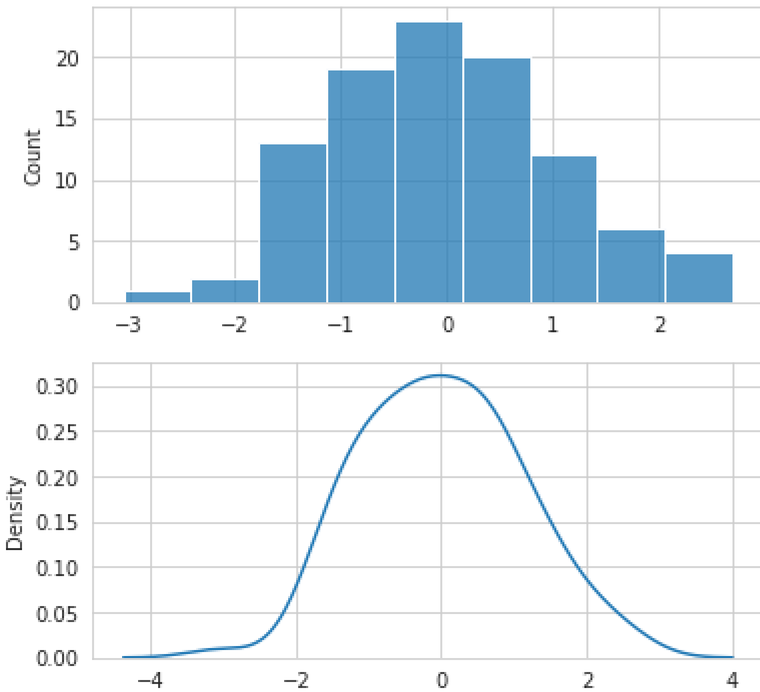



```
array([154. , 160. , 177. , 181.5])
```

KS Test

```
X = stats.norm.rvs(size=100)
fig, axes = plt.subplots(nrows = 2, ncols=1, figsize=(6,6) )
sns.set_style('whitegrid')
#sns.scatterplot(x=range(0,1000), y=X)
```

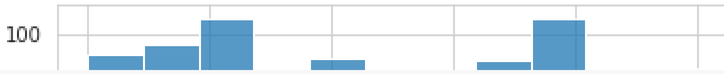
```
sns.histplot(X, ax=axes[0])
sns.kdeplot(X, ax=axes[1])
plt.show()
print(stats.kstest(X, 'norm'))
```




```
KstestResult(statistic=0.07403083345749373, pvalue=0.6368378097402114)
```

```
X = stats.uniform.rvs(size=1000)
print(X[0:50])
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(6,6))
sns.set_style('whitegrid')
#sns.scatterplot(data=X, ax=axes[0])
sns.histplot(X, ax=axes[0])
sns.kdeplot(X, bw=0.5, ax=axes[1])
plt.show()
print(stats.kstest(X, 'norm'))
```

```
[0.31906831 0.81668202 0.46445003 0.60371156 0.96736522 0.8039904
 0.78349851 0.25667239 0.11282044 0.26346924 0.81493582 0.45306526
 0.37891529 0.31140773 0.4583943  0.17486253 0.83452177 0.1480231
 0.03743587 0.86591515 0.81731302 0.69751853 0.90149015 0.40924313
 0.32390345 0.65339424 0.57019138 0.3665048  0.36414264 0.89174567
 0.27038761 0.42762324 0.24452063 0.78719723 0.0614169  0.47512458
 0.00561059 0.22037126 0.41811627 0.68009393 0.20992509 0.00196789
 0.36169091 0.29999564 0.18642837 0.947758   0.66751821 0.08403089
 0.05087222 0.88002339]
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:1657: Futur
  warnings.warn(msg, FutureWarning)
```

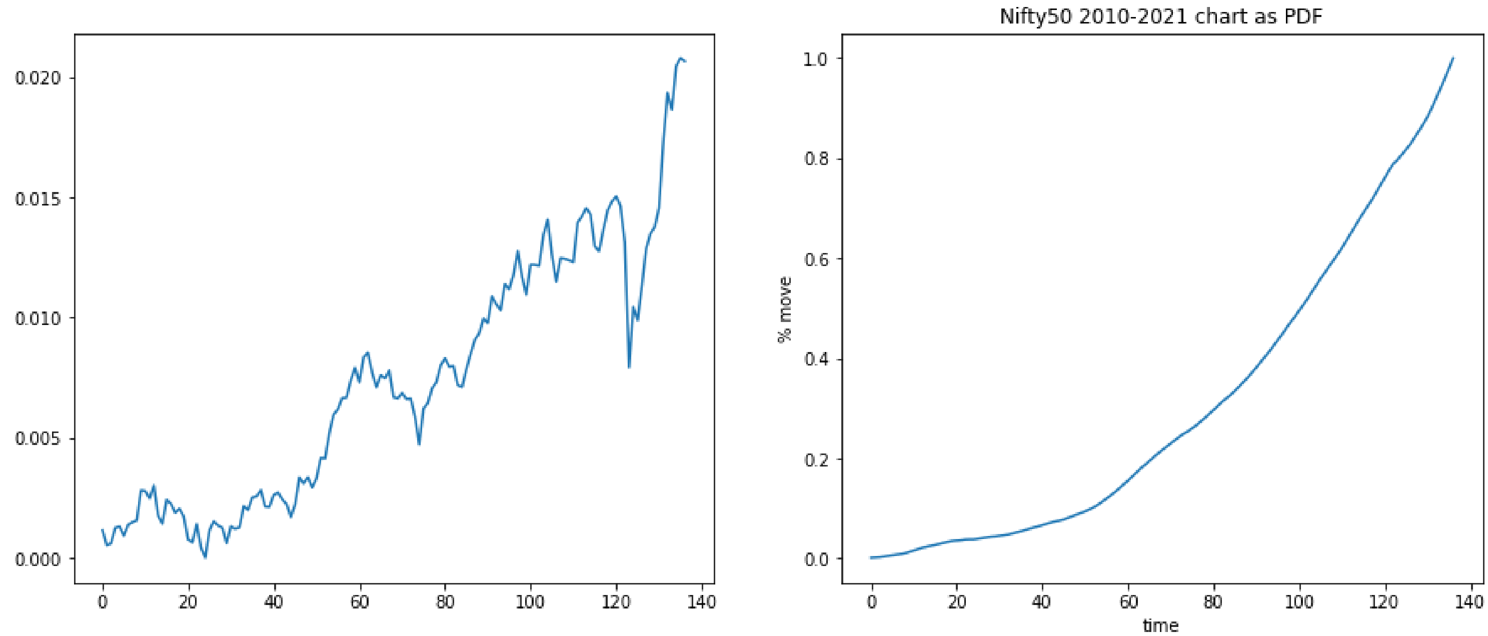



```python
X = stats.norm.rvs(size=100)
Y = stats.uniform.rvs(size=1000)
Z = stats.norm.rvs(size=1000)

print(stats.ks_2samp(X, Y))
print(stats.ks_2samp(X, Z))
```

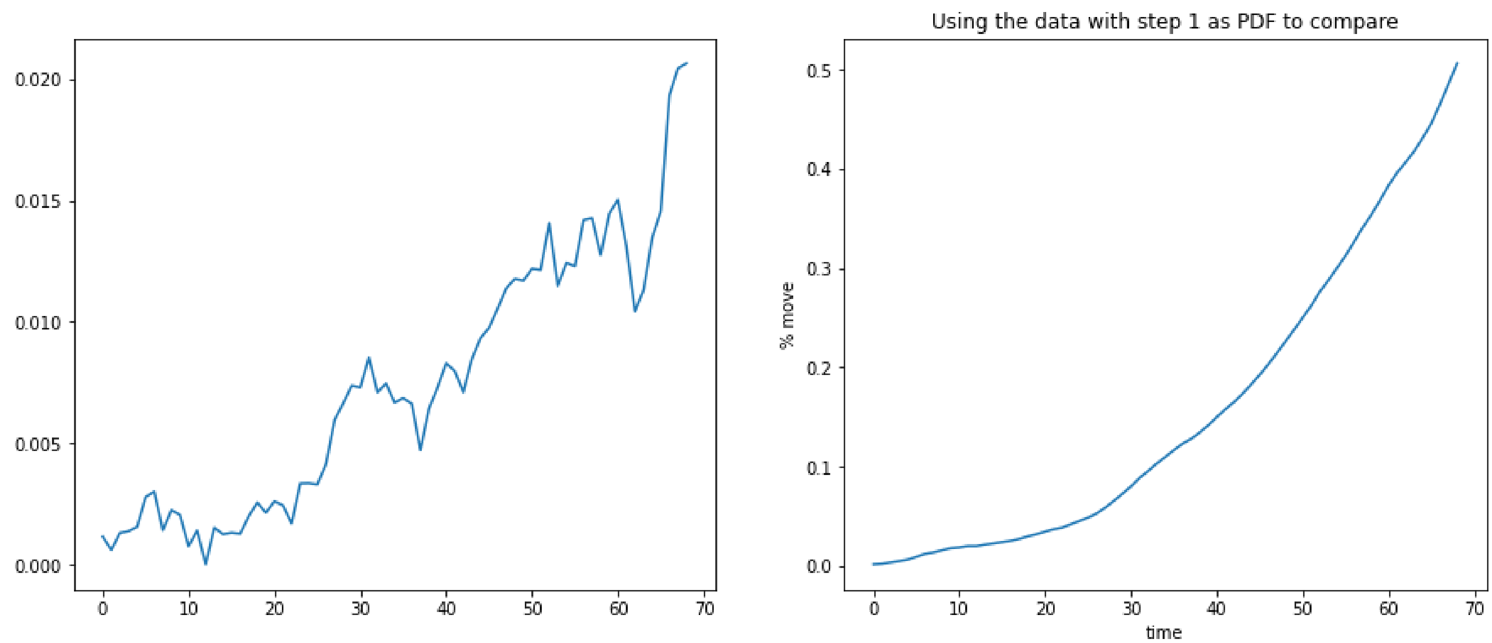
```
Ks_2sampResult(statistic=0.55, pvalue=2.5991628500150062e-24)
Ks_2sampResult(statistic=0.101, pvalue=0.2955254321543853)
```

```python
#Thinking the stock market graph as PDF and comparing it with another
percentages = np.array([-6.13,0.82,6.64,0.55,-3.63,4.45,1.04,0.65,11.62,-0.20,-2.58,4.64,-10.25
X = []
prev = 100
for i in range(len(percentages)):
  pp = percentages[i]/100
  if(i==0):
    X.append(100)
  val = prev*(1+(pp))
  X.append(val)
  prev = val

X_norm = ((X - np.min(X))/(np.max(X) - np.min(X)))
X_norm = X_norm/np.sum(X_norm)
fig, axes = plt.subplots(nrows = 1, ncols=2, figsize=(15,6))
axes[0].plot(range(len(X_norm)), X_norm)
axes[1].plot(range(len(X_norm)), np.cumsum(X_norm))
plt.xlabel('time')
plt.ylabel('% move')
plt.title("Nifty50 2010-2021 chart as PDF")
plt.show()

#print(stats.kstest(X, lambda X: cdf_norm))
step = 1
test = X_norm[::2]
d,p_val = stats.ks_2samp(X_norm, test)
print("P-value : ", p_val)
fig, axes = plt.subplots(nrows = 1, ncols=2, figsize=(15,6))
axes[0].plot(range(len(test)), test)
axes[1].plot(range(len(test)), np.cumsum(test), label='Test CDF')
plt.xlabel('time')
plt.ylabel('% move')
```

```
plt.title("Using the data with step {} as PDF to compare".format(step))
plt.show()
```




```
P-value :  0.9999999863173562
```



1s completed at 7:29 PM