index

January 20, 2022

1 The Kolmogorov-Smirnov Test - Lab

1.1 Introduction

In the previous lesson, we saw that the Kolmogorov–Smirnov statistic quantifies a distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution, or between the empirical distribution functions of two samples. In this lab, we shall see how to perform this test in Python.

1.2 Objectives

In this lab you will:

- Calculate a one- and two-sample Kolmogorov-Smirnov test
- Interpret the results of a one- and two-sample Kolmogorov-Smirnov test
- Compare K-S test to visual approaches for testing for normality assumption

1.2.1 Data

Let's import the necessary libraries and generate some data. Run the following cell:

[1]:
```
import scipy.stats as stats
import statsmodels.api as sm
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('ggplot')

# Create the normal random variables with mean 0, and sd 3
x_10 = stats.norm.rvs(loc=0, scale=3, size=10)
x_50 = stats.norm.rvs(loc=0, scale=3, size=50)
x_100 = stats.norm.rvs(loc=0, scale=3, size=100)
x_1000 = stats.norm.rvs(loc=0, scale=3, size=1000)
```

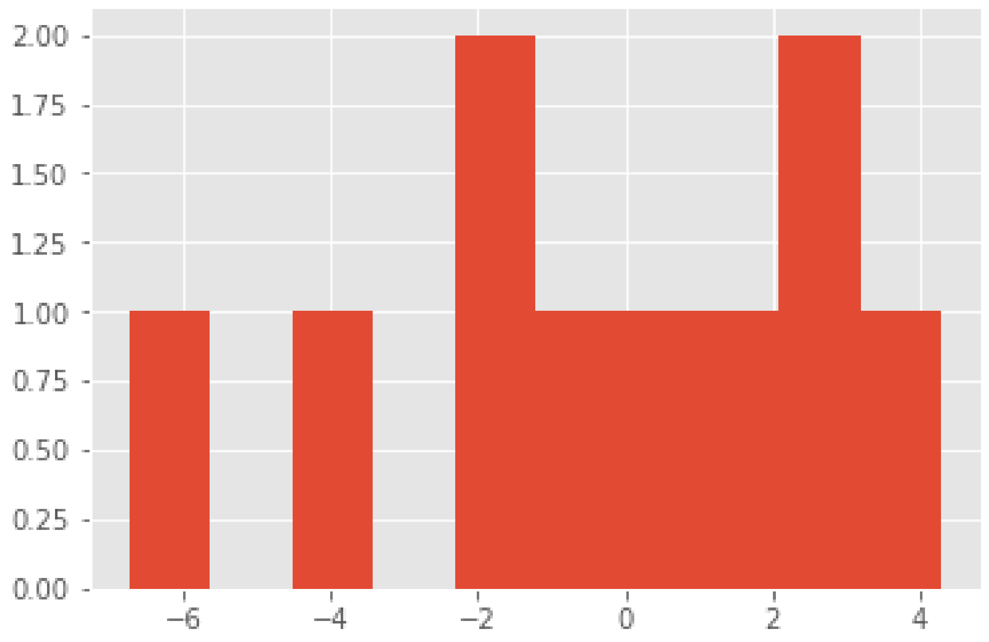
1.2.2 Plots

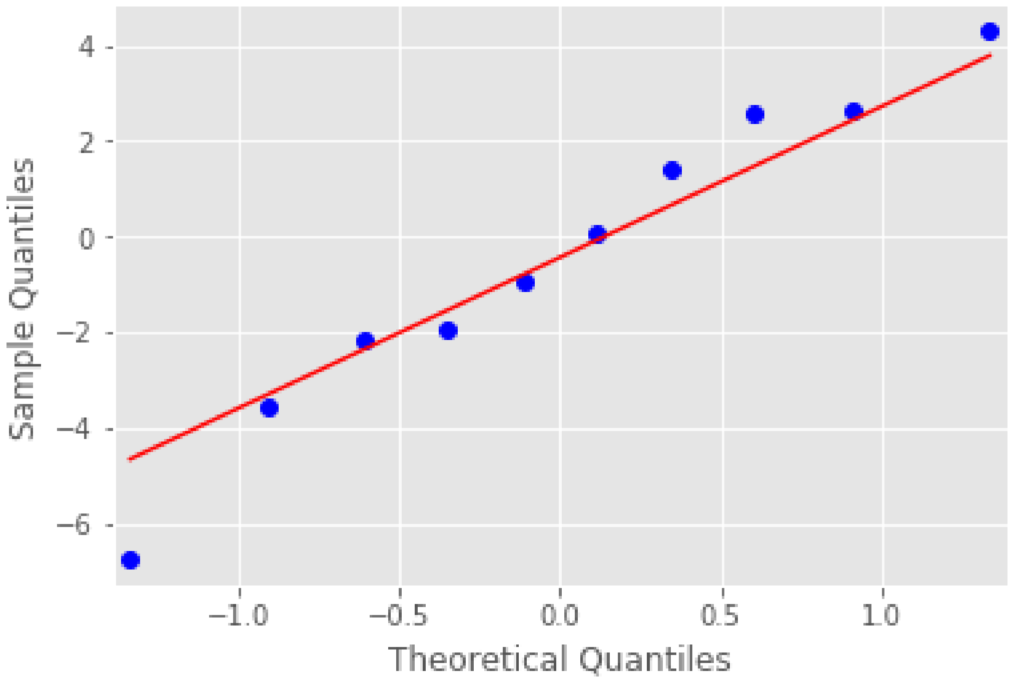
Plot histograms and Q-Q plots of above datasets and comment on the output

- How good are these techniques for checking normality assumptions?
- Compare both these techniques and identify their limitations/benefits etc.

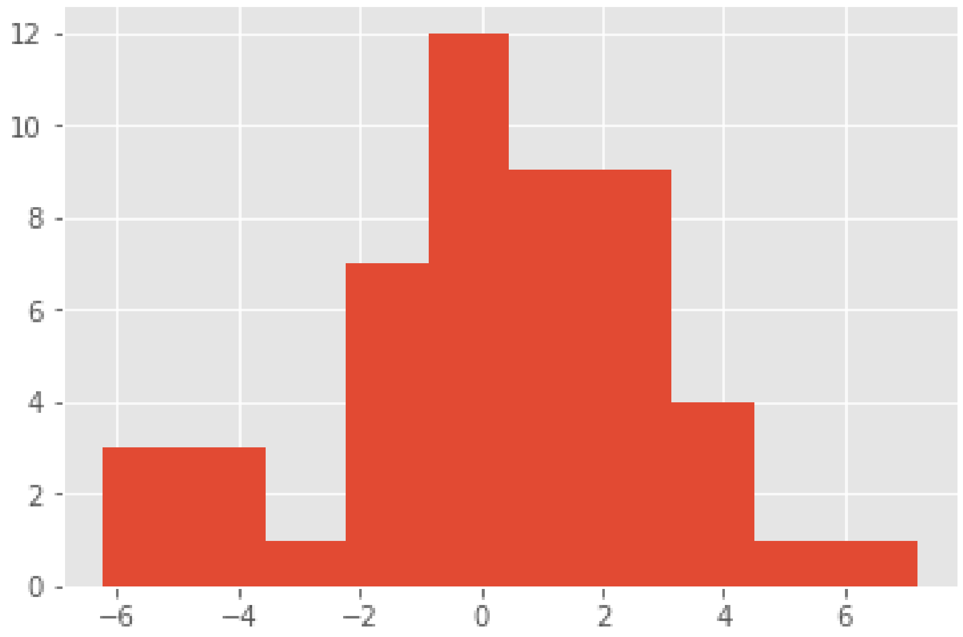
```
[2]: # # Plot histograms and Q-Q plots for above datasets
items = [x_10, x_50, x_100, x_1000]
names = ["x_10", "x_50", "x_100", "x_1000"]

for i, item in enumerate(items):
    print(names[i])

    plt.hist(item)

    sm.qqplot(item, line = "s")
    plt.show()
```

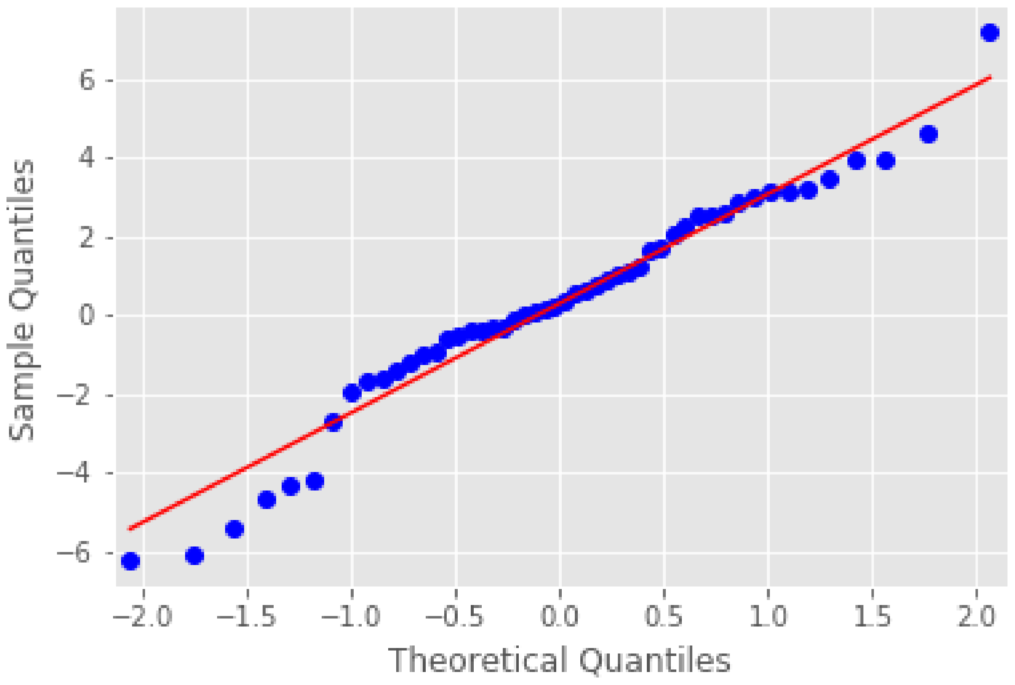
```
x_10
```



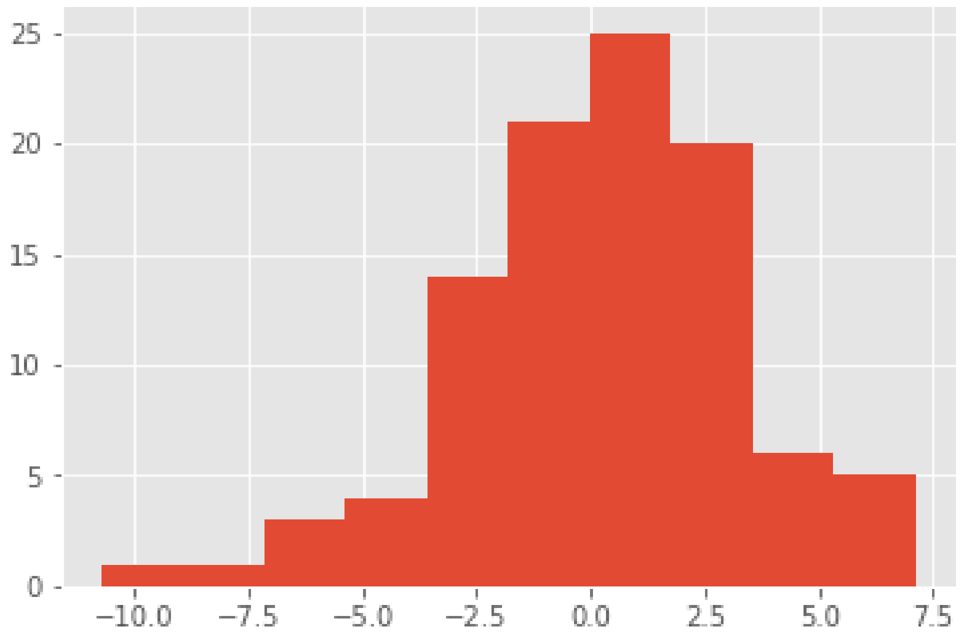

Sample Quantiles
Theoretical Quantiles
4
2
0
−2
−4
−6
−1.0
−0.5
0.0
0.5
1.0

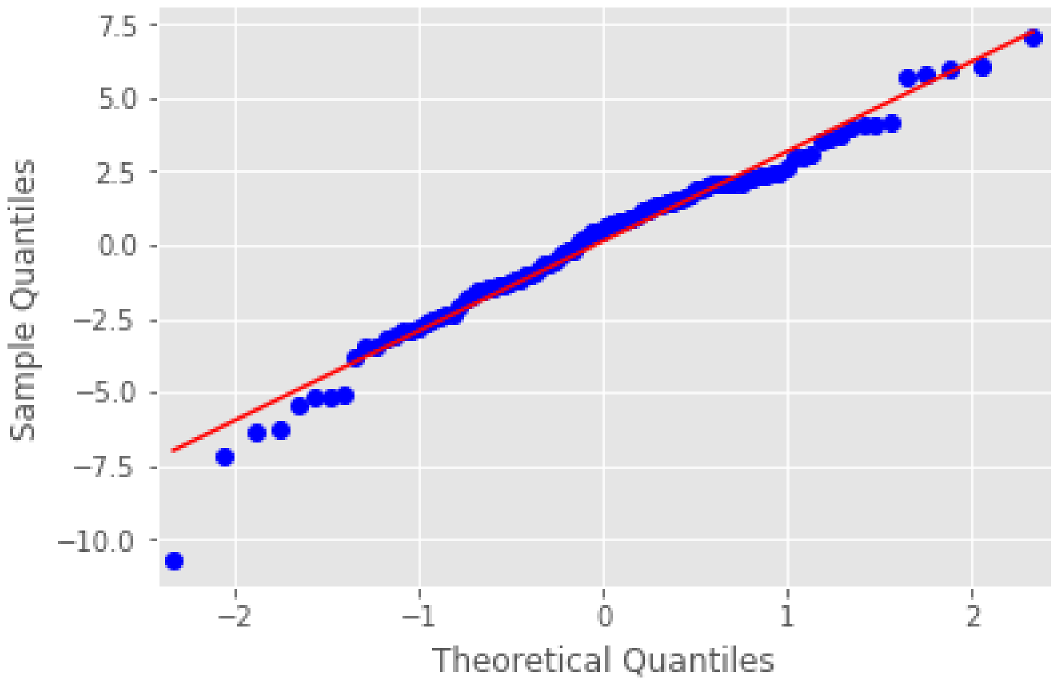
x_50


12
10
8
6
4
2
0
−6
−4
−2
0
2
4
6


Sample Quantiles
Theoretical Quantiles
6
4
2
0
−2
−4
−6
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

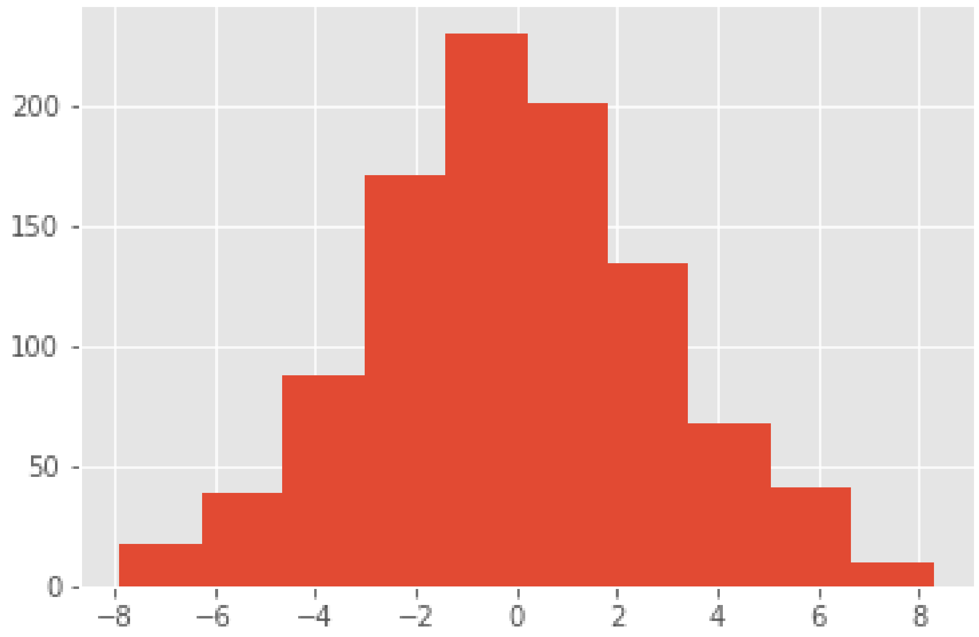
x_100


25
20
15
10
5
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

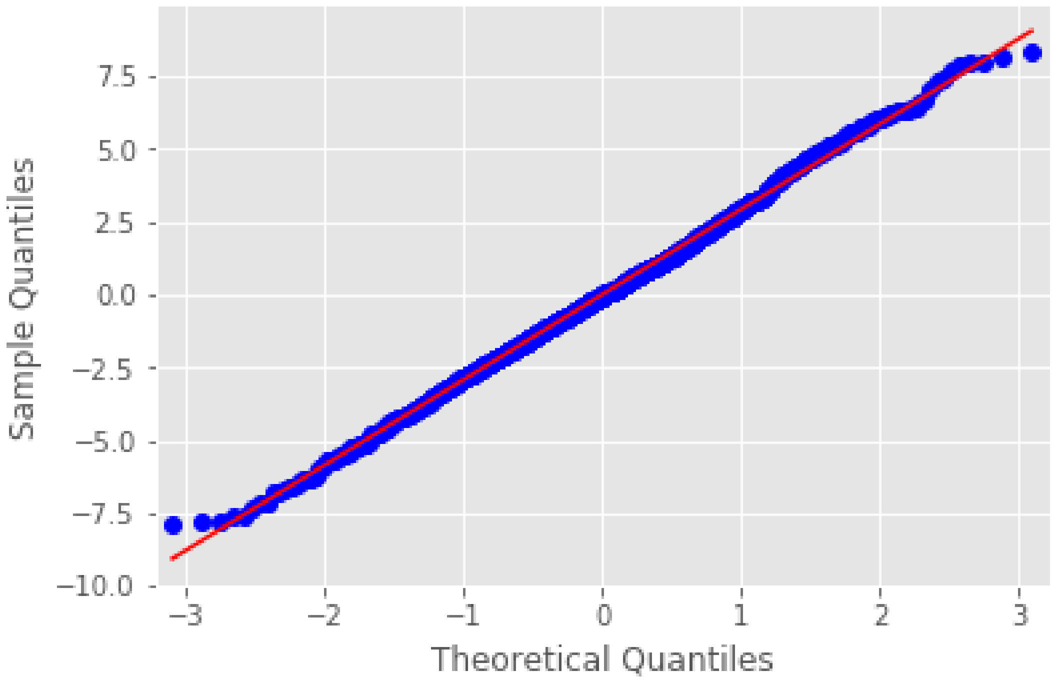


Sample Quantiles
Theoretical Quantiles
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−2
−1
0
1
2

`x_1000`



200
150
100
50
0
−8
−6
−4
−2
0
2
4
6
8

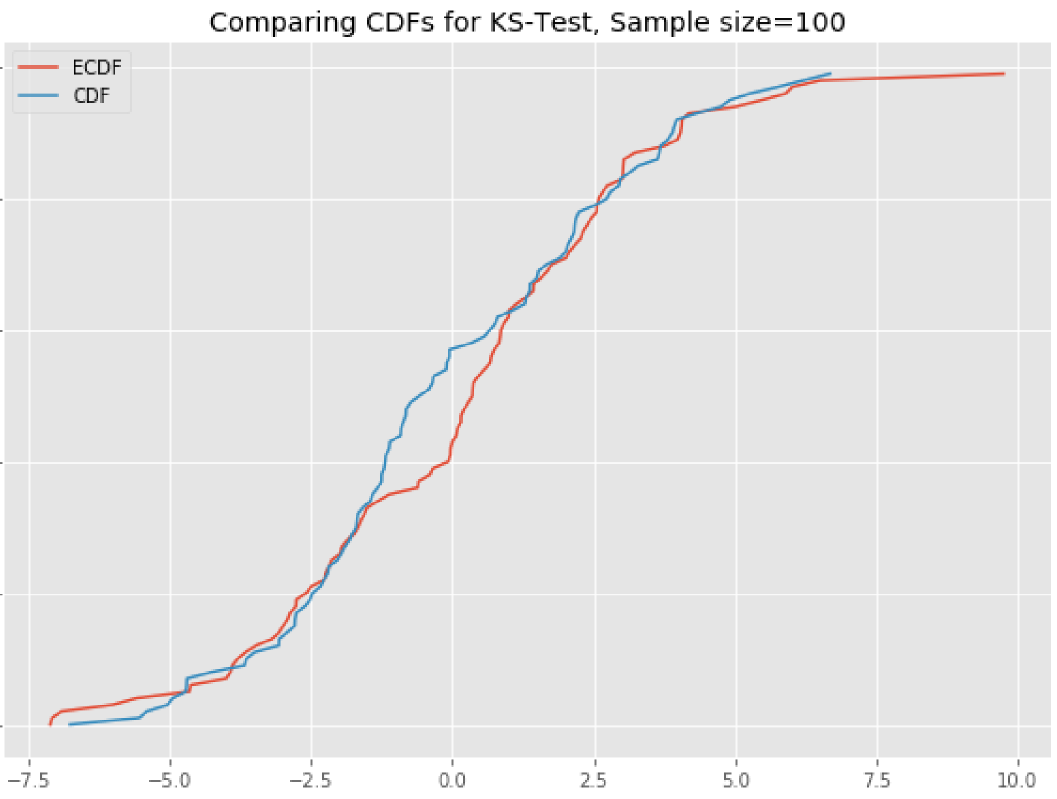


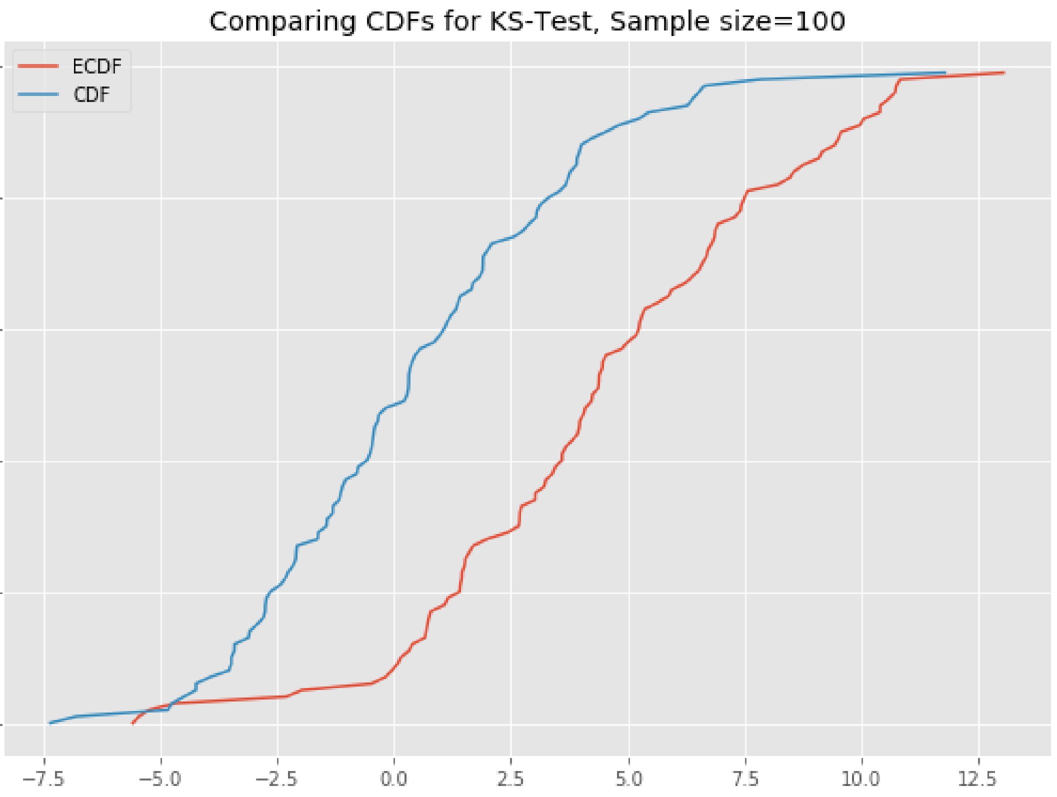
[3]:
```
# Your comments here

# From GitHub Solution

# Histograms should not be used solely to detect normality directly
# Histograms are better to look for symmetry, skewness, and outliers
# These can instead be used to get an indications of non-normality.

# We see some outliers in our datasets
# no clear indications of non-normality for each plot.

# The Q-Q plot is a much better visualization of data as gives a reference to
# compare against
# Shows a better picture about normality instead of relying on the histograms
# (or box plots).
# From Q-Q plot we can be more assured our data is normal
# compared to non normality check in histogram
```

1.2.3 Create a function to plot the normal CDF and ECDF for a given dataset

- Create a function to generate an empirical CDF from data
- Create a normal CDF using the same mean = 0 and sd = 3, having the same number of values as data

```
[13]: # You code here

def ks_plot(data):

    plt.figure(figsize = (10,10))
    plt.plot(np.sort(data), np.linspace(0, 1, len(data)))
    x = np.random.normal(0, 3, len(data))
    plt.plot(np.sort(x), np.linspace(0, 1, len(x)), c = "black")

    plt.legend(['ECDF', 'CDF'])
    plt.title('Comparing CDFs for K-S test, Sample size=' + str(len(data)))
    plt.show()

#     Or we can use the following code to make ECDF and CDF

#     kwargs = {'cumulative': True}
#     sns.distplot(data, hist_kws=kwargs, kde_kws=kwargs)
#     sns.ecdfplot(data = data)


# Uncomment below to run the test
ks_plot(stats.norm.rvs(loc=0, scale=3, size=100))
ks_plot(stats.norm.rvs(loc=5, scale=4, size=100))



#### From GitHub


# def ks_plot(data):

#     plt.figure(figsize=(10, 7))
#     plt.plot(np.sort(data), np.linspace(0, 1, len(data), endpoint=False))
#     plt.plot(np.sort(stats.norm.rvs(loc=0, scale=3, size=len(data))), np.
 ↪linspace(0, 1, len(data), endpoint=False))

#     plt.legend(['ECDF', 'CDF'])
#     plt.title('Comparing CDFs for K-S test, Sample size=' + str(len(data)))
```


Comparing CDFs for KS-Test, Sample size=100
ECDF
CDF
1.0
0.8
0.6
0.4
0.2
0.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

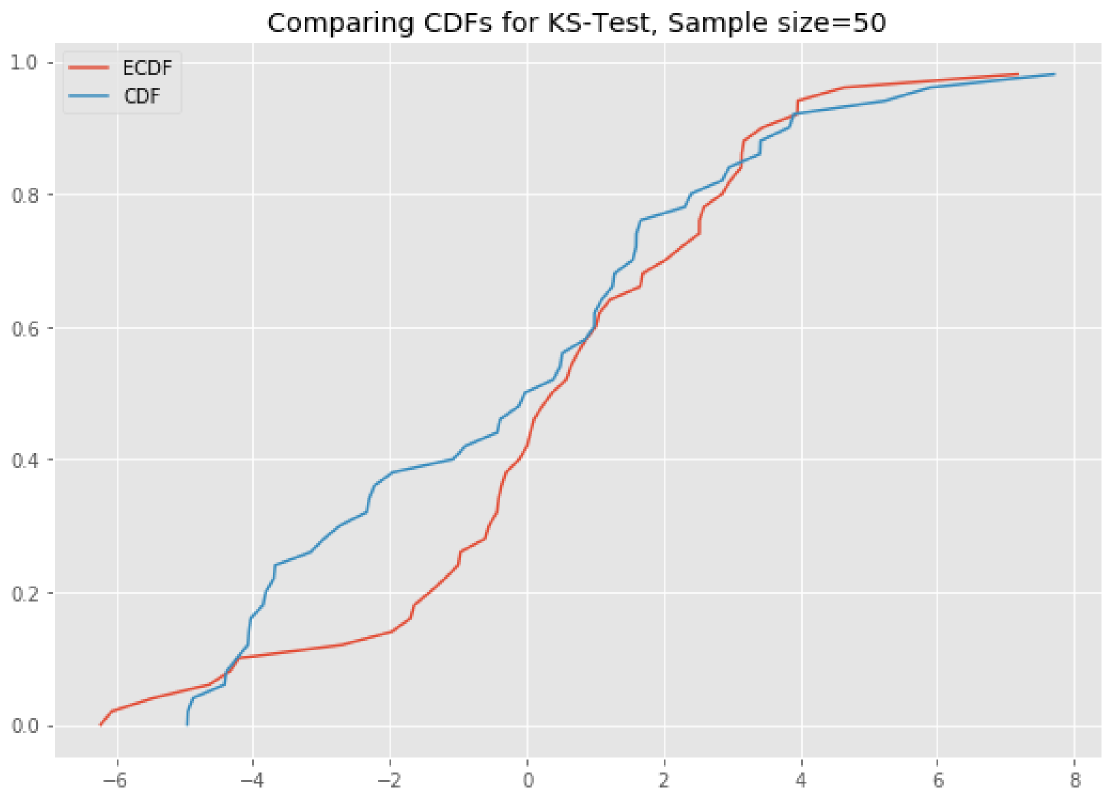

Comparing CDFs for KS-Test, Sample size=100
ECDF
CDF
1.0
0.8
0.6
0.4
0.2
0.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5

This is awesome. The difference between the two CDFs in the second plot shows that the sample did not come from the distribution which we tried to compare it against.

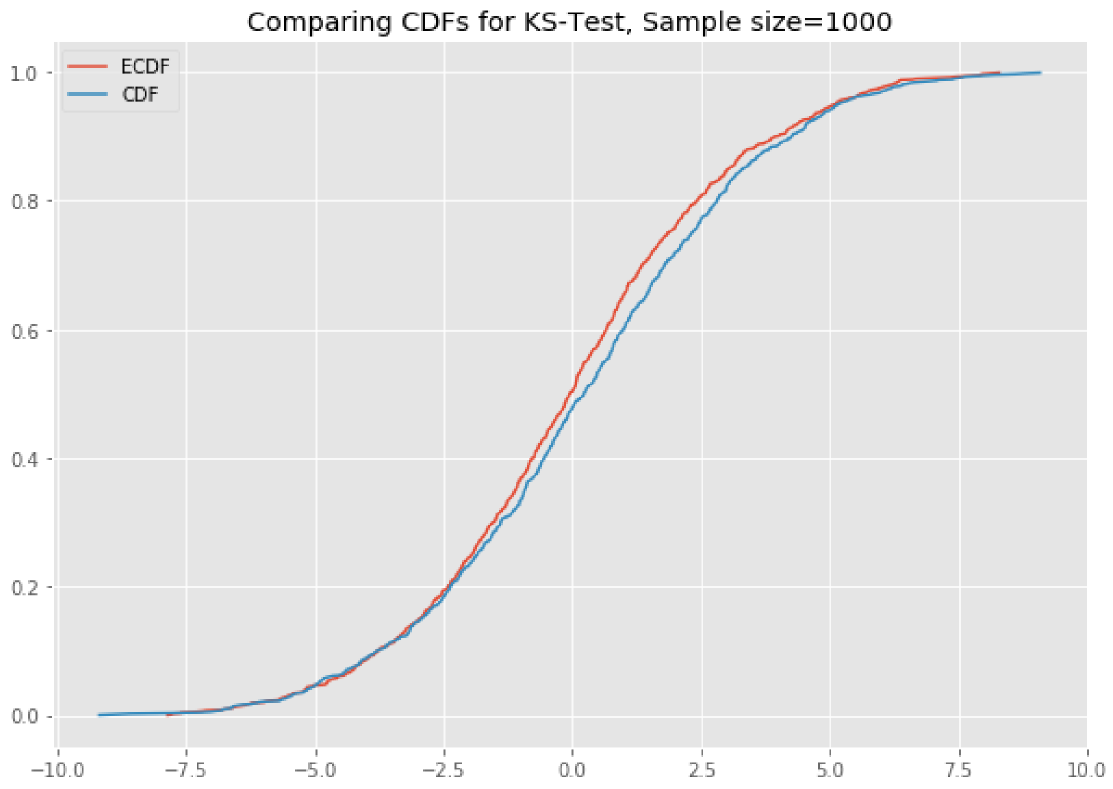
Now you can run all the generated datasets through the function `ks_plot()` and comment on the output.

[15]:
```
# Your code here
names = ["x_10", "x_50", "x_100", "x_1000"]
items = [x_10, x_50, x_100, x_1000]

for i, item in enumerate(items):
    print(names[i])
    ks_plot(item)

# ks_plot(x_10)
# ks_plot(x_50)
# ks_plot(x_100)
# ks_plot(x_1000)
```

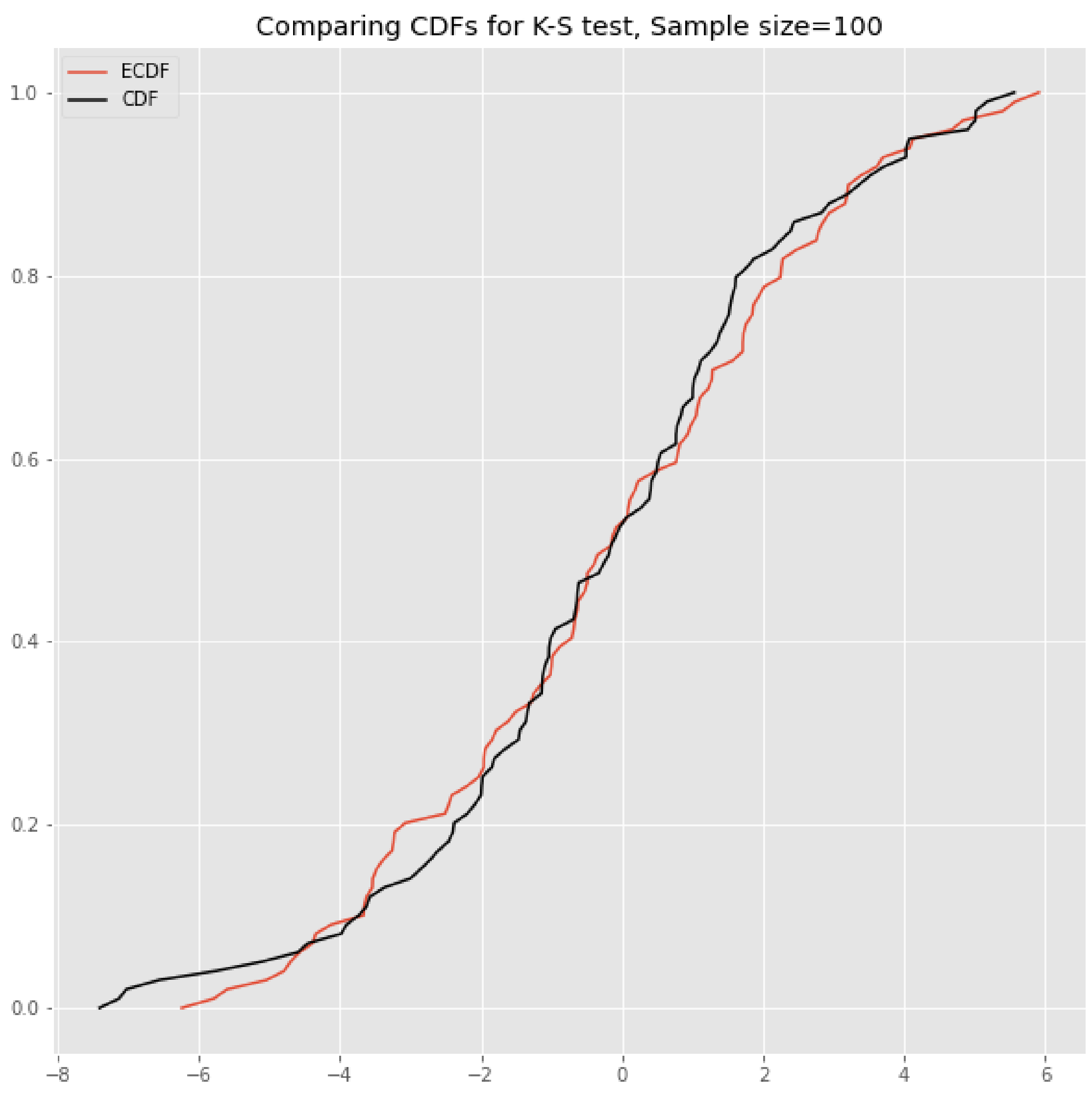
```
x_10
```



x_50

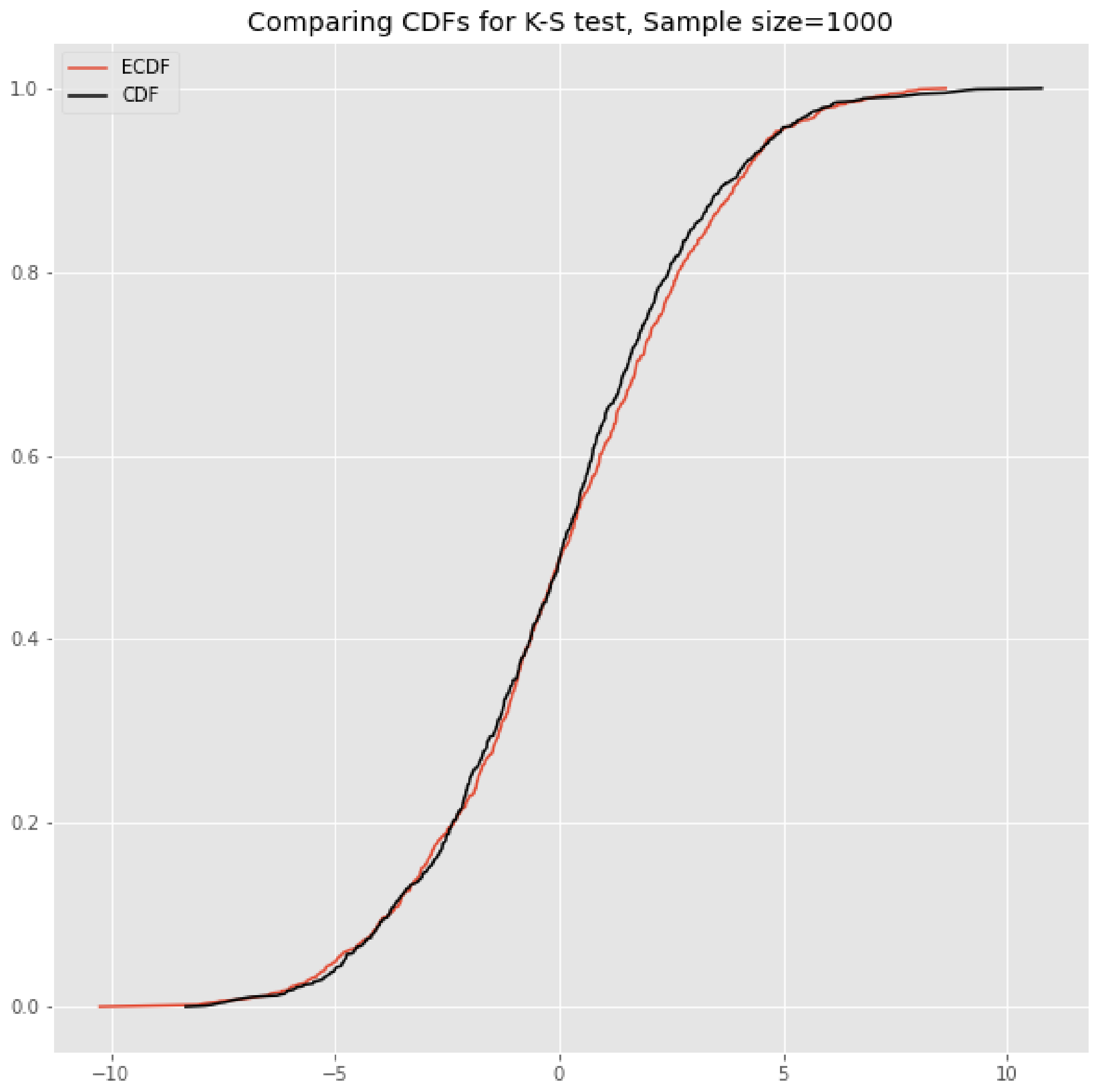

Comparing CDFs for KS-Test, Sample size=1000
ECDF
CDF
1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

x_100



Comparing CDFs for K-S test, Sample size=100
ECDF
CDF
1.0
0.8
0.6
0.4
0.2
0.0
−8
−6
−4
−2
0
2
4
6

```
x_1000
```



[87]:
```
# Your comments here

# From GitHub

# Your comments here

# As we have more data values to compare, we get a better idea of normality
# Due to randomness in smaller sample sizes, it is very likely that the value
# of d would be high

# As our sample size goes from 50 to a 1000, we are in a much better position
# to comment on normality
```

1.2.4 K-S test in SciPy

Let's run the Kolmogorov-Smirnov test, and use some statistics to get a final verdict on normality. We will test the hypothesis that the sample is a part of the standard t-distribution. In SciPy, we run this test using the function below:

```
scipy.stats.kstest(rvs, cdf, args=(), N=20, alternative='two-sided', mode='approx')
```

Details on arguments being passed in can be viewed at this link to the official doc.

Run the K-S test for normality assumption using the datasets created earlier and comment on the output: - Perform the K-S test against a normal distribution with mean = 0 and sd = 3 - If $p < .05$ we can reject the null hypothesis and conclude our sample distribution is not identical to a normal distribution

[108]:
```
# Perform K-S test
# np.random.seed(999)
# # stats.kstest(x_10,   "norm", )
# # stats.kstest(x_50,   "norm", )
# # stats.kstest(x_100,  "norm", )
print(stats.kstest(x_10, "norm", args = (0, 3)))
print(stats.kstest(x_50, "norm", args = (0, 3)))
print(stats.kstest(x_100, "norm", args = (0, 3)))
print(stats.kstest(x_1000, "norm", args = (0, 3)))

# KstestResult(statistic=0.1377823669421559, pvalue=0.9913389045954595)
# KstestResult(statistic=0.13970573965633104, pvalue=0.2587483380087914)
# KstestResult(statistic=0.0901015276393986, pvalue=0.37158535281797134)
# KstestResult(statistic=0.030748345486274697, pvalue=0.29574612286614443)
```

```
KstestResult(statistic=0.22546848555648163, pvalue=0.6129979582147573)
KstestResult(statistic=0.13269859042606313, pvalue=0.3137507368413688)
KstestResult(statistic=0.05816785829434465, pvalue=0.8678334353802855)
KstestResult(statistic=0.022692585822075895, pvalue=0.6730302420775669)
```

[109]:
```
# Your comments here

# We know that the data we made is normal, however, with this test, we
# find that the p-value is much greter than 0.05, so we cannot reject
# the Null Hypothesis
```

Generate a uniform distribution and plot / calculate the K-S test against a uniform as well as a normal distribution:

[112]:
```
x_uni = np.random.rand(1000)
# Try with a uniform distribution
print(stats.kstest(x_uni, lambda x: x))
print(stats.kstest(x_uni, "norm", args = (0, 3)))
# The first Code is From GitHub
# KstestResult(statistic=0.023778383763166322, pvalue=0.6239045200710681)
# KstestResult(statistic=0.5000553288071681, pvalue=0.0)
```

```
KstestResult(statistic=0.018771636391163304, pvalue=0.8658988171856125)
KstestResult(statistic=0.5000336492467594, pvalue=9.85871422898e-232)
```

[113]:
```
# Your comments here

# From GitHub
# Your comments here

# In the first case, the p-value is much larger than 0.05 so we cannot reject
# the Null Hypothesis and conclude that our sample is a uniform distribution

# In the second case, comparing a uniform distribution against a normal CDF,
# the p value - 0 so we reject the Null Hypothesis with
# a high degree of confidence
```

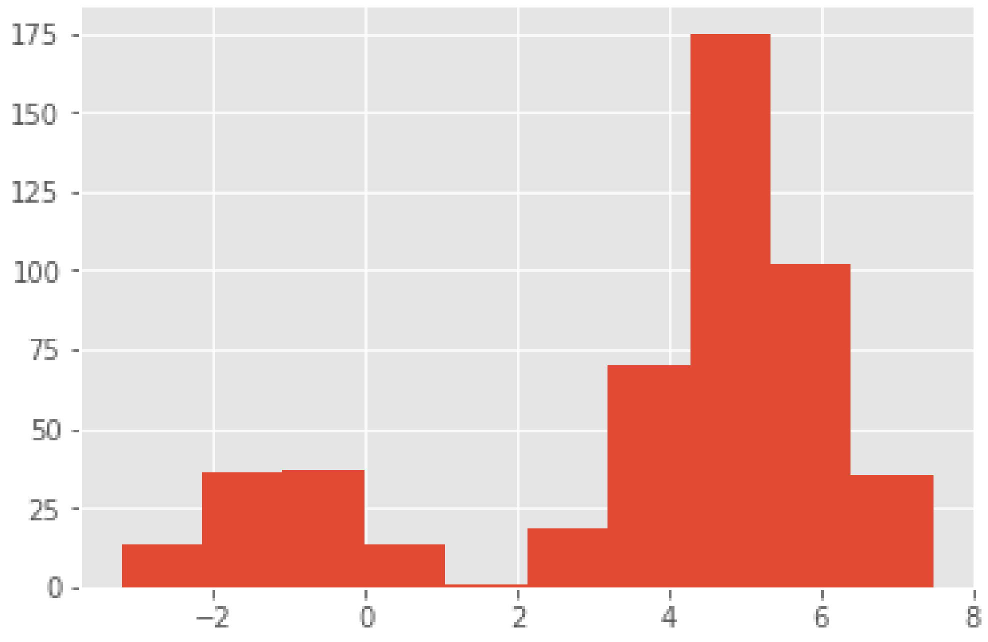
1.3 Two-sample K-S test

A two-sample K-S test is available in SciPy using following function:

`scipy.stats.ks_2samp(data1, data2)[source]`

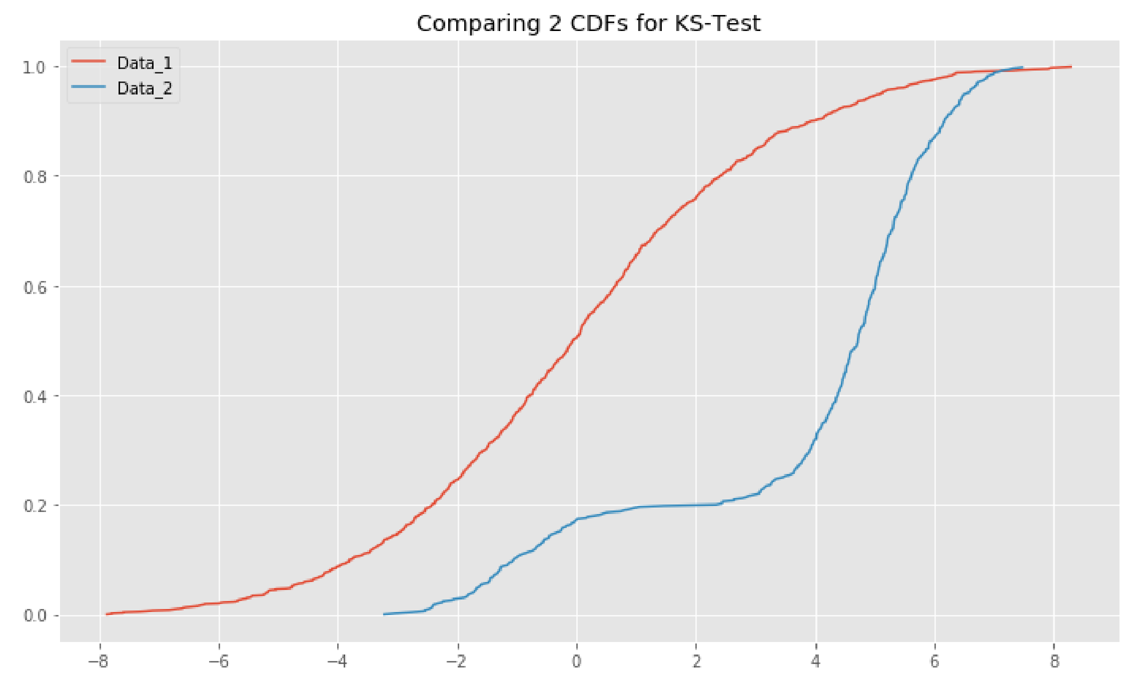
Let's generate some bi-modal data first for this test:

[18]:
```
# Generate binomial data
N = 1000
x_1000_bi = np.concatenate(
    (
    np.random.normal(-1, 1, int(0.1 * N)),
    np.random.normal(5, 1, int(0.4 * N))))[:, np.newaxis]
plt.hist(x_1000_bi);
```



Plot the CDFs for `x_1000_bimodal` and `x_1000` and comment on the output.

[30]:
```python
# Plot the CDFs
def ks_plot_2sample(data_1, data_2):
    '''
    Data entered must be the same size.
    '''
    plt.figure(figsize=(12, 7))
    plt.plot(np.sort(data_1), np.linspace(0, 1, len(data_1)),
             color = "red", label = "Data_1")

    plt.plot(np.sort(data_2), np.linspace(0, 1, len(data_2)),
             color = "blue", label = "Data_2")

    plt.legend()
    plt.title("Comparing two CDFs for KS test")
    plt.show()

# Uncomment below to run
ks_plot_2sample(x_1000, x_1000_bi[:,0])
```



```
[31]: # You comments here

      # x_1000 and x_1000_bi diverge a lot
      # We can expect a high value for the d statistic
```

Run the two-sample K-S test on `x_1000` and `x_1000_bi` and comment on the results.

```
[33]: # Your code here
      stats.ks_2samp(x_1000, x_1000_bi[:,0] )
      # Ks_2sampResult(statistic=0.633, pvalue=4.814801487740621e-118)
```

```
[33]: KstestResult(statistic=0.6, pvalue=1.7763568394002505e-15)
```

```
[ ]: # Your comments here

     # P Value is much less than alpha so we reject the Null Hypothesis
     # stating that these two samples are drwan from a same population
     # So we conclude that the two samples belong to two different distributions
```

1.4 Summary

In this lesson, we saw how to check for normality (and other distributions) using one- and two-sample K-S tests. You are encouraged to use this test for all the upcoming algorithms and techniques that require a normality assumption. We saw that we can actually make assumptions for different distributions by providing the correct CDF function into Scipy K-S test functions.