Probability plot and Data Transformation

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
from sklearn.preprocessing import FunctionTransformer
```

Entering data

In [2]:
```
data = pd.read_excel('data.xlsx')
data["di * ni"] = data['di'] * data['ni']
data.head()
```

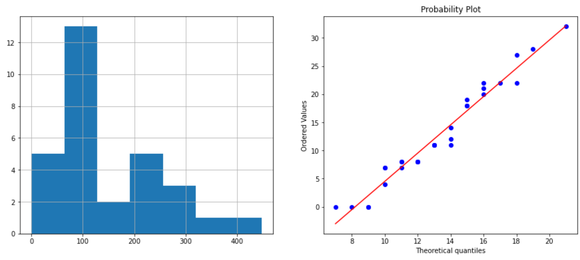
Out[2]:

	zi	di	ni	di * ni
0	2.461784	20	10	200
1	3.816171	28	12	336
2	-1.397182	7	10	70
3	-3.495034	0	10	0
4	4.545412	27	10	270

Q-Q plot for poisson distribution

In [3]:
```
def diagnostic_plots_poisson(df,variable):
    plt.figure(figsize=(15,6))
    plt.subplot(1,2,1)
    df[variable].hist(bins = 7)
    plt.subplot(1,2,2)
    param = np.sum(df[variable])/np.sum(df['ni'])
    stats.probplot(df['di'], dist='poisson', sparams=(param,) , plot = plt)
    plt.show()
```

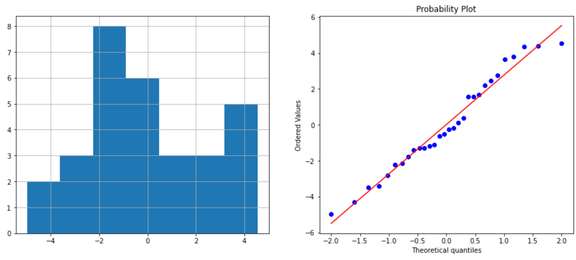
In [4]:
```
diagnostic_plots_poisson(data,"di * ni")
```



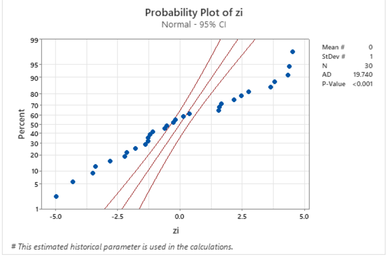
Q-Q plot for normal distribution

In [5]:
```
def diagnostic_plots_normal(df,variable):
    plt.figure(figsize=(15,6))
    plt.subplot(1,2,1)
    df[variable].hist(bins = 7)
    plt.subplot(1,2,2)
    stats.probplot(df[variable], dist='norm', plot = plt)
    plt.show()
```

In [6]:
```
diagnostic_plots_normal(data,'zi')
```



Q-Q plot for normal distribution in Minitab



Data transformation with Yeo-johnson method

In [7]:
```
from sklearn.preprocessing import PowerTransformer
# choose a method
transformer = PowerTransformer(method='yeo-johnson')
# fit the lamda parameter to data
transformer.fit(data[['zi']])
# transform data
Zi_tf = transformer.transform(data[['zi']])
```

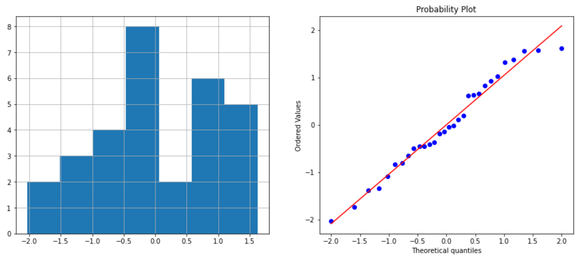
In [8]:
```
Zi_tf = pd.DataFrame(Zi_tf,columns=['Zi_tf'])
Zi_tf.head()
```

Out[8]:

	Zi_tf
0	0.930396
1	1.384697
2	-0.498984
3	-1.384139
4	1.624494

Q-Q plot for normal distribution after Yeo-Johnson transformation

In [9]:
```
diagnostic_plots_normal(Zi_tf,'Zi_tf')
```



Q-Q plot for normal distribution after Yeo-Johnson transformation in Minitab