In [1]:
```
### FITTING MULTI LINEAR REGRESSION MODEL FOR COVID DATASET
```

In [2]:
```
## Modules required
import pandas as pd
import seaborn as sns
import numpy as np
import pylab
import math
import matplotlib.pyplot as plt
```

In [3]:
```
from scipy import stats
import statsmodels.api as sm
from statsmodels.stats import diagnostic as diag
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
%matplotlib inline
```

In [4]:
```
 ## Load the dataset into pandas
covid19=pd.read_excel('covid19.xlsx')
```

In [5]:
```
covid19.head()
```

Out[5]:

	STATE	STCD	REGION	CDHS	HOSC	ICU	PLUF	SINC	POPD	POPS	...	UNEM	MEDA	LI
0	Alabama	AL	Southeast	1265	1547	0	16.8	219230	96.9221	4908620	...	7.5	20	
1	Alaska	AK	West	11	34	0	11.1	46099	1.2863	734002	...	12.4	21	
2	Arizona	AZ	Southwest	2443	3094	870	14.1	346009	64.9549	7378490	...	10.0	22	
3	Arkansas	AR	Southeast	362	474	0	16.8	137609	58.4030	3039000	...	8.0	27	
4	California	CA	West	7100	8820	2284	12.8	2701899	256.3728	39937500	...	14.9	26	

5 rows × 22 columns

In [6]:
```
## set the index equal to the year column
covid19.index = covid19['CDHS']
covid19 = covid19.drop(['STATE', 'STCD','REGION','CDHS'], axis = 1)
```

In [7]:
```
covid19.head()
```

Out[7]:

	HOSC	ICU	PLUF	SINC	POPD	POPS	HOML	HUMI	UNEM	MEDA	LEXP	ADEP	ATEM
CDHS													
1265	1547	0	16.8	219230	96.9221	4908620	3261	76.49	7.5	20	75.4	63.1	62.8
11	34	0	11.1	46099	1.2863	734002	1907	81.46	12.4	21	78.3	55.8	26.6
2443	3094	870	14.1	346009	64.9549	7378490	10007	79.40	10.0	22	79.5	67.2	60.3
362	474	0	16.8	137609	58.4030	3039000	2717	76.92	8.0	27	76.0	66.4	60.4
7100	8820	2284	12.8	2701899	256.3728	39937500	151278	80.36	14.9	26	80.8	58.1	59.4

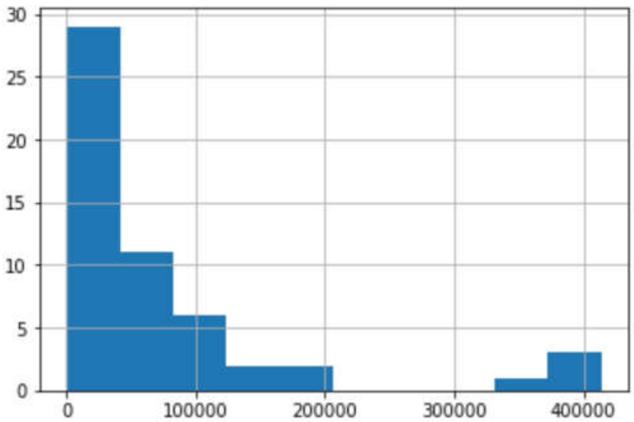
In [8]:
```
## Get the summary of our original data set
desc_covid19 = covid19.describe()
## Add the standard deviation metric
desc_covid19.loc['+3_std']=desc_covid19.loc['mean']+(desc_covid19.loc['std']*3)
desc_covid19.loc['-3_std']=desc_covid19.loc['mean']-(desc_covid19.loc['std']*3)
desc_covid19
```

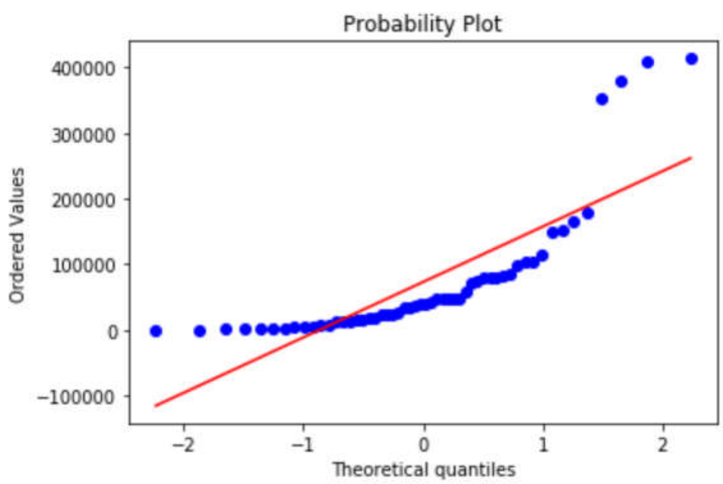
Out[8]:

	HOSC	ICU	PLUF	SINC	POPD	POPS	HOML	
count	54.000000	54.000000	54.000000	5.400000e+01	54.000000	5.400000e+01	54.000000	54.
mean	1104.222222	193.648148	12.148148	3.509831e+05	188.797204	6.122194e+06	10290.055556	73.
std	2233.213293	543.764695	4.018483	4.635854e+05	262.712798	7.401568e+06	23642.778670	18.
min	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000e+00	0.000000	0.
25%	64.000000	0.000000	10.625000	7.769300e+04	36.683350	1.381610e+06	1524.500000	75.
50%	403.500000	13.000000	12.350000	2.097215e+05	93.333700	4.127955e+06	4011.500000	77.
75%	1101.000000	149.500000	14.100000	4.808068e+05	218.398050	7.278018e+06	9201.000000	79.
max	10893.000000	3281.000000	19.800000	2.701899e+06	1215.198500	3.993750e+07	151278.000000	82.
+3_std	7803.862100	1824.942234	24.203597	1.741739e+06	976.935597	2.832690e+07	81218.391565	127.
-3_std	-5595.417655	-1437.645937	0.092700	-1.039773e+06	-599.341189	-1.608251e+07	-60638.280453	19.

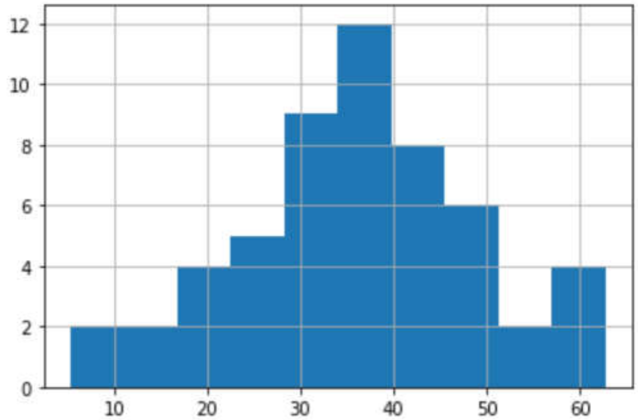
In [9]:
```
## Data preprocessing ##
## How is the distribution of the dependent variables?
```

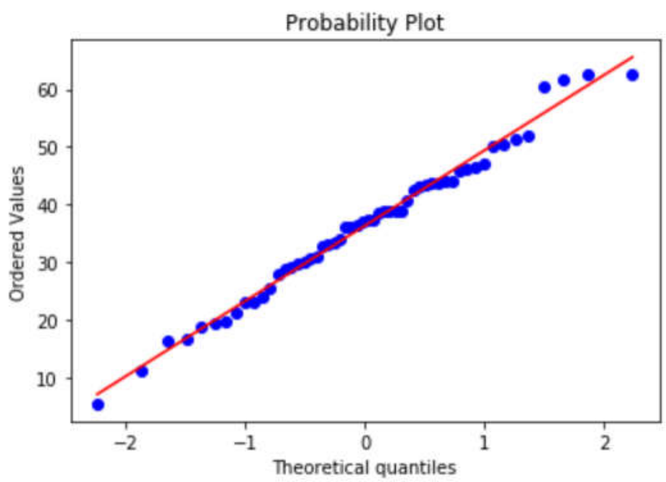
In [10]:
```
## Condisder CNCS
CNCS = covid19.CNCS
pd.Series(CNCS).hist()
plt.show()
stats.probplot(CNCS, dist="norm", plot=pylab)
pylab.show()
```





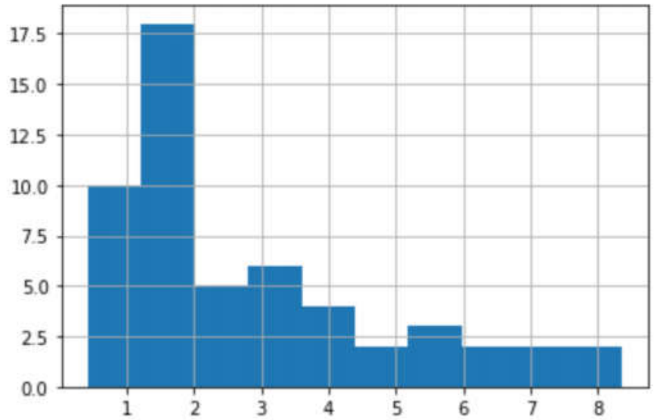
In [11]:
```
## Performing data transformation on this variable for normality
CNCS_bc, lmda = stats.boxcox(CNCS)
pd.Series(CNCS_bc).hist()
plt.show()
stats.probplot(CNCS_bc, dist = "norm", plot=pylab)
pylab.show()
print("lambda parameter for Box-Cox Transformation is {}".format(lmda))
```

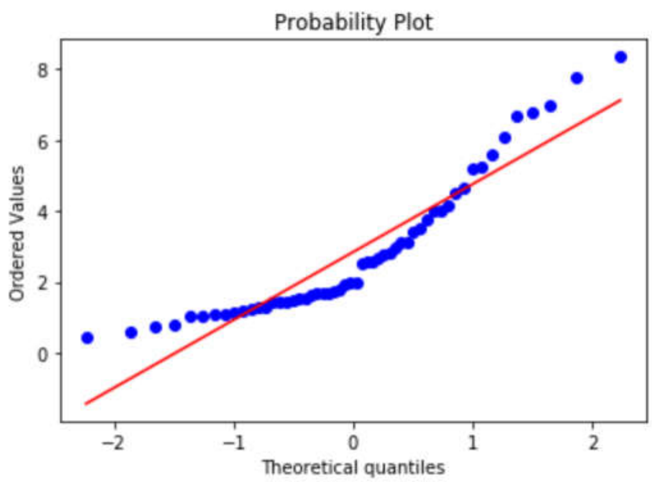




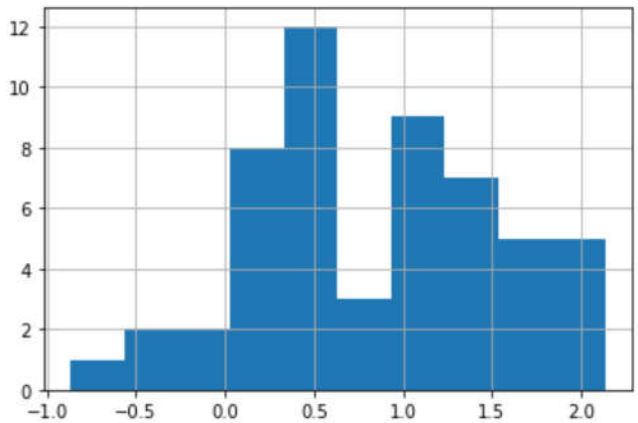

```
lambda parameter for Box-Cox Transformation is 0.20232519582590952
```

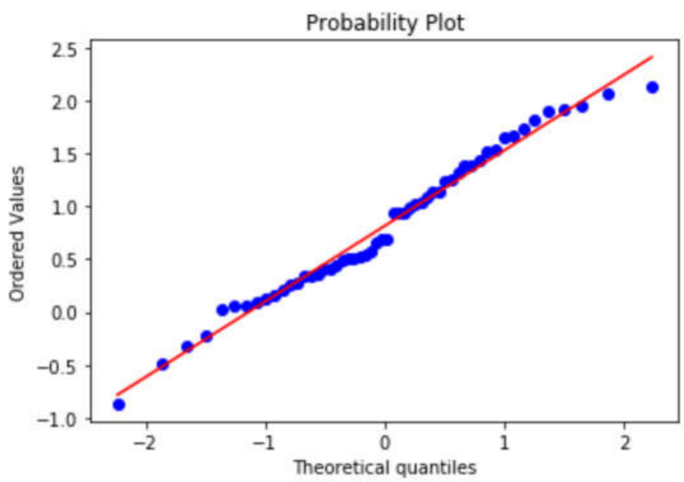
In [12]:
```
## Condisder MRAT
MRAT = covid19.MRAT
pd.Series(MRAT).hist()
plt.show()
stats.probplot(MRAT, dist="norm", plot=pylab)
pylab.show()
```





In [13]:
```
## Performing data transformation on this variable for normality
MRAT_bc, lmda = stats.boxcox(MRAT)
pd.Series(MRAT_bc).hist()
plt.show()
stats.probplot(MRAT_bc, dist = "norm", plot=pylab)
pylab.show()
print("lambda parameter for Box-Cox Transformation is {}".format(lmda))
```






```
lambda parameter for Box-Cox Transformation is 0.003882782809026342
```

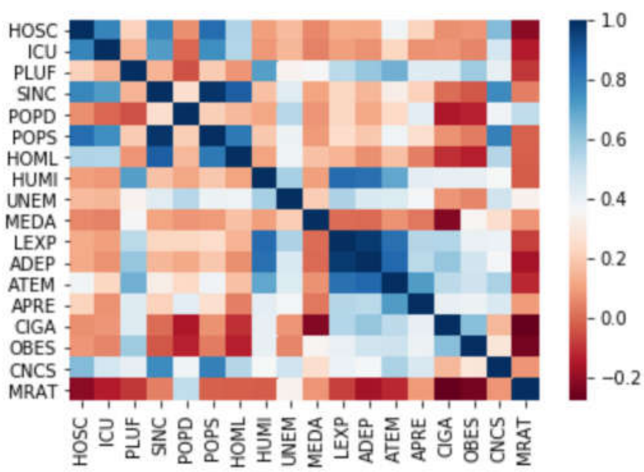
In [14]:
```
covid19["MRAT"] = MRAT_bc
covid19["CNCS"] = CNCS_bc
```

In [15]:
```
## Checking the Model Assumptions
######## Multicolinearity ################
## printing out correlation matrix of the data frame
corr=covid19.corr()
## Display the correlation matrix
display(corr)
```

	HOSC	ICU	PLUF	SINC	POPD	POPS	HOML	HUMI	UNEM	MED
HOSC	1.000000	0.783354	0.214592	0.778834	0.069699	0.846651	0.555732	0.098759	0.157446	0.05607
ICU	0.783354	1.000000	0.138199	0.719634	-0.007543	0.753193	0.548475	0.084027	0.147211	0.04533
PLUF	0.214592	0.138199	1.000000	0.144771	-0.043094	0.200162	0.078809	0.714648	0.345506	0.35768
SINC	0.778834	0.719634	0.144771	1.000000	0.254539	0.984750	0.889213	0.175348	0.432327	0.10786
POPD	0.069699	-0.007543	-0.043094	0.254539	1.000000	0.209218	0.153910	0.112114	0.537803	0.07914
POPS	0.846651	0.753193	0.200162	0.984750	0.209218	1.000000	0.818138	0.189892	0.392972	0.09159
HOML	0.555732	0.548475	0.078809	0.889213	0.153910	0.818138	1.000000	0.111731	0.390642	0.17410
HUMI	0.098759	0.084027	0.714648	0.175348	0.112114	0.189892	0.111731	1.000000	0.570755	0.10040
UNEM	0.157446	0.147211	0.345506	0.432327	0.537803	0.392972	0.390642	0.570755	1.000000	0.20359
MEDA	0.056074	0.045335	0.357682	0.107861	0.079143	0.091591	0.174102	0.100406	0.203597	1.00000
LEXP	0.124298	0.097515	0.533521	0.222027	0.226472	0.245149	0.144839	0.850670	0.559182	0.00212
ADEP	0.118983	0.073675	0.607745	0.148435	0.120808	0.189620	0.066961	0.835359	0.451502	0.00080
ATEM	0.378562	0.228042	0.667526	0.319816	0.231584	0.382708	0.174210	0.692897	0.449188	0.07585
APRE	0.217130	0.072459	0.437666	0.214580	0.423055	0.260359	0.038462	0.424430	0.372698	0.02496
CIGA	0.067580	0.078908	0.438371	0.003127	-0.152823	0.074792	-0.105630	0.408857	0.080647	-0.22357
OBES	0.080038	0.048655	0.587960	-0.035069	-0.131573	0.032612	-0.133165	0.409182	0.049998	0.34934
CNCS	0.635924	0.471339	0.408797	0.762188	0.384925	0.793436	0.544864	0.362717	0.486544	0.25632
MRAT	-0.208674	-0.138535	-0.090166	0.037871	0.520542	-0.014079	-0.019613	-0.026401	0.335970	0.07793

In [16]:
```
## plot a heatmap
sns.heatmap(corr, xticklabels = corr.columns, yticklabels = corr.columns, cmap="RdB
u")
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x24f6befb7c8>



In [52]:
```
### Using the VIF to measure to detect the above and dropping all variable with gre
ater than 10 VIF
covid19_before = covid19
covid19_after = covid19.drop(['SINC','POPS','HOML','LEXP','HOSC'], axis = 1)
x1 = sm.tools.add_constant(covid19_before)
x2 = sm.tools.add_constant(covid19_after)

#Create a series for both
series_before = pd.Series([variance_inflation_factor(x1.values, i) for i in range(x
1.shape[1])], index = x1.columns)
series_after = pd.Series([variance_inflation_factor(x2.values, i) for i in range(x
2.shape[1])], index = x2.columns)

## dispay the series
print('DATA BEFORE')
print('-'*100)
display(series_before)

print('DATA AFTER')
print('-'*100)
display(series_after)
```

```
DATA BEFORE
-------------------------------------------------------------------------------
--------------------

const     55.162561
HOSC      10.124455
ICU        3.527610
PLUF       7.779758
SINC     333.587585
POPD       3.405971
POPS     253.498730
HOML      23.741745
HUMI      12.113615
UNEM       3.217367
MEDA       2.885858
LEXP      68.271441
ADEP      56.743280
ATEM      13.095885
APRE       3.135952
CIGA       4.354244
OBES       3.902259
CNCS       6.506840
MRAT       2.262323
dtype: float64

DATA AFTER
-------------------------------------------------------------------------------
--------------------

const    49.460115
ICU       1.541582
PLUF      4.319143
POPD      2.567941
HUMI      7.396901
UNEM      2.789414
MEDA      2.493145
ADEP     12.665588
ATEM     10.440902
APRE      2.935465
CIGA      3.911578
OBES      3.469457
CNCS      2.431905
MRAT      1.704872
dtype: float64
```

In [53]:
```
covid19_after
```

Out[53]:

CDHS	ICU	PLUF	POPD	HUMI	UNEM	MEDA	ADEP	ATEM	APRE	CIGA	OBES	CNCS	MRAT
1265	0	16.8	96.9221	76.49	7.5	20	63.1	62.8	58.3	19.2	36.2	42.534352	0.566800
11	0	11.1	1.2863	81.46	12.4	21	55.8	26.6	22.5	19.1	29.5	19.352913	-0.866023
2443	870	14.1	64.9549	79.40	10.0	22	67.2	60.3	13.6	14.0	29.5	50.209267	0.484164
362	0	16.8	58.4030	76.92	8.0	27	66.4	60.4	50.6	22.7	37.1	36.167220	0.026708
7100	2284	12.8	256.3728	80.36	14.9	26	58.1	59.4	22.2	11.2	25.8	62.715700	0.540991
1643	0	9.7	56.4012	79.71	10.5	18	56.7	45.1	15.9	14.5	23.0	37.456771	1.390424
4031	0	10.3	735.8695	79.34	9.8	21	59.8	49.0	50.3	12.2	27.4	38.859101	2.132126
517	7	12.2	504.3073	72.02	12.5	21	64.0	55.3	45.7	16.5	33.5	29.059185	1.324765
644	18	16.1	0.0000	77.41	8.6	28	0.0	0.0	0.0	0.0	24.7	27.848323	1.726006
4341	0	13.7	410.1259	77.05	10.4	18	66.3	70.7	54.5	14.5	30.7	61.553029	0.134142
2547	0	14.5	186.6726	75.76	7.6	17	59.8	63.5	50.7	16.1	32.5	50.334143	0.514735
5	0	0.0	0.0000	0.00	0.0	0	0.0	0.0	0.0	21.9	29.8	11.034963	0.415851
20	0	9.0	219.9424	74.64	13.9	17	63.5	70.0	63.7	13.4	24.9	16.516663	0.344129
114	46	11.7	22.0970	79.51	5.6	17	69.7	44.4	18.9	14.7	28.4	30.017058	-0.327580
6652	337	12.1	228.0246	76.94	14.6	17	60.3	51.8	39.2	15.5	31.8	51.338360	1.388969
2733	327	13.0	188.2809	75.86	11.2	18	63.2	51.7	41.7	21.1	34.1	40.632311	1.543195
794	71	11.2	56.9284	82.01	8.0	19	65.8	47.8	34.0	16.6	35.3	37.188952	0.691735
315	112	11.9	35.5968	79.37	7.5	14	65.7	54.3	28.9	17.2	34.4	33.125200	0.267749
656	145	16.7	113.9566	76.42	4.3	26	62.2	55.6	48.9	23.4	36.6	33.263522	0.985151
3090	0	18.7	107.5174	75.71	9.7	29	62.0	66.4	60.1	20.5	36.8	45.757296	1.137155
130	8	11.6	43.6336	80.76	6.6	18	62.7	41.0	2.2	17.8	30.4	21.144914	1.253460
3622	137	9.1	626.6735	74.35	8.0	19	58.7	54.2	44.5	12.5	30.9	43.603902	1.512446
7753	63	10.0	894.4359	75.08	17.4	23	56.2	47.9	47.7	13.4	25.7	47.217224	1.921380
5596	210	14.0	177.6650	74.78	14.8	22	62.2	44.4	32.8	18.9	33.0	44.032289	1.906648
1484	119	9.6	71.5922	80.61	8.6	18	62.3	41.2	27.3	15.7	30.1	38.810847	1.132004
1206	293	19.8	63.7056	75.73	8.7	23	64.3	63.4	59.0	20.5	39.5	38.645345	0.942543
1016	0	13.2	89.7453	78.07	7.9	15	63.6	54.5	42.2	19.4	35.0	36.358262	1.037862
30	0	12.9	7.4668	80.40	7.1	21	65.3	42.7	15.3	18.0	26.9	19.706727	0.064369
286	0	11.0	25.4161	78.87	6.7	13	66.2	48.8	23.6	16.0	34.1	32.828625	0.209771
577	299	13.1	28.5993	78.26	15.0	19	61.5	49.9	9.5	15.7	29.5	36.942784	0.400841
383	0	7.6	153.1610	81.86	11.8	14	57.5	43.8	43.3	15.6	29.6	24.038954	1.817332
13811	151	9.5	1215.1985	71.31	16.6	17	60.3	52.7	47.1	13.1	25.7	52.082668	2.059036
516	0	18.8	17.2850	76.63	8.3	33	66.5	53.4	14.6	15.2	32.3	30.744364	1.082619
11242	179	13.7	412.5218	75.60	15.7	26	58.1	45.4	41.8	12.8	27.6	62.559759	1.013379
1222	338	14.1	218.2710	77.05	7.6	18	61.4	59.0	50.3	17.4	33.0	46.327540	0.151734
104	0	10.6	11.0393	80.74	6.1	12	60.5	40.4	17.8	19.1	35.1	23.148559	0.662387
2	0	0.0	0.0000	0.00	0.0	0	0.0	0.0	0.0	0.0	0.0	5.375094	1.666097
2703	347	13.8	287.5040	77.91	10.9	21	63.3	50.7	39.1	20.5	34.0	43.427447	1.236314
421	257	15.5	57.6546	76.76	6.6	18	65.3	59.6	36.5	19.7	34.8	34.096645	0.433489

In [54]:
```
#### Building the model ####
## considering CNCS as our dependent Variable ##
## define our input variable and our output variable where ###
x = covid19_after.drop(['CNCS', 'MRAT'], axis = 1)
y = covid19_after['CNCS']
```

In [55]:
```
## Split dataset into training and testing portion
from sklearn.model_selection import train_test_split
import numpy as np

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_
state = 1)

## Scale the independent variables gives
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler= preprocessing.MinMaxScaler()
x_train_minmax = min_max_scaler.fit_transform(x_train)
x_test_minmax = min_max_scaler.fit_transform(x_test)
```

In [56]:
```
x_train = x_train_minmax
x_test= x_test_minmax
```

In [57]:
```
## Create an instance of our model
regression_model = LinearRegression()

## Fit the model
regression_model.fit(x_train, y_train)
```

Out[57]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [58]:
```
## Getting multiple prediction
y_predict = regression_model.predict(x_test)
## Show the first five
y_predict[:5]
```

Out[58]:
```
array([13.36137701, 32.5885226 ,  6.77703101, 31.03883019, 42.33630306])
```

In [59]:
```
## Evaluating the model
import statsmodels.api as sm
from statsmodels.stats import diagnostic as diag
from statsmodels.stats.outliers_influence import variance_inflation_factor

## Define our input variable
x2 = sm.add_constant(x)

## Create an OLS model
model = sm.OLS(y, x2)
## fit the data
est = model.fit()
```

In [60]:

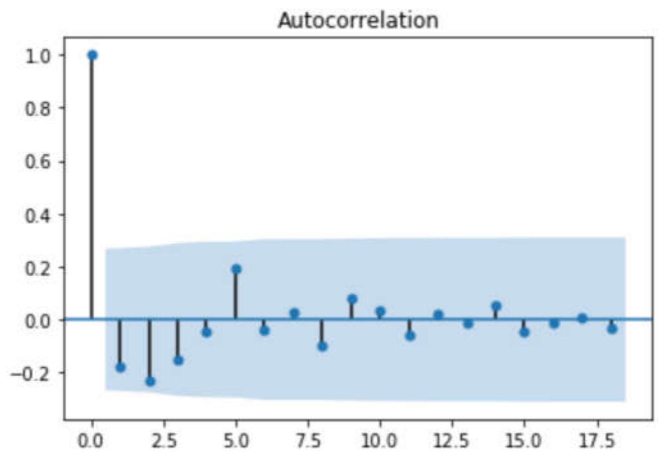
```
## Testing the Model Assumptions
# Heteroscedasticity using the Breusch-Pegan test
#H0:σ2=σ2
#H1:σ2!=σ2

## Grab the p-values
_, pval, _, f_pval = diag.het_breuschpagan(est.resid, est.model.exog)
print(pval, f_pval)
print('_'*100)
if pval > 0.05:
    print("For the Breusch Pagan's Test")
    print("The p-value was {:.4}".format(pval))
    print("we fail to reject the null hypothesis, and conclude that there is no het
eroscedasticity.")
else:
    print("For the Breusch Pagan's Test")
    print("The p-value was {:.4}".format(pval))
    print("we reject the null hypothesis, and conclude that there is heteroscedasti
city.")
```

```
0.06811122202682957 0.05174166821430557
________________________________________________________________________________
____________________
For the Breusch Pagan's Test
The p-value was 0.06811
we fail to reject the null hypothesis, and conclude that there is no heterosceda
sticity.
```

In [61]:

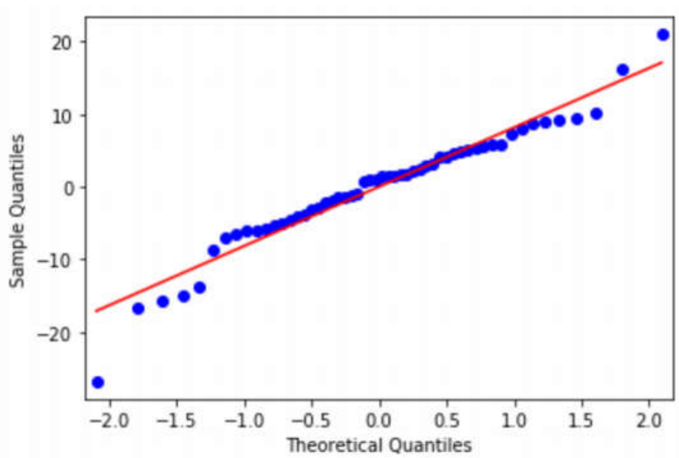
```
### Checking for Autocorrelation using the Ljungbox test
#H0: The data are random
#H1: The data are not random
## Calculate the lag
lag = min(10, (len(x)//5))
print('The number of lags will be {}'.format(lag))
print('_'*100)

## Perform the test
test_results = diag.acorr_ljungbox(est.resid, lags = lag)
## print the result for the test
print(test_results)

## Grab the P-Value and the test statistics
ibvalue, p_val = test_results

## print the result for the test
if min(p_val) > 0.05:
    print("The lowest p-value found was {:.4}".format(min(p_val)))
    print("we fail to reject the null hypothesis, and conclude that there is no Aut
ocorrelation.")
    print('_'*100)
else:
    print("The lowest p-value found was {:.4}".format(min(p_val)))
    print("we reject the null hypothesis, and conclude that there is Autocorrelatio
n.")
    print('_'*100)

## Plotting Autocorrelation
import matplotlib.pyplot as plt
from scipy import stats
import statsmodels.api as sm
from statsmodels.stats import diagnostic as diag
sm.graphics.tsa.plot_acf(est.resid)
plt.show()
```

```
The number of lags will be 10
____________________________________________________________________________________________________
(array([1.74934168, 4.85216623, 6.20121829, 6.32270801, 8.61634465,
       8.71243281, 8.75217815, 9.37804163, 9.79834729, 9.87389321]), array([0.18
595951, 0.08838234, 0.10222052, 0.17630964, 0.12538064,
       0.19040819, 0.27094176, 0.31141494, 0.36705559, 0.45162596]))
The lowest p-value found was 0.08838
we fail to reject the null hypothesis, and conclude that there is no Autocorrela
tion.
____________________________________________________________________________________________________

C:\Users\AGYEMANG ERIC\anaconda3\lib\site-packages\statsmodels\stats\diagnostic.
py:524: FutureWarning: The value returned will change to a single DataFrame afte
r 0.12 is released.  Set return_df to True to use to return a DataFrame now.  Se
t return_df to False to silence this warning.
  warnings.warn(msg, FutureWarning)
```



In [62]:

```python
## Check for Linearity of the residuals using the Q-Q plot
import pylab
sm.qqplot(est.resid, line = 's')
pylab.show()

## Checking that mean of the residuals is approximately zero
mean_residuals = sum(est.resid)/len(est.resid)
mean_residuals
```



Out[62]: -1.7829359388054364e-14

In [63]:
```
## Model summary
print(est.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   CNCS   R-squared:                       0.589
Model:                            OLS   Adj. R-squared:                  0.481
Method:                 Least Squares   F-statistic:                     5.462
Date:                Wed, 21 Oct 2020   Prob (F-statistic):           2.50e-05
Time:                        01:56:00   Log-Likelihood:                -190.05
No. Observations:                  54   AIC:                             404.1
Df Residuals:                      42   BIC:                             428.0
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          6.7159      8.416      0.798      0.429     -10.269      23.701
ICU            0.0080      0.003      3.050      0.004       0.003       0.013
PLUF           0.1181      0.656      0.180      0.858      -1.206       1.442
POPD           0.0103      0.007      1.466      0.150      -0.004       0.025
HUMI           0.0073      0.191      0.038      0.970      -0.378       0.392
UNEM           0.6344      0.549      1.156      0.254      -0.473       1.742
MEDA           0.1111      0.279      0.399      0.692      -0.451       0.673
ADEP          -0.2184      0.265     -0.824      0.414      -0.753       0.316
ATEM           0.4171      0.247      1.688      0.099      -0.081       0.916
APRE          -0.0048      0.124     -0.039      0.969      -0.255       0.246
CIGA          -0.1614      0.507     -0.318      0.752      -1.185       0.862
OBES           0.3678      0.410      0.897      0.375      -0.460       1.195
==============================================================================
Omnibus:                        7.383   Durbin-Watson:                   2.334
Prob(Omnibus):                  0.025   Jarque-Bera (JB):                7.595
Skew:                          -0.560   Prob(JB):                       0.0224
Kurtosis:                       4.457   Cond. No.                     3.87e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.87e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [ ]:

In [ ]:

In [64]:
```
#### Building the model ####
## considering MRAT as our dependent Variable ##
## define our input variable and our output variable where ###
x = covid19_after.drop(['CNCS', 'MRAT'], axis = 1)
y = covid19_after['MRAT']
```

In [65]:
```
## Split dataset into training and testing portion
from sklearn.model_selection import train_test_split
import numpy as np

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_
state = 1)

## Scale the independent variables gives
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
import numpy as np

min_max_scaler= preprocessing.MinMaxScaler()
x_train_minmax = min_max_scaler.fit_transform(x_train)
x_test_minmax = min_max_scaler.fit_transform(x_test)
```

In [66]:
```
x_train = x_train_minmax
x_test= x_test_minmax
```

In [67]:
```
## Create an instance of our model
regression_model = LinearRegression()

## Fit the model
regression_model.fit(x_train, y_train)
```

Out[67]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [68]:
```
## Getting multiple prediction
y_predict = regression_model.predict(x_test)
## Show the first five
y_predict[:5]
```

Out[68]:
```
array([1.11697455, 0.56256124, 1.05272564, 0.80999288, 2.50372713])
```

In [69]:
```
## Evaluating the model
import statsmodels.api as sm
from statsmodels.stats import diagnostic as diag
from statsmodels.stats.outliers_influence import variance_inflation_factor

## Define our input variable
x2 = sm.add_constant(x)

## Create an OLS model
model = sm.OLS(y, x2)
## fit the data
est = model.fit()
```

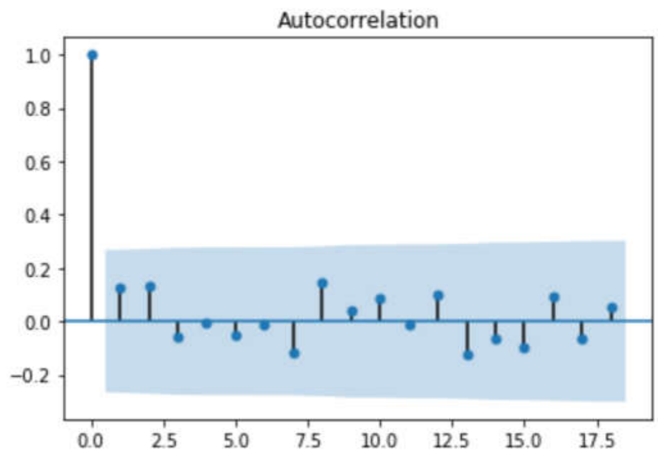
In [70]:

```
## Testing the Model Assumptions
# Heteroscedasticity using the Breusch-Pegan test
#H0:σ2=σ2
#H1:σ2!=σ2

## Grab the p-values
_, pval, _, f_pval = diag.het_breuschpagan(est.resid, est.model.exog)
print(pval, f_pval)
print('_'*100)
if pval > 0.05:
    print("For the Breusch Pagan's Test")
    print("The p-value was {:.4}".format(pval))
    print("we fail to reject the null hypothesis, and conclude that there is no het
eroscedasticity.")
else:
    print("For the Breusch Pagan's Test")
    print("The p-value was {:.4}".format(pval))
    print("we reject the null hypothesis, and conclude that there is heteroscedasti
city.")
```

```
0.19496889194734351 0.19445491270187962
________________________________________________________________________________
____________________
For the Breusch Pagan's Test
The p-value was 0.195
we fail to reject the null hypothesis, and conclude that there is no heterosceda
sticity.
```

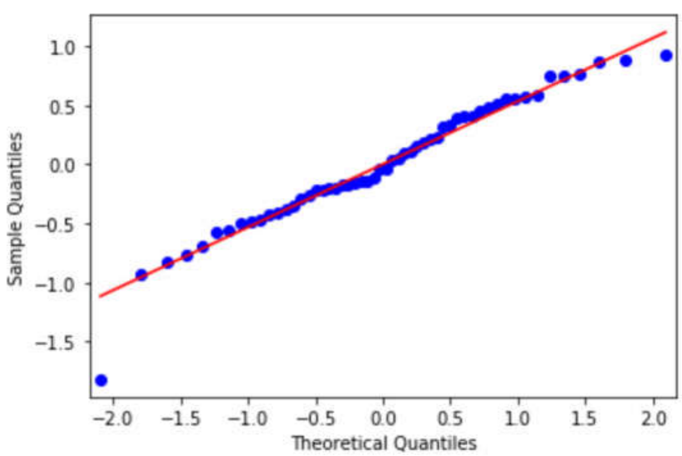
In [71]:
```
### Checking for Autocorrelation using the Ljungbox test
#H0: The data are random
#H1: The data are not random
## Calculate the lag
lag = min(10, (len(x)//5))
print('The number of lags will be {}'.format(lag))
print('_'*100)

## Perform the test
test_results = diag.acorr_ljungbox(est.resid, lags = lag)
## print the result for the test
print(test_results)

## Grab the P-Value and the test statistics
ibvalue, p_val = test_results

## print the result for the test
if min(p_val) > 0.05:
    print("The lowest p-value found was {:.4}".format(min(p_val)))
    print("we fail to reject the null hypothesis, and conclude that there is no Aut
ocorrelation.")
    print('_'*100)
else:
    print("The lowest p-value found was {:.4}".format(min(p_val)))
    print("we reject the null hypothesis, and conclude that there is Autocorrelatio
n.")
    print('_'*100)

## Plotting Autocorrelation
import matplotlib.pyplot as plt
from scipy import stats
import statsmodels.api as sm
from statsmodels.stats import diagnostic as diag
sm.graphics.tsa.plot_acf(est.resid)
plt.show()
```

```
The number of lags will be 10
_______________________________________________________________________________
____________________
(array([0.95212652, 2.00227003, 2.20090099, 2.20273294, 2.38205494,
       2.39558071, 3.26256125, 4.64234634, 4.73793016, 5.26275146]), array([0.32
91786 , 0.36746213, 0.53177093, 0.69852896, 0.79414346,
       0.87996593, 0.85969769, 0.79502865, 0.85652907, 0.87294901]))
The lowest p-value found was 0.3292
we fail to reject the null hypothesis, and conclude that there is no Autocorrela
tion.
_______________________________________________________________________________
____________________

C:\Users\AGYEMANG ERIC\anaconda3\lib\site-packages\statsmodels\stats\diagnostic.
py:524: FutureWarning: The value returned will change to a single DataFrame afte
r 0.12 is released.  Set return_df to True to use to return a DataFrame now.  Se
t return_df to False to silence this warning.
  warnings.warn(msg, FutureWarning)
```



In [72]:

```python
## Check for Linearity of the residuals using the Q-Q plot
import pylab
sm.qqplot(est.resid, line = 's')
pylab.show()

## Checking that mean of the residuals is approximately zero
mean_residuals = sum(est.resid)/len(est.resid)
mean_residuals
```



Out[72]: -1.0392921091629937e-15

In [73]:
```python
## Model summary
print(est.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MRAT   R-squared:                       0.413
Model:                            OLS   Adj. R-squared:                  0.259
Method:                 Least Squares   F-statistic:                     2.688
Date:                Wed, 21 Oct 2020   Prob (F-statistic):             0.0104
Time:                        02:05:03   Log-Likelihood:                -42.660
No. Observations:                  54   AIC:                             109.3
Df Residuals:                      42   BIC:                             133.2
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.0834      0.549      1.972      0.055      -0.025       2.192
ICU           -0.0002      0.000     -1.011      0.318      -0.001       0.000
PLUF           0.0204      0.043      0.476      0.636      -0.066       0.107
POPD           0.0012      0.000      2.538      0.015       0.000       0.002
HUMI           0.0068      0.012      0.544      0.589      -0.018       0.032
UNEM           0.0370      0.036      1.033      0.308      -0.035       0.109
MEDA           0.0009      0.018      0.049      0.961      -0.036       0.038
ADEP          -0.0159      0.017     -0.921      0.362      -0.051       0.019
ATEM          -0.0068      0.016     -0.423      0.675      -0.039       0.026
APRE           0.0028      0.008      0.351      0.727      -0.014       0.019
CIGA           0.0036      0.033      0.109      0.914      -0.063       0.070
OBES          -0.0153      0.027     -0.570      0.572      -0.069       0.039
==============================================================================
Omnibus:                        5.327   Durbin-Watson:                   1.727
Prob(Omnibus):                  0.070   Jarque-Bera (JB):                4.325
Skew:                          -0.544   Prob(JB):                        0.115
Kurtosis:                       3.859   Cond. No.                     3.87e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.87e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [ ]:

In [ ]: