WashburnPaulHW3

October 22, 2018

0.0.1 CSCI E-82 Homework 3

0.0.2 Due by 10/22/18 at 11:59pm EST to the Canvas dropbox

0.1 This is an individual homework so there should be no collaboration for this homework.

0.2 ### Under each problem, we have a place for you to write the answer, or write runnable code that will produce the answer. Show your work.

0.3 Your Name:

Paul Washburn

0.4 Problem 1 Climate Change (30 points)

Scientists and politicians are often at odds on the topic of whether global warming is real and debate the various causes. This problem uses "globalWarm3.csv" data. This is a real data set.

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import statsmodels.api as sm
        from sklearn.base import BaseEstimator, TransformerMixin
        from sklearn.pipeline import Pipeline
        from sklearn.linear_model import LinearRegression
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import r2_score
        from sklearn.preprocessing import StandardScaler
        from sklearn.preprocessing import PolynomialFeatures
        from sklearn.preprocessing import StandardScaler
        from statsmodels.graphics.gofplots import qqplot
        from matplotlib import pyplot as plt
        from yellowbrick.regressor import ResidualsPlot
        import statsmodels.tsa.api as smt
        import warnings
        import seaborn as sns
        from statsmodels.tsa.seasonal import seasonal_decompose
        from matplotlib.colors import ListedColormap
        import itertools
```

```python
from statsmodels.tsa.stattools import adfuller
warnings.filterwarnings('ignore')
%matplotlib inline

def set_mpl_preferences(ax):
    ax.grid(alpha=.4)
    sns.despine()

def tsplot(y, lags=None, figsize=(12, 12)):

    q, p = sm.stats.diagnostic.acorr_ljungbox(y,lags)
    fig = plt.figure(figsize=figsize)
    layout = (4, 2)
    ts_ax = plt.subplot2grid(layout, (0, 0), colspan=2)
    acf_ax = plt.subplot2grid(layout, (1, 0))
    pacf_ax = plt.subplot2grid(layout, (1, 1))
    qq_ax = plt.subplot2grid(layout, (2, 0), colspan=2, title='QQ plot')
    lbox_ax = plt.subplot2grid(layout, (3, 0), colspan=2, title='Ljung-Box statistic')

    y.plot(ax=ts_ax, title='Given')
    smt.graphics.plot_acf(y, lags=lags, ax=acf_ax)
    smt.graphics.plot_pacf(y, lags=lags, ax=pacf_ax)

    qqplot(y,  line='q', ax=qq_ax, fit=True)
    if np.max(p) > 0.05:
        lbox_ax.axhline(y=0.05, xmin=0, xmax=lags, c ='r')
    lbox_ax.plot(p)
    sns.despine()
    plt.tight_layout()
    plt.show()
    return ts_ax, acf_ax, pacf_ax, lbox_ax

dark2_colors = [(0.10588235294117647, 0.6196078431372549, 0.4666666666666667),
                (0.9058823529411765, 0.1607843137254902, 0.5411764705882353),
                (0.8509803921568627, 0.37254901960784315, 0.00784313725490196),
                (0.4588235294117647, 0.4392156862745098, 0.7019607843137254),
                (0.4, 0.6509803921568628, 0.11764705882352941),
                (0.9019607843137255, 0.6705882352941176, 0.00784313725490196),
                (0.6509803921568628, 0.4627450980392157, 0.11372549019607843)]

cmap_set1 = ListedColormap(['#e41a1c', '#377eb8', '#4daf4a'])
dark2_cmap=ListedColormap(dark2_colors)
```

In [2]:
```python
# read in gw data
gw = pd.read_csv('data/globalWarm3.csv')
gw.set_index('Year', inplace=True)

# get log
```

```
gw['log_temp'] = np.log(gw.Temp)

# fill na
not_missing = ~gw.Transmission.isnull()
gw.loc[~not_missing, 'Transmission'] = gw.loc[not_missing, 'Transmission'].mean()

gw.head()
```

Out[2]:

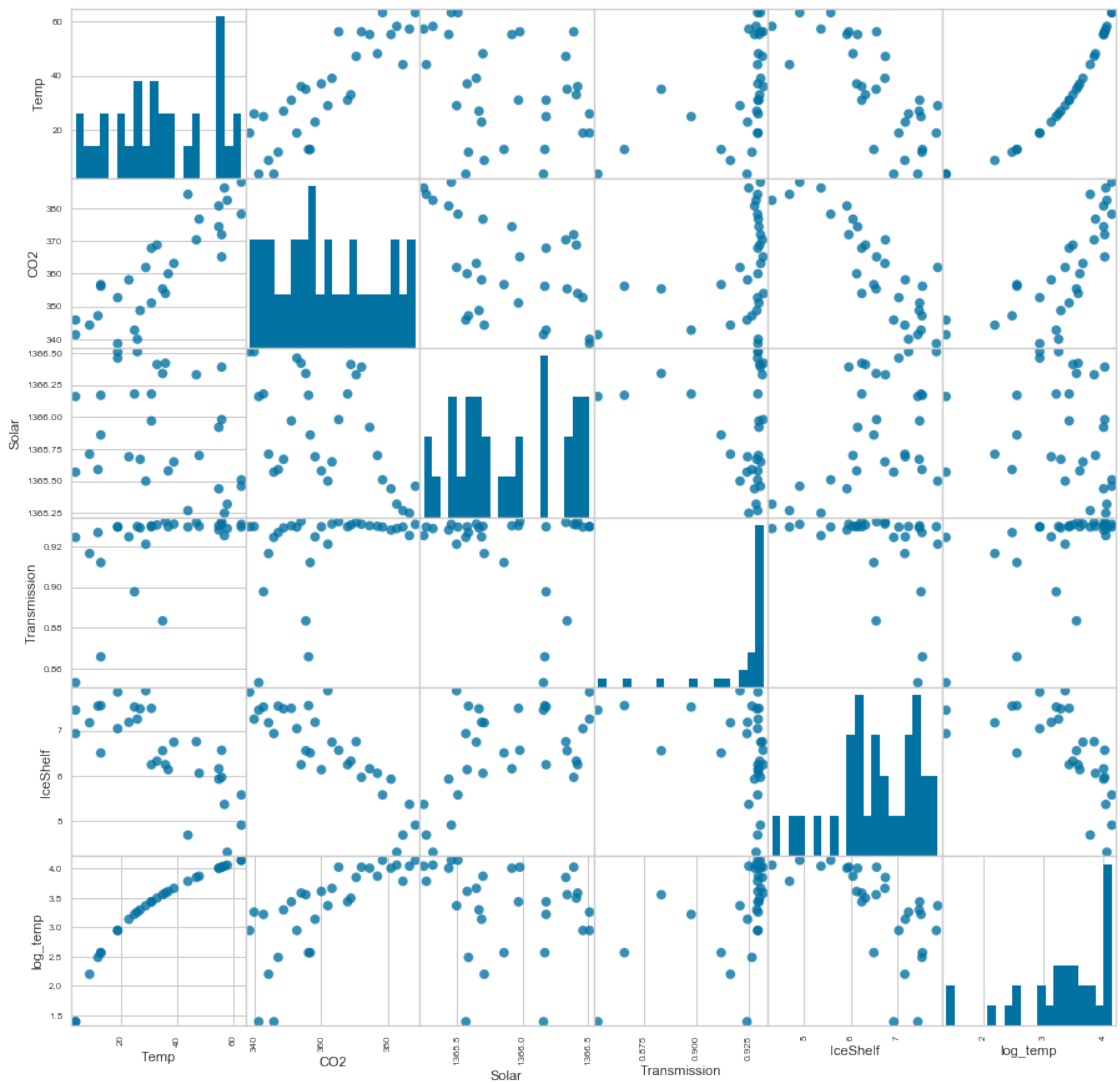
	Temp	CO2	Solar	Transmission	IceShelf	log_temp
Year						
1980	19	338.57	1366.51	0.929667	7.85	2.944439
1981	26	339.92	1366.51	0.929767	7.25	3.258097
1982	4	341.30	1366.16	0.853067	7.45	1.386294
1983	25	342.71	1366.18	0.897717	7.52	3.218876
1984	9	344.24	1365.71	0.916492	7.17	2.197225

0.4.1 Problem 1a

Plot a scatter plot of the following variables in a lattice: Temp, CO2, Solar, Transmission, and IceShelf.

The variables represent the following: - Temp = annual surface temperature measured in $1/100°C$ over the 1950-1980 mean. - Solar = annual mean intensity of sunlight piercing the atmosphere - CO2 = annual average fraction CO2 in atmosphere (#molecules/#molecules of dry air) - IceShelf = sea ice in 1MM square miles hypothesized to reflect heat - Transmission = volcanic MLO transmission data where eruptions release greenhouse gases but also decrease the temperature

In [3]:
```
pd.plotting.scatter_matrix(gw, figsize=(15, 15), marker='o',
                                 hist_kwds={'bins': 20}, s=60, alpha=.8)
plt.show()
```



0.4.2 Problem 1b

Compute a multiple linear regression model of log(Temp) against the other variables. Note that since there are limited number of annual measurements, you cannot run all combinations of variables. In fact, you can only do complete pairwise interactions. Be sure to remove the non-significant variables while still maintaining the hierarchy principle in your final model. You do not need to show full diagnostics for the different models that you try, but do show the equations that you tried.

Isolate `X` & `y` and Add Interaction Terms

```
In [4]: X_cols = ['CO2', 'Solar', 'Transmission', 'IceShelf']
        X, y = gw[X_cols], gw['log_temp']
```

```
# fit polynomial features
poly = PolynomialFeatures(2, include_bias=False)
poly.fit(X)
X = pd.DataFrame(poly.transform(X), columns=poly.get_feature_names(X.columns))

# preview
X.head()
```

Out[4]:

```
      CO2    Solar  Transmission  IceShelf        CO2^2    CO2 Solar  \
0  338.57  1366.51      0.929667      7.85  114629.6449  462659.2907
1  339.92  1366.51      0.929767      7.25  115545.6064  464504.0792
2  341.30  1366.16      0.853067      7.45  116485.6900  466270.4080
3  342.71  1366.18      0.897717      7.52  117450.1441  468203.5478
4  344.24  1365.71      0.916492      7.17  118501.1776  470132.0104

   CO2 Transmission  CO2 IceShelf       Solar^2  Solar Transmission  \
0        314.757243     2657.7745  1.867350e+06         1270.398797
1        316.046285     2464.4200  1.867350e+06         1270.535448
2        291.151653     2542.6850  1.866393e+06         1165.425558
3        307.656479     2577.1792  1.866448e+06         1226.442556
4        315.493091     2468.2008  1.865164e+06         1251.661835

   Solar IceShelf  Transmission^2  Transmission IceShelf  IceShelf^2
0      10727.1035        0.864280               7.297883     61.6225
1       9907.1975        0.864466               6.740808     52.5625
2      10177.8920        0.727723               6.355347     55.5025
3      10273.6736        0.805895               6.750829     56.5504
4       9792.1407        0.839957               6.571245     51.4089
```

Fit Model with All Variables and Interactions

In [5]:

```
# fit using statsmodels to get access to more metrics
#X_cols = ['CO2', 'Solar', 'Transmission', 'IceShelf']

# scale with standard scaler & add constant
X = pd.DataFrame(StandardScaler().fit_transform(X), columns=X.columns)
X = sm.add_constant(X)

lm = sm.OLS(y.values.reshape(-1), X)
results = lm.fit()

print(results.summary(), '\n')
print('Parameters: ')
print(results.params, '\n')
print('R2: %.3f' %results.rsquared)
```

```
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:                      y   R-squared:                       0.891
Model:                            OLS   Adj. R-squared:                  0.795
Method:                 Least Squares   F-statistic:                     9.306
Date:                Mon, 22 Oct 2018   Prob (F-statistic):           3.34e-05
Time:                        19:53:21   Log-Likelihood:                0.13549
No. Observations:                  31   AIC:                             29.73
Df Residuals:                      16   BIC:                             51.24
Df Model:                          14
Covariance Type:            nonrobust
=========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
const                     3.3383      0.060     55.427      0.000       3.211       3.466
CO2                    1438.1413    536.315      2.682      0.016     301.205    2575.078
Solar                  1079.1572    998.475      1.081      0.296   -1037.514    3195.829
Transmission           2891.4229    972.750      2.972      0.009     829.285    4953.561
IceShelf                694.4330    589.687      1.178      0.256    -555.647    1944.513
CO2^2                    -6.6612     15.687     -0.425      0.677     -39.916      26.594
CO2 Solar             -1429.8939    525.945     -2.719      0.015   -2544.847    -314.941
CO2 Transmission          4.7096      9.947      0.473      0.642     -16.377      25.796
CO2 IceShelf              1.7116      8.748      0.196      0.847     -16.834      20.257
Solar^2               -1026.6651    999.880     -1.027      0.320   -3146.316    1092.986
Solar Transmission    -2868.4520    966.826     -2.967      0.009   -4918.032    -818.872
Solar IceShelf         -690.5105    583.463     -1.183      0.254   -1927.397     546.376
Transmission^2          -14.7768      8.791     -1.681      0.112     -33.412       3.858
Transmission IceShelf    -7.8814      9.841     -0.801      0.435     -28.744      12.981
IceShelf^2                1.6476      2.580      0.639      0.532      -3.822       7.117
==============================================================================
Omnibus:                        1.750   Durbin-Watson:                   2.482
Prob(Omnibus):                  0.417   Jarque-Bera (JB):                1.060
Skew:                          -0.451   Prob(JB):                        0.589
Kurtosis:                       3.089   Cond. No.                     7.78e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.78e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Parameters:
const                      3.338310
CO2                     1438.141349
Solar                   1079.157185
Transmission            2891.422905
IceShelf                 694.432992
CO2^2                     -6.661171
CO2 Solar              -1429.893859
CO2 Transmission           4.709600
```

```
CO2 IceShelf               1.711557
Solar^2                -1026.665099
Solar Transmission     -2868.451956
Solar IceShelf          -690.510488
Transmission^2           -14.776784
Transmission IceShelf     -7.881407
IceShelf^2                 1.647635
dtype: float64

R2: 0.891
```

Observe p-values and Select Significant Variables from Above Model

In [6]:
```python
p_df = pd.DataFrame({'coef': results.params, 'p_value': results.pvalues})
p_df = p_df.apply(lambda f: round(f, 3))
p_df
```

Out[6]:

	coef	p_value
const	3.338	0.000
CO2	1438.141	0.016
Solar	1079.157	0.296
Transmission	2891.423	0.009
IceShelf	694.433	0.256
CO2^2	-6.661	0.677
CO2 Solar	-1429.894	0.015
CO2 Transmission	4.710	0.642
CO2 IceShelf	1.712	0.847
Solar^2	-1026.665	0.320
Solar Transmission	-2868.452	0.009
Solar IceShelf	-690.510	0.254
Transmission^2	-14.777	0.112
Transmission IceShelf	-7.881	0.435
IceShelf^2	1.648	0.532

In [7]:
```python
p_df = p_df.loc[p_df.p_value <= .05]
print('''
Significant Variables
''')
p_df
```

```
Significant Variables
```

Out[7]:

	coef	p_value
const	3.338	0.000
CO2	1438.141	0.016

```
Transmission        2891.423    0.009
CO2 Solar          -1429.894    0.015
Solar Transmission -2868.452    0.009
```

Isolate `X` Columns that are Significant `Solar` is added back in because we cannot have interactions with `Solar` without accounting for it by itself.

```
In [8]: signif_X = p_df.index.values.tolist() + ['Solar']
```

Fit `sklearn` Version of Model

```
In [9]: # re-specify X with significant columns
        X = X[signif_X]

        pipeline = Pipeline([
            ('lm', LinearRegression(fit_intercept=0))
        ])

        pipeline.fit(X, y)
        y_pred = pipeline.predict(X)
        r2 = r2_score(y, y_pred)

        print('''
        R2 score = %.3f
        ''' %(r2))
```

```
R2 score = 0.792
```

0.4.3 Problem 1c

Run the diagnostics to determine whether your final model is appropriate.

Fit `statsmodels.api.OLS` with Significant `X` Columns & Observe Summary

```
In [10]: lm = sm.OLS(y.values.reshape(-1), X)
         results = lm.fit()

         print(results.summary(), '\n')
         print('Parameters: ')
         print(results.params, '\n')
         print('R2: %.3f' %results.rsquared)
```

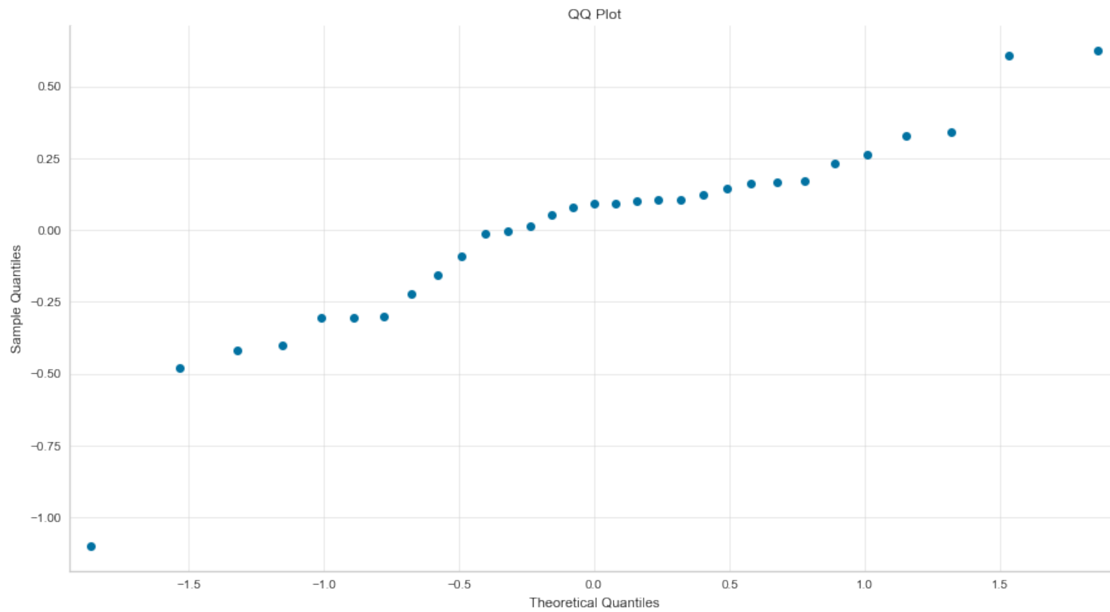
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.792
```

```
Model:                            OLS   Adj. R-squared:                  0.751
Method:                 Least Squares   F-statistic:                     19.07
Date:                Mon, 22 Oct 2018   Prob (F-statistic):           8.19e-08
Time:                        19:53:21   Log-Likelihood:                -9.8078
No. Observations:                  31   AIC:                             31.62
Df Residuals:                      25   BIC:                             40.22
Df Model:                           5
Covariance Type:            nonrobust
========================================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
const                  3.3383      0.066     50.273      0.000       3.202       3.475
CO2                  248.2254    225.959      1.099      0.282    -217.146     713.597
Transmission        2341.3006    841.312      2.783      0.010     608.586    4074.015
CO2 Solar           -246.8360    225.129     -1.096      0.283    -710.497     216.825
Solar Transmission -2334.2245    838.858     -2.783      0.010   -4061.885    -606.564
Solar                 34.9008     11.714      2.979      0.006      10.775      59.027
==============================================================================
Omnibus:                       10.462   Durbin-Watson:                   2.038
Prob(Omnibus):                  0.005   Jarque-Bera (JB):               10.416
Skew:                          -0.950   Prob(JB):                      0.00547
Kurtosis:                       5.111   Cond. No.                     3.09e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.09e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Parameters:
const                    3.338310
CO2                    248.225444
Transmission          2341.300603
CO2 Solar             -246.835963
Solar Transmission   -2334.224511
Solar                   34.900824
dtype: float64

R2: 0.792
```

Plot QQ Plot

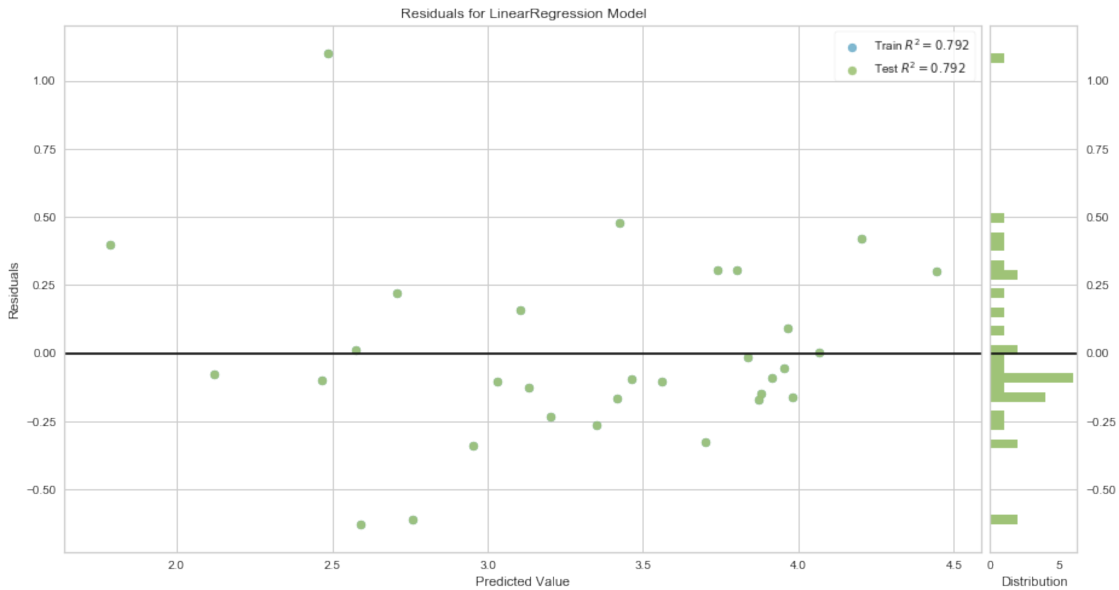
```
In [11]: res = results.resid

         fig, ax = plt.subplots(figsize=(15, 8))
         fig = sm.qqplot(res, ax=ax)
         set_mpl_preferences(ax)
```

```
ax.set_title('QQ Plot')
plt.show()
```



Plot Residuals Plot

```
In [12]: fig, ax = plt.subplots(figsize=(15, 8))

         # Instantiate the linear model and visualizer
         visualizer = ResidualsPlot(LinearRegression())
         visualizer.fit(X, y)  # Fit the training data to the model
         visualizer.score(X, y)  # Evaluate the model on the test data
         visualizer.poof(ax=ax)
         set_mpl_preferences(ax)
         plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```

0.4.4 Problem 1d

Describe in what way the model diagnostics are appropriate or not. Be specific.

Both `CO2` and `CO2 Solar` became non-significant when modeled without the original factors, and may have just introduced some noise. We can see by the histogram on the right side of the residuals plot that there is not enough data to give it a label of Normal distribution. The QQ plot appears reasonable for this model as well, despite its lack of data points. According to this model, `Solar`, `Transmission` and `Solar Transmission` are significant predictors of `log_temp`.

0.4.5 Problem 1e

Using your knowledge of statistics, what would you conclude about climate change?

I'd conclude that we need some mroe data! I also find it surprising that `CO2` became non-significant when adjusting the accounted-for-variables list. Like any good scientist, I wouldn't draw any set-in-stone conclusions in particular from this surface analysis of a limited dataset.

0.5 Problem 2 Matrix model for regression (8 points)

0.5.1 Problem 2a

Using the features that you deemed important in Problem 1, construct the matrix forms of the appropriate variables. Specifically you will need a matrix X that has the features used in your solution and a Y = logTemp. Print the head of each of these.

```
In [13]: print('''
        X and y have been segregated, scaled and processed already.
```

```
''')

print('X: \n', X.head(), '\n')
print('y: \n', y.head())
```

```
X and y have been segregated, scaled and processed already.

X:
    const       CO2  Transmission  CO2 Solar  Solar Transmission     Solar
0    1.0 -1.579838      0.446017  -1.575164            0.469142  1.545449
1    1.0 -1.487757      0.451212  -1.482695            0.474354  1.545449
2    1.0 -1.393629     -3.533328  -1.394159           -3.534160  0.667996
3    1.0 -1.297456     -1.213775  -1.297261           -1.207191  0.718136
4    1.0 -1.193097     -0.238420  -1.200598           -0.245418 -0.460157

y:
 Year
1980    2.944439
1981    3.258097
1982    1.386294
1983    3.218876
1984    2.197225
Name: log_temp, dtype: float64
```

0.5.2 Problem 2b

Use the matrix calculation for the pseudo-inverse provided in lecture.

Perform Matrix Calculation to Get `W` Weights for Model

```
In [14]: W = np.dot(np.linalg.inv(np.matmul(X.T, X)), np.matmul(X.T, y.values.reshape(-1)))
         print('Model coefficients')
         pd.Series(W, index=signif_X)
```

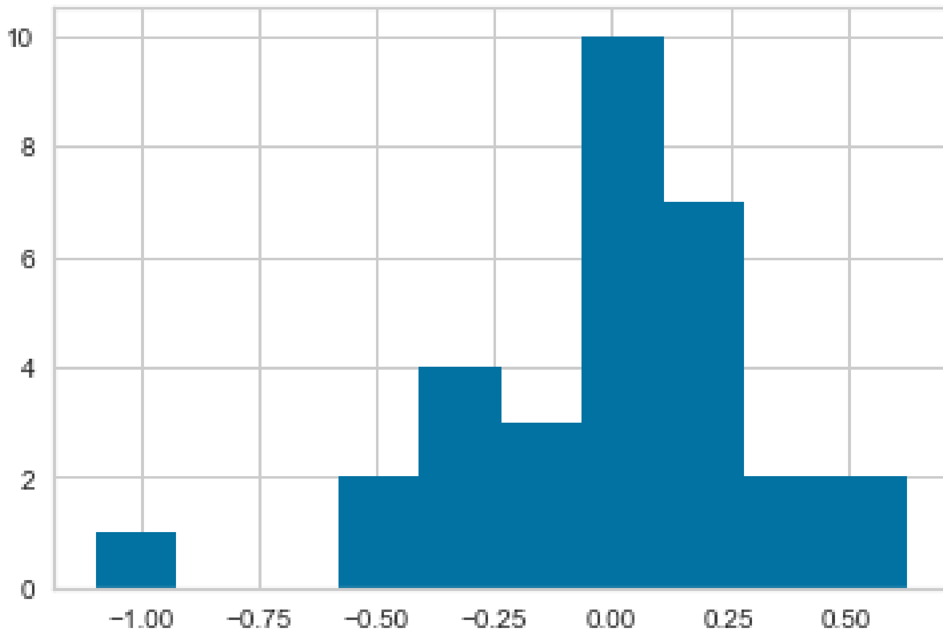
```
Model coefficients
```

```
Out[14]: const                    3.338310
         CO2                    248.225436
         Transmission          2341.300786
         CO2 Solar             -246.835955
         Solar Transmission   -2334.224694
         Solar                  34.900826
         dtype: float64
```

Observe Errors

```
In [15]: y_hat = np.dot(X, W)
         e = y - y_hat
         e.hist()
```

```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1c26f938d0>
```



Observe `r2_score`

```
In [16]: r2_score(y, y_hat)
```

```
Out[16]: 0.792261177152638
```

0.6 Problem 2c

How does the answer in Problem 2b compare to that of 1b?

Comparison of Different Approaches to Model The answers are very similar between the models. We can observe that the coefficients are different by a very, very small amount (e.g. the first 3 decimal places are the same for each coefficient, and off by very little beyond that). The R^2 value is equal between the two models.

Model coefficients from `statsmodels` (1b) approach:

```
const              3.338310
CO2              248.225444
Transmission    2341.300603
```

```
CO2 Solar              -246.835963
Solar Transmission    -2334.224511
Solar                   34.900824
```

Model coefficients from Matrix (2b) form:

```
const                    3.338310
CO2                    248.225436
Transmission          2341.300786
CO2 Solar             -246.835955
Solar Transmission   -2334.224694
Solar                   34.900826
```

0.7 Problem 3 Time Series Modeling (40 points)

Use the data timeSeries4.csv for this problem. The data are monthly reports of production.

0.7.1 Problem 3a

Plot the data and perform an exploratory analysis on the raw time series file. Comment on any trends, outliers, seasonality, whether it's stationary, etc.

Read in Time Series Data

```
In [17]: ts_df = pd.read_csv('data/timeSeries4.csv', header=None)
         ts_df.set_index(0, inplace=True)
         ts_df.head()
```

```
Out[17]:            1
         0
         0  21.684748
         1  21.622112
         2  19.583297
         3  23.290602
         4  21.729621
```

Describe Dataset & Check Missing Values

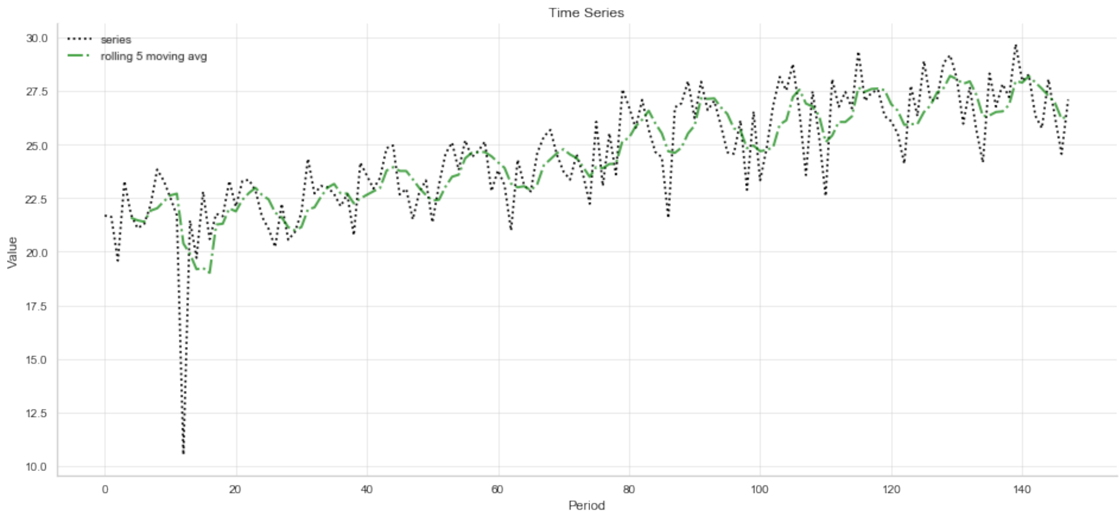
```
In [18]: print('There are %i missing values in the dataset' %ts_df.isnull().sum())

         ts_df.describe().T
```

```
There are 0 missing values in the dataset
```

```
Out[18]:    count       mean       std        min        25%        50%        75%  \
         1  148.0  24.529022  2.648555  10.521345  22.811038  24.536239  26.703908

                max
         1  29.68814
```

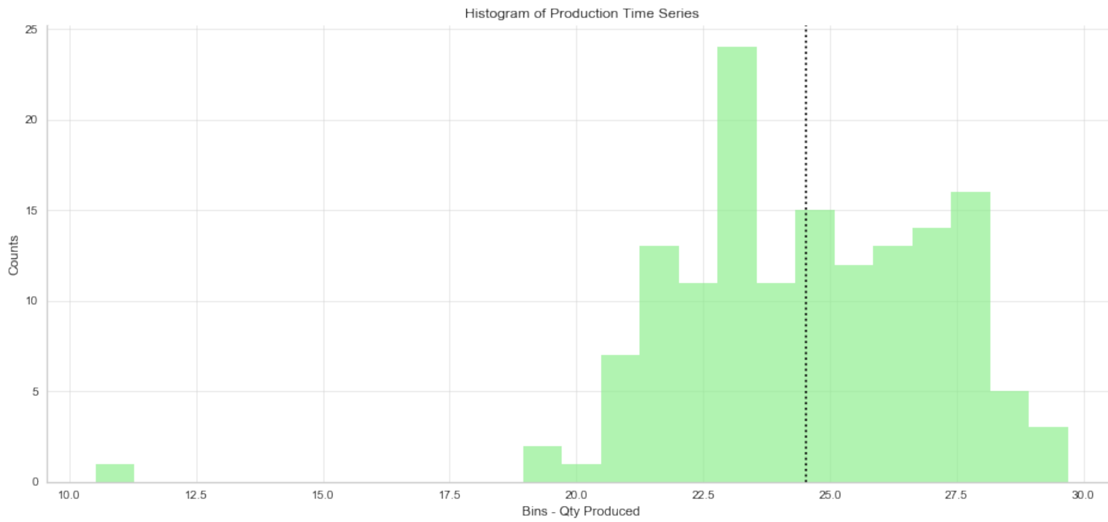
Plot TIme Series with 5 Period Moving Average

```
In [19]: fig, ax = plt.subplots(figsize=(16, 7))
         ts_df[1].plot(ax=ax, linestyle=':', color='black', label='series')
         set_mpl_preferences(ax)
         ax.set_title('Time Series')
         ax.set_xlabel('Period')
         ax.set_ylabel('Value')
         ax.plot(ts_df[1].rolling(5).mean(), linestyle='-.', color='green', alpha=.7, linewidtl
         ax.legend(loc='best')
         plt.show()
```



Plot Histogram

```
In [20]: fig, ax = plt.subplots(figsize=(16, 7))
         ts_df.hist(ax=ax, alpha=.7, color='lightgreen', bins=25)
         set_mpl_preferences(ax)
         ax.set_title('Histogram of Production Time Series')
         ax.set_xlabel('Bins - Qty Produced')
         ax.set_ylabel('Counts')
         ax.axvline(ts_df[1].mean(), linestyle=':', color='black', label='mean')
         plt.show()
```

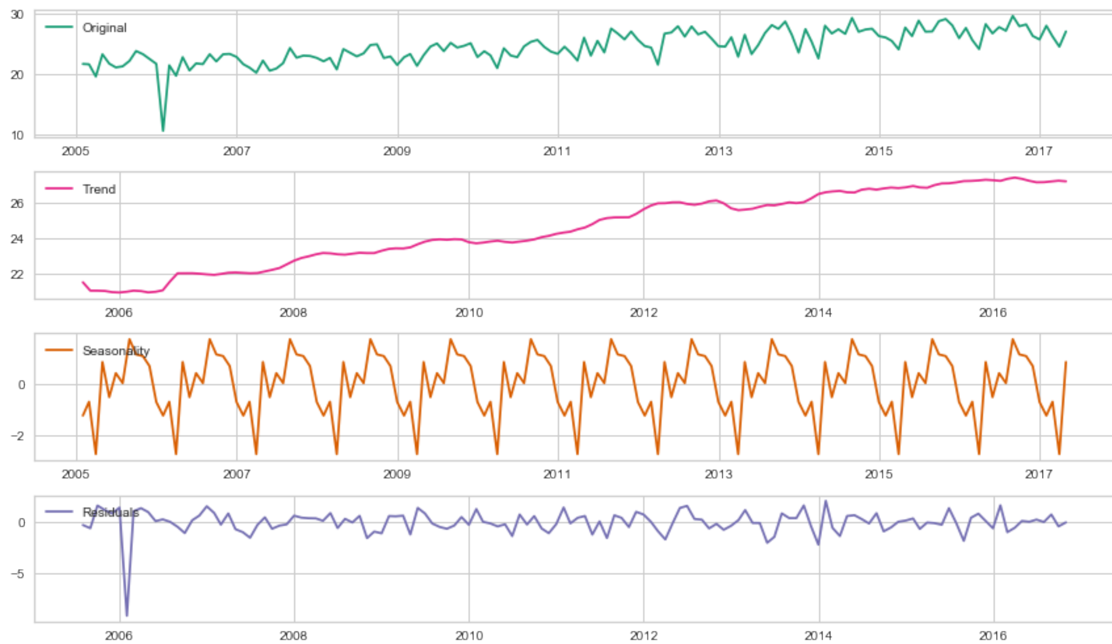


Add months to index to enable plotting functions

```
In [21]: ts_df.index = pd.date_range(start='01/01/2005', periods=ts_df.shape[0], freq='M')
```

Plot breakout Plots seasonal_decompose

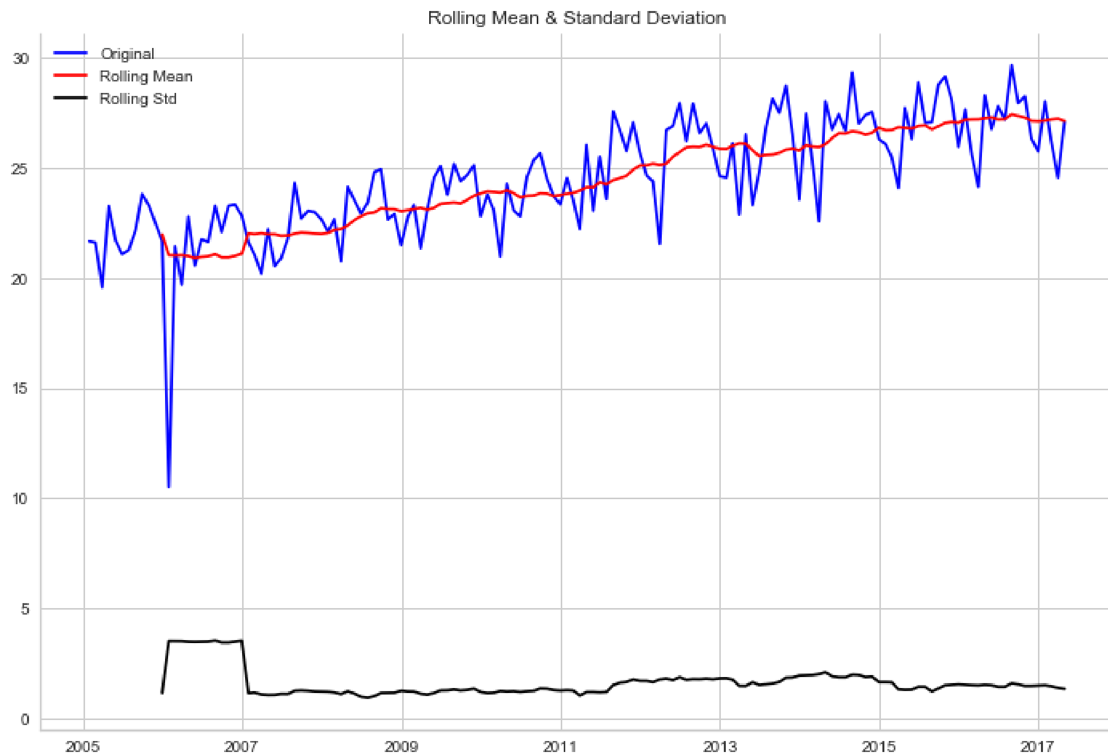
```
In [22]: def breakout_plots(seas_series):
             decomposition = seasonal_decompose(seas_series)

             f, ax = plt.subplots(1,4,figsize=(12, 7))

             plt.subplot(411)
             plt.plot(seas_series, label='Original', c=dark2_colors[0])
             plt.legend(loc='upper left')
             plt.subplot(412)
             plt.plot(decomposition.trend, label='Trend', c=dark2_colors[1])
             plt.legend(loc='upper left')
             plt.subplot(413)
             plt.plot(decomposition.seasonal,label='Seasonality', c=dark2_colors[2])
             plt.legend(loc='upper left')
             plt.subplot(414)
             plt.plot(decomposition.resid, label='Residuals', c=dark2_colors[3])
             plt.legend(loc='upper left')
             plt.tight_layout()

             return decomposition

         decomposition = breakout_plots(ts_df[1])
```



Test Stationarity with Dickey-Fuller Test

```
In [23]: def test_stationarity(timeseries):

             #Determing rolling statistics
             rolmean = timeseries.rolling(12).mean()#pd.rolling_mean(timeseries, window=12)
             rolstd = timeseries.rolling(12).std()#pd.rolling_std(timeseries, window=12)

             #Plot rolling statistics:
             fig = plt.figure(figsize=(12, 8))
             orig = plt.plot(timeseries, color='blue',label='Original')
             mean = plt.plot(rolmean, color='red', label='Rolling Mean')
             std = plt.plot(rolstd, color='black', label = 'Rolling Std')
             plt.legend(loc='best')
             plt.title('Rolling Mean & Standard Deviation')
             sns.despine()
             plt.show()

             #Perform Dickey-Fuller test:
             print('Results of Dickey-Fuller Test:')
             dftest = adfuller(timeseries, autolag='AIC')
             dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used',
             for key,value in dftest[4].items():
                 dfoutput['Critical Value (%s)'%key] = value
             print(dfoutput)
```

```
test_stationarity(ts_df[1])
```




```
Results of Dickey-Fuller Test:
Test Statistic                  -1.593535
p-value                          0.486840
#Lags Used                      13.000000
Number of Observations Used    134.000000
Critical Value (1%)             -3.480119
Critical Value (5%)             -2.883362
Critical Value (10%)            -2.578407
dtype: float64
```

Note On Stationarity The data is not stationary and there is a clear upward trend. This is evident by simply looking at the data -- but is also evident via the Dickey-Fuller test. There also appears to be very consistent seasonality, as evidenced by the `breakout` plot. There does appear to be one outlier

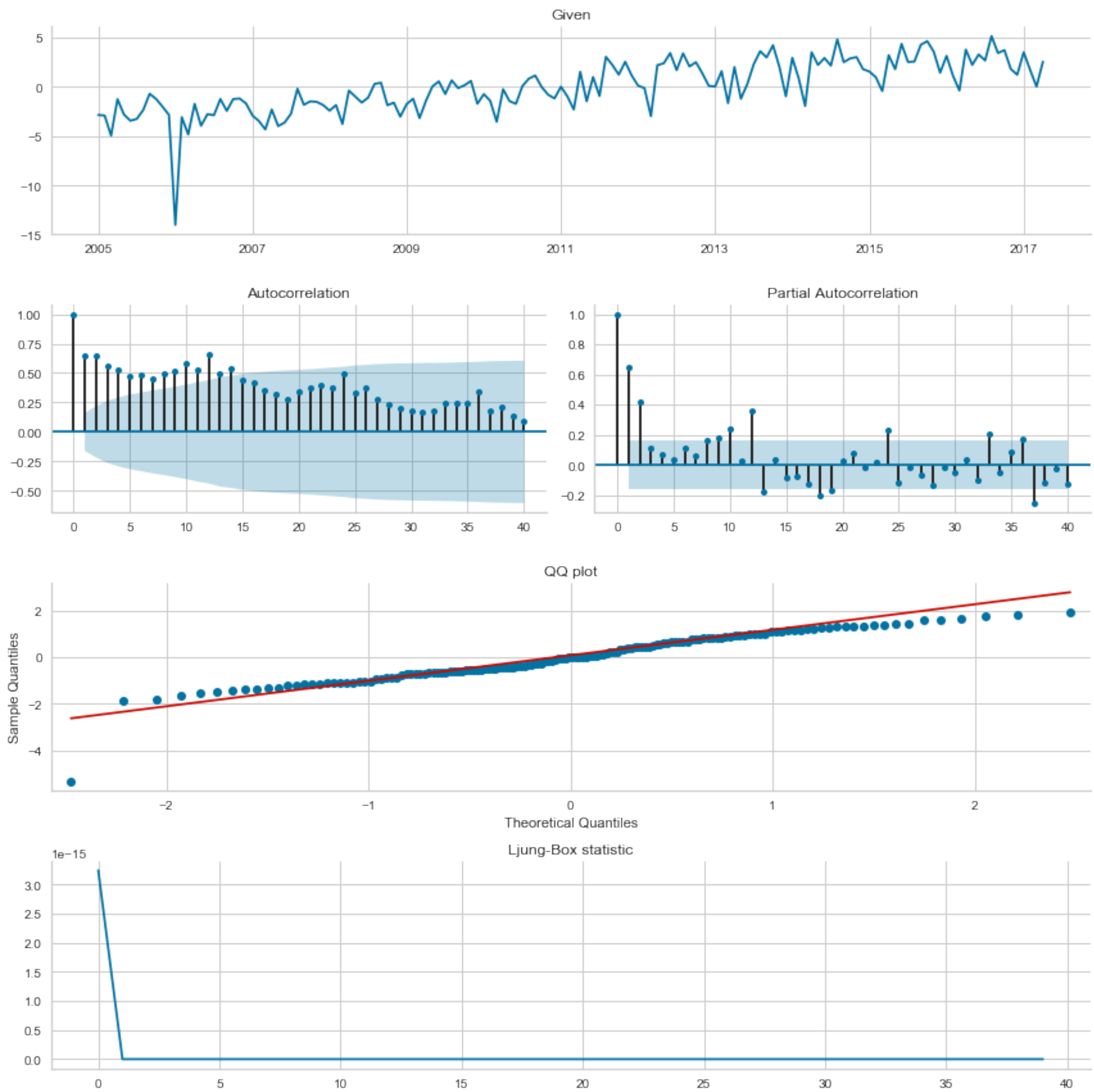
0.7.2 Problem 3b

Using your knowledge of ACF, PACF and other diagnostics, walk us through the selection of an appropriate time series model for the data. We are interested in both the result and your logical journey to reach that model. That journey should begin with observations from the ACF and PACF pattern.

Fit Baseline Model Without Differencing

```
In [24]: # arima model
         mod = sm.tsa.statespace.SARIMAX(ts_df[1],
                                         trend='c',
                                         freq='M',
                                         order=(0,0,0))

         results_ARIMA = mod.fit(disp=False)
         print(results_ARIMA.summary())

         print('\n\nResiduals\n\n')
         _ = tsplot(results_ARIMA.resid, 40)
```

```
                           Statespace Model Results
==============================================================================
Dep. Variable:                      1   No. Observations:                  148
Model:                        SARIMAX   Log Likelihood                -353.655
Date:                Mon, 22 Oct 2018   AIC                            711.311
Time:                        19:53:30   BIC                            717.305
Sample:                    01-31-2005   HQIC                           713.746
                         - 04-30-2017
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     24.5290      0.233    105.145      0.000      24.072      24.986
sigma2         6.9674      0.515     13.521      0.000       5.957       7.977
===================================================================================
Ljung-Box (Q):                     1085.33   Jarque-Bera (JB):               103.96
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               0.75   Skew:                            -0.88
Prob(H) (two-sided):                  0.33   Kurtosis:                         6.71
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).


Residuals
```



The partial autocorrelation plot (on the raw data) suggests that we need a seasonality adjustment at 12 months. It is unclear what other parameters for `(p, d, q)` should be selected, so a grid-search is performed below.

```
(p, d, q):
* p: auto-regressive part (warm today if warm past 3 days)
* d: integrated part (amount of differencing)
* q: moving avg part
```

Create Hyperparameter Set

```
In [25]: # Define the p, d and q parameters to take any value between 0 and 2
         p = d = q = range(0, 2)
```

```
# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

Perform Grid Search Over Hyperparameter Set Created

In [26]:
```
# method borrowed from
# https://www.digitalocean.com/community/tutorials/a-guide-to-time-series-forecasting
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(ts_df[1],
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, round(results
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:1361.846
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:1111.236
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:525.466
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:433.511
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:563.298
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:485.68
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:462.147
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:434.232
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:1185.662
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:950.436
```

```
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:518.242
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:417.295
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:562.617
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:486.091
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:447.616
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:416.751
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:650.324
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:525.017
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:580.803
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:443.182
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:553.53
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:507.696
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:473.333
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:433.915
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:584.28
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:464.424
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:529.567
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:389.202
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:519.451
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:455.847
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:424.222
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:398.091
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:655.641
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:563.479
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:521.914
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:420.934
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:516.561
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:486.056
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:419.344
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:421.029
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:582.836
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:464.695
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:519.292
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:391.804
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:518.18
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:456.132
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:419.035
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:400.323
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:606.882
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:495.251
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:563.116
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:430.266
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:487.664
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:489.025
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:437.785
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:424.194
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:584.391
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:462.22
```

```
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:527.379
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:388.979
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:456.483
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:453.534
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:404.401
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:396.623
```

It appears that the model with the parameters `ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:389.202` performed the best in our grid search, so this one was tested. It appeared to fit the non-outlier points well, yet the Ljung-Box statistic did not maintain a value above the 0.1 threshold.

Fit Model with Best Hyperparameters

```
In [27]: # arima model
         mod = sm.tsa.statespace.SARIMAX(ts_df[1],
                                         trend='c',
                                         freq='M',
                                         order=(0,1,1),
                                         seasonal_order=(0, 0, 0, 12),#(0,1,1,12)
                                         enforce_stationarity=False,
                                         enforce_invertibility=False)

         results_ARIMA = mod.fit(disp=False)
```

0.7.3 Problem 3c

Apply and show the appropriate diagnostics to the model to assert that it is valid. Include not just a plot but your interpretation of the plot in your justification.

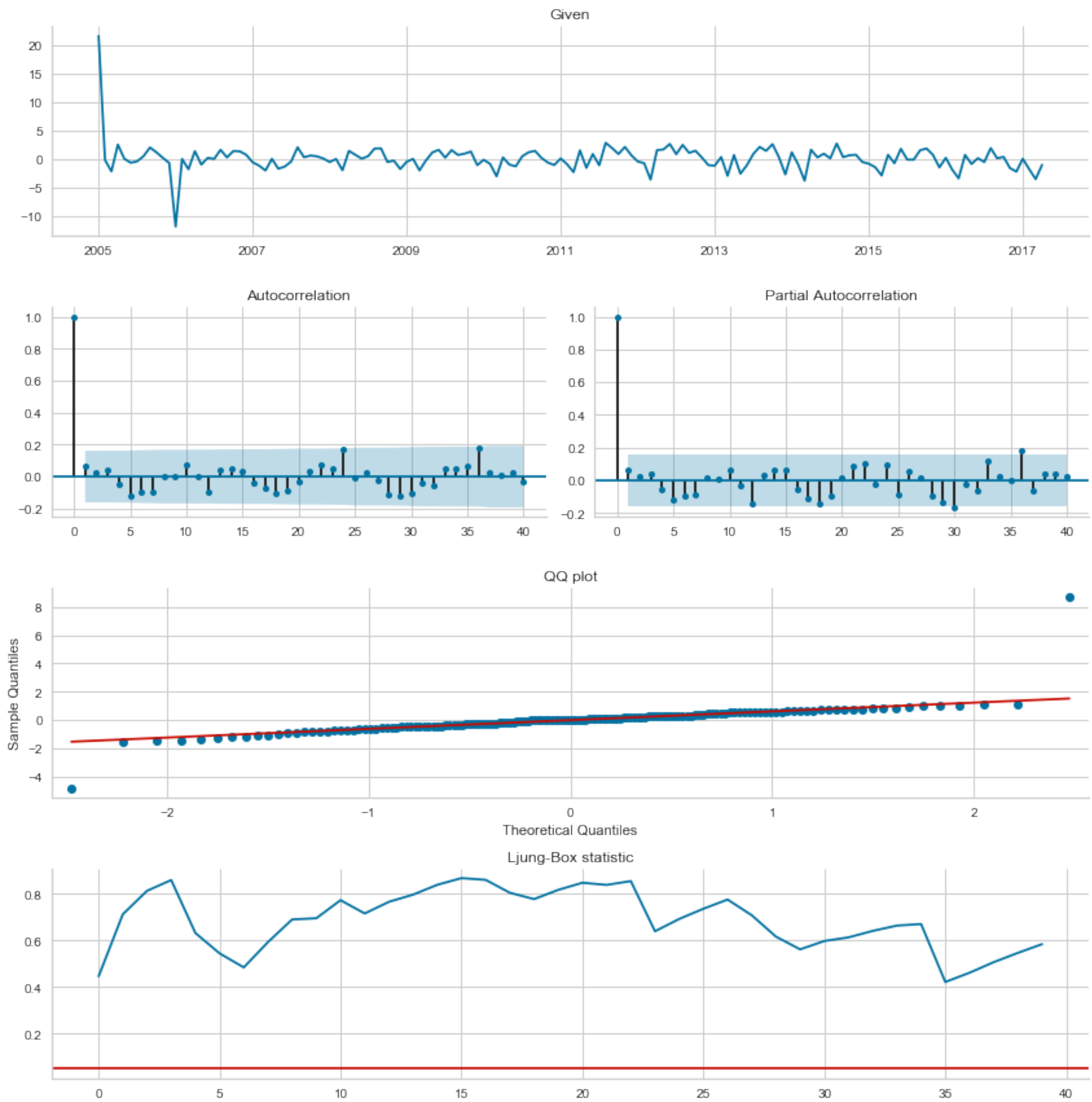
Get Model Summary & Residual Plot

```
In [28]: print(results_ARIMA.summary())

         print('\n\nResiduals\n\n')
         _ = tsplot(results_ARIMA.resid, 40)
```

```
                           Statespace Model Results
==============================================================================
Dep. Variable:                      1   No. Observations:                  148
Model:               SARIMAX(0, 1, 1)   Log Likelihood                -284.820
Date:                Mon, 22 Oct 2018   AIC                            575.640
Time:                        19:53:37   BIC                            584.570
Sample:                    01-31-2005   HQIC                           579.269
                         - 04-30-2017
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
```

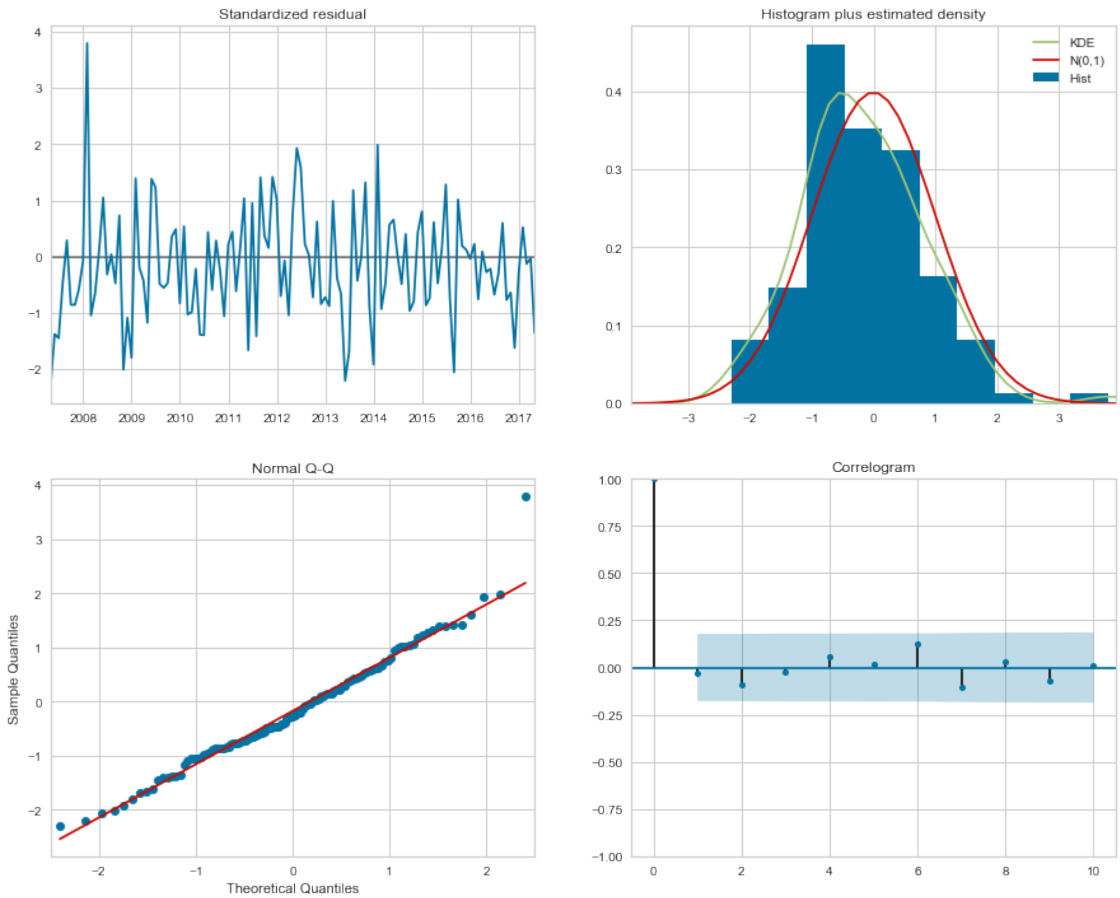
```
------------------------------------------------------------------------------
intercept      0.0477      0.007      7.146      0.000      0.035      0.061
ma.L1         -1.0000   1593.794     -0.001      0.999  -3124.779   3122.779
sigma2         2.8893   4605.036      0.001      0.999  -9022.816   9028.595
===================================================================================
Ljung-Box (Q):                      188.50   Jarque-Bera (JB):               964.05
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               0.67   Skew:                            -2.19
Prob(H) (two-sided):                  0.17   Kurtosis:                        14.85
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

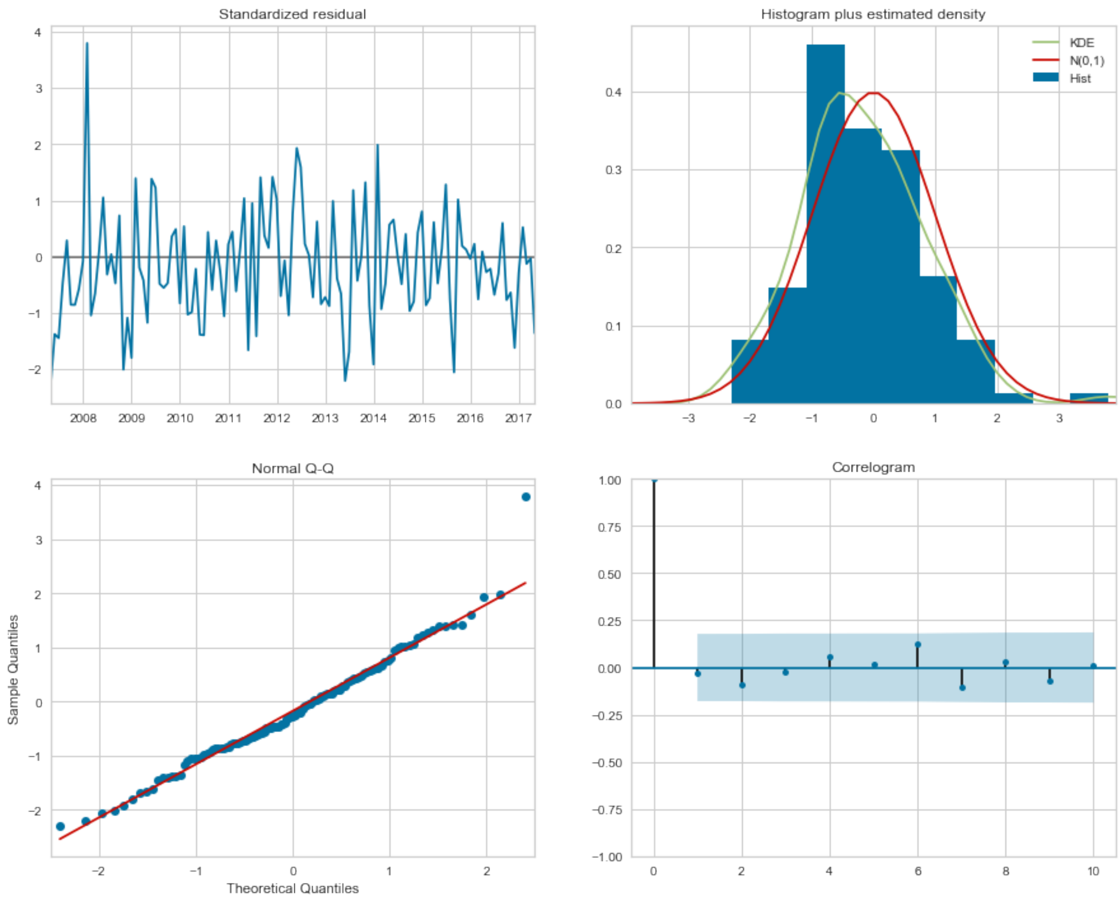
Residuals



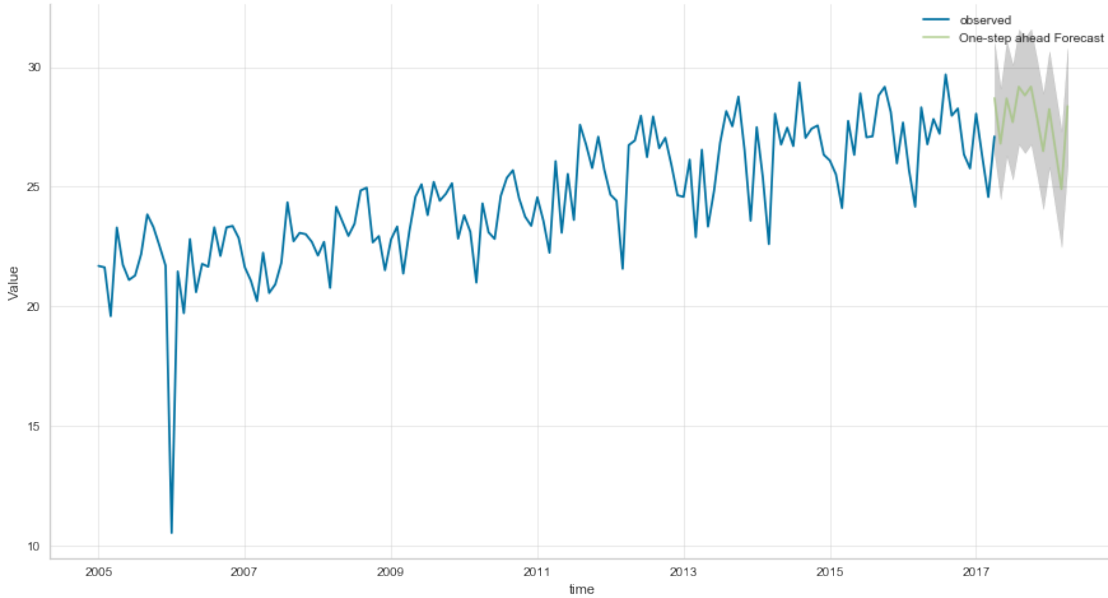

```
In [29]: results.plot_diagnostics(figsize=(15, 12))
```

Out[29]:


Standardized residual
Histogram plus estimated density
KDE
N(0,1)
Hist
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Correlogram




```
In [30]: pred = results.get_prediction(start=pd.to_datetime('2017-04-30'),
                                       end=pd.to_datetime('2018-04-30'), dynamic=False)
         pred_ci = pred.conf_int()

         fig, ax = plt.subplots(figsize=(15, 8))
         ts_df[1].plot(label='observed', ax=ax)
         pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7)
         set_mpl_preferences(ax)

         ax.fill_between(pred_ci.index,
                         pred_ci.iloc[:, 0],
                         pred_ci.iloc[:, 1], color='k', alpha=.2)

         ax.set_xlabel('time')
         ax.set_ylabel('Value')
         plt.legend()

         plt.show()
```



Out of the models that were fit via brute force that had the lowest AIC values, the model that was chosen was the one fit with the parameters `order=(0, 1, 1)` and `seasonal_order=(0, 0, 0, 12)`. This model had ACF and PACF plots that maintained non-significant status, a Ljung-Box statistic that maintained levels above 0.1, and the standardized residuals appear to be random noise.

0.8 Problem 4 (15 points)

For a time series data set, a (2,1,1) was derived with the following coefficients:

const -0.3916
ar1 0.9172
ar2 -0.2390
ma1 0.4012

The last 5 points are -104.6, -102.1, -103.2, -109.8, -115.7

Compute the next 3 data points by writing the calculation in python. Note that this will require not only plugging values into the equation, but also taking the d term of the (p,d,q) ARIMA model into account. We do not need a general form or function--just the required calculations.

```
In [31]: X = pd.Series([-104.6, -102.1, -103.2, -109.8, -115.7])
         X

Out[31]: 0   -104.6
         1   -102.1
         2   -103.2
         3   -109.8
         4   -115.7
         dtype: float64

In [32]: X_diff = X.diff()
         X_diff
```

```
Out[32]: 0    NaN
         1    2.5
         2   -1.1
         3   -6.6
         4   -5.9
         dtype: float64
```

```
In [33]: coefs = dict(const=-0.3916,
                      ar1=0.9172,
                      ar2=-0.2390,
                      ma1=0.4012)

         params = dict(p=2, d=1, q=1)
```

```
In [34]: yhat_5 = X.mean() + coefs['const'] + coefs['ar1']*X_diff[4] + coefs['ar2']*X_diff[3] +
         yhat_5
```

```
Out[34]: -113.67276000000001
```

```
In [35]: X[5] = yhat_5
         X
```

```
Out[35]: 0   -104.60000
         1   -102.10000
         2   -103.20000
         3   -109.80000
         4   -115.70000
         5   -113.67276
         dtype: float64
```

```
In [36]: X_diff = X.diff()
         X_diff
```

```
Out[36]: 0         NaN
         1    2.50000
         2   -1.10000
         3   -6.60000
         4   -5.90000
         5    2.02724
         dtype: float64
```

```
In [37]: yhat_6 = X.mean() + coefs['const'] + coefs['ar1']*X_diff[5] + coefs['ar2']*X_diff[4] +
         yhat_6
```

```
Out[37]: -104.48758011733334
```

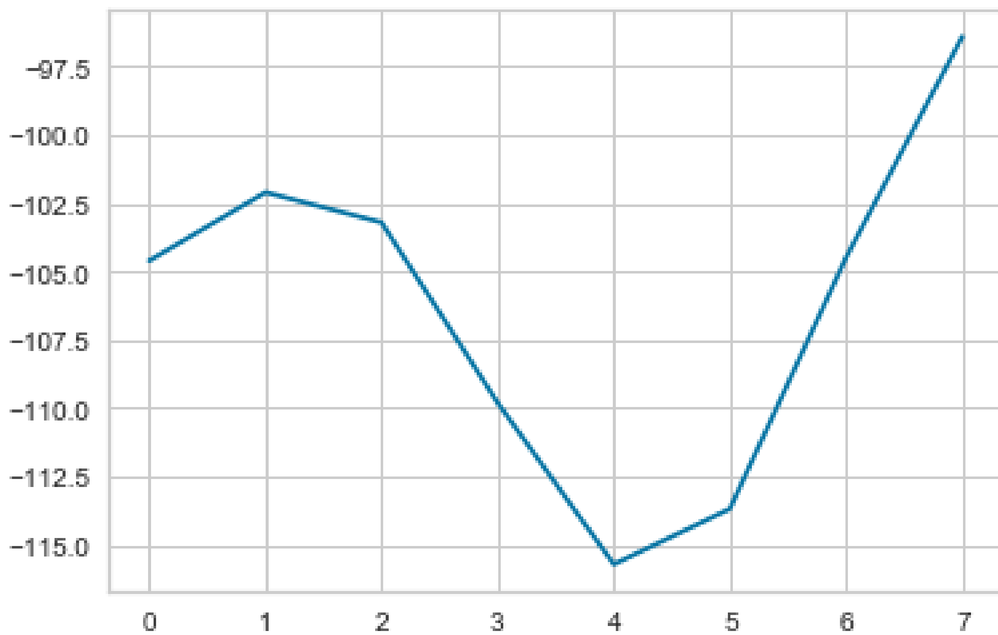
```
In [38]: X[6] = yhat_6
         X_diff = X.diff()
```

```
In [39]: yhat_7 = X.mean() + coefs['const'] + coefs['ar1']*X_diff[6] + coefs['ar2']*X_diff[5] +
         X[7] = yhat_7
         X
```

```
Out[39]: 0   -104.600000
         1   -102.100000
         2   -103.200000
         3   -109.800000
         4   -115.700000
         5   -113.672760
         6   -104.487580
         7    -96.417846
         dtype: float64
```

```
In [40]: X.plot()
```

```
Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x1198cfcc0>
```



0.9 Problem 5 (2 points)

How many hours did this homework take you? The answer to this question will not affect your grade.

About 14.

0.10 Last step (5 points)

Save this notebook as LastnameFirstnameHW3.ipynb such as BradyTom.ipynb. Create a pdf of this notebook named similarly. Submit both the python notebook and the pdf version to the Canvas dropbox. We require both versions.