**Blog**

**Time Series: Converting Transformed Data to Original Scale**

So, you just worked through several Stationarity Assumptions to properly

transform your original dataset in preparation for time series modeling.

To make a time series stationary, you may have performed the following:

**To Remove Trend:**

If your data has a strong trend (positive or negative)

You can consider using the following to reduce the trend. These

transformations may be: **log, square root, cube root**, and/or **smoothing using a rolling average/moving average or exponentially weighted**

**moving average,** etc.

**To Remove Seasonality and Trend:**

If your data shows both a strong trend and seasonality, then you may

consider using **differencing,** etc.

**Note:** With some data you may need to use a combination of these

transformations. In order to satisfy the Stationarity assumptions.

Once you have performed the transformation(s) and re-test using a statisticaltest like **Dickey-Fuller test** and confirm that your p-value is less than .05, then you are ready to continue modeling the transformed data.

However, after the modeling process and it is time to forecast your data you will want to perform the forecasting on the original scale. So, we need to

**un-transform** your data.

So, the following examples show you how to work backwards from **log,**

**differencing and/or rolling average** to get to the original scale.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**EXAMPLE** |The monthly median home price in a specific zip code (2008-2018).

**Original Dataset**

ts\_77478.head()

time

2008-04-01 218900.0

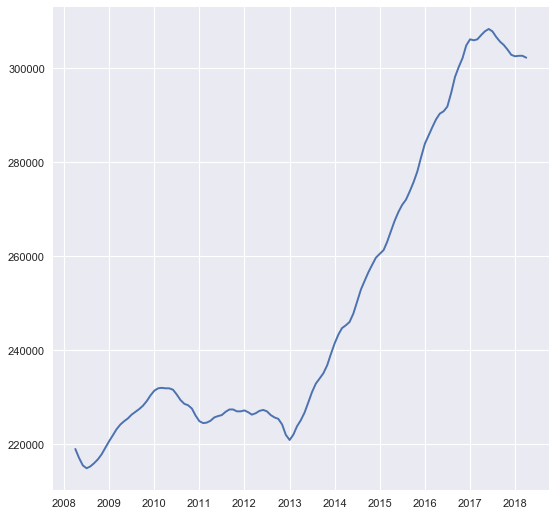
2008-05-01 217000.0

2008-06-01 215400.0

2008-07-01 214800.0

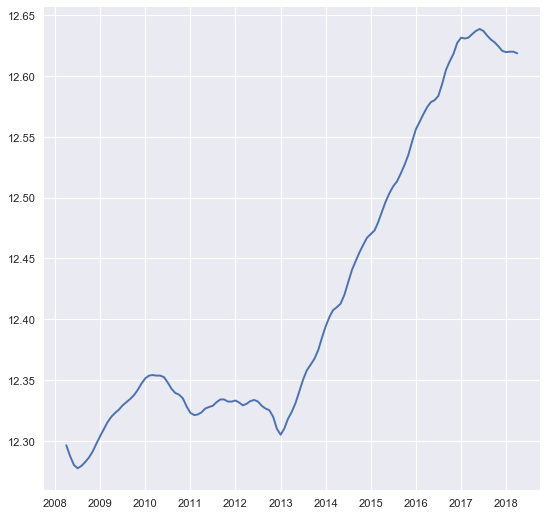
2008-08-01 215200.0

plt.plot(ts\_77478)



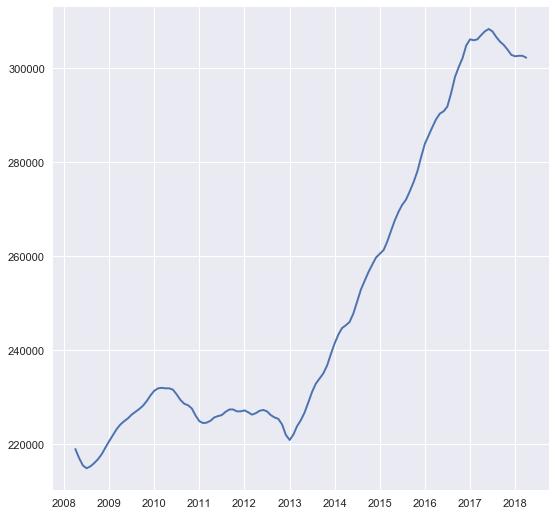
**Convert Log back to Original Scale**

plt.plot(np.log(ts\_77478))



LogDataexp = np.exp(np.log(ts\_77478))

plt.plot(LogDataexp)



LogDataexp.head(24)

time

2008-04-01 218900.0

2008-05-01 217000.0

2008-06-01 215400.0

2008-07-01 214800.0

2008-08-01 215200.0

2008-09-01 215900.0

2008-10-01 216700.0

2008-11-01 217800.0

2008-12-01 219200.0

2009-01-01 220600.0

2009-02-01 221900.0

2009-03-01 223100.0

2009-04-01 224100.0

2009-05-01 224800.0

2009-06-01 225400.0

2009-07-01 226200.0

2009-08-01 226800.0

2009-09-01 227400.0

2009-10-01 228100.0

2009-11-01 229100.0

2009-12-01 230300.0

2010-01-01 231300.0

2010-02-01 231800.0

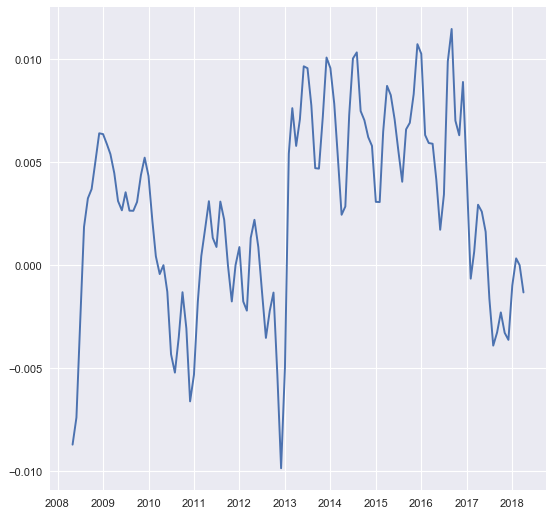
2010-03-01 231900.0

**Convert Log + Diff(1) back to Original Scale**

logData = np.log(ts\_77478)

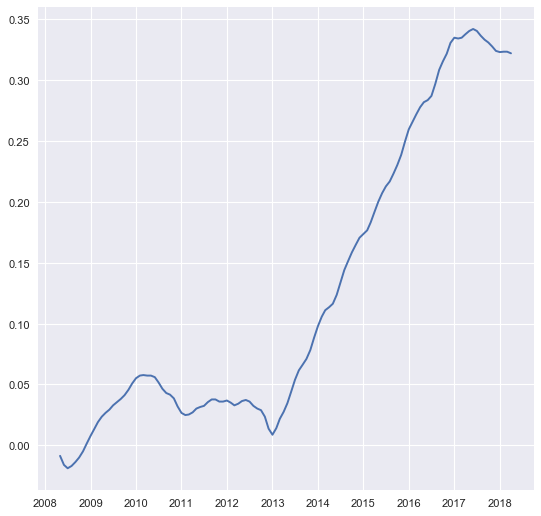
logDiff = logData.diff(1)

plt.plot(logDiff)



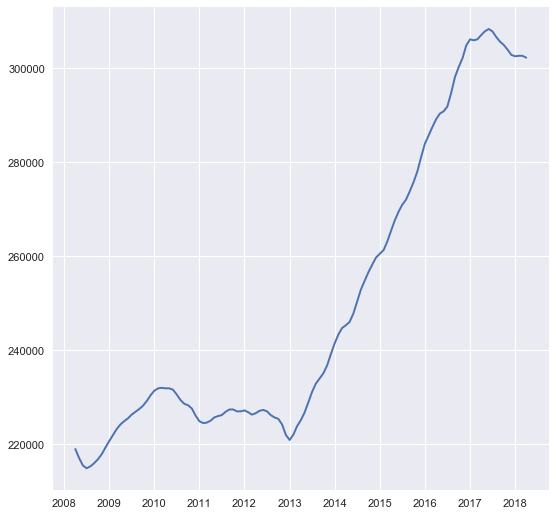
logDiffcumsum = logDiff.cumsum()

plt.plot(logDiffcumsum)



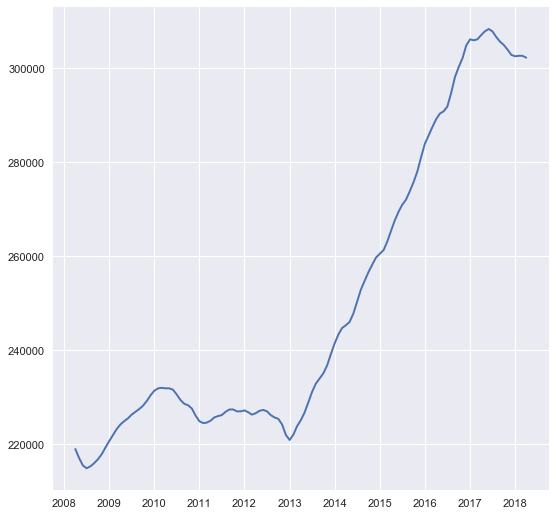
logAftercumsum = np.exp(logData).add(logDiffcumsum, fill\_value=0)

plt.plot(logAftercumsum)



OriginalData = logAftercumsum

plt.plot(OriginalData )



OriginalData.head(12)

time

2008-04-01 218900.000000

2008-05-01 216999.991282

2008-06-01 215399.983882

2008-07-01 214799.981092

2008-08-01 215199.982953

2008-09-01 215899.986200

2008-10-01 216699.989899

2008-11-01 217799.994962

2008-12-01 219200.001370

2009-01-01 220600.007736

2009-02-01 221900.013612

2009-03-01 223100.019005

2009-04-01 224100.023477

**Convert Subtract Rolling Mean + Log + Diff (1) back to Original Scale**

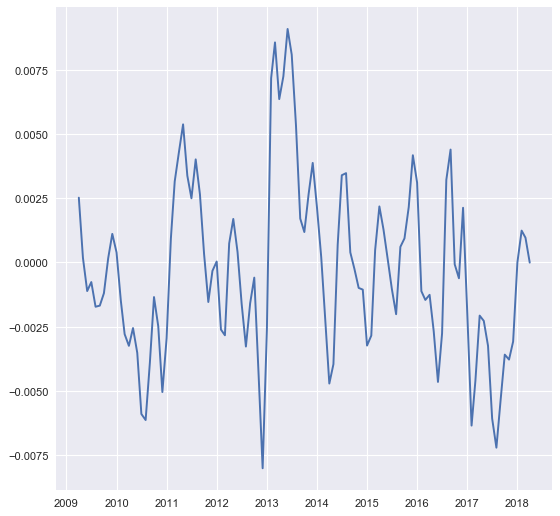
log\_data = np.log(ts\_77478)

rolmean = log\_data.rolling(window = 12).mean()

log\_data\_minus\_rolmean = log\_data - rolmean

diff\_log\_data\_minus\_rolmean = log\_data\_minus\_rolmean.diff(periods=1)

plt.plot(diff\_log\_data\_minus\_rolmean)



diff\_log\_data\_minus\_rolmean.head(24)

time

2008-04-01 NaN

2008-05-01 NaN

2008-06-01 NaN

2008-07-01 NaN

2008-08-01 NaN

2008-09-01 NaN

2008-10-01 NaN

2008-11-01 NaN

2008-12-01 NaN

2009-01-01 NaN

2009-02-01 NaN

2009-03-01 NaN

2009-04-01 0.002516

2009-05-01 0.000176

2009-06-01 -0.001116

2009-07-01 -0.000766

2009-08-01 -0.001726

2009-09-01 -0.001683

2009-10-01 -0.001199

2009-11-01 0.000159

2009-12-01 0.001108

2010-01-01 0.000386

2010-02-01 -0.001478

2010-03-01 -0.002793

rolmeancumsum = diff\_log\_data\_minus\_rolmean.cumsum()

rolmeanlogAftercumsum = np.exp(logData).add(rolmeancumsum, fill\_value=0)

rolmeanlogAftercumsum.head(12)

time

2008-04-01 218900.000000

2008-05-01 217000.000000

2008-06-01 215400.000000

2008-07-01 214800.000000

2008-08-01 215200.000000

2008-09-01 215900.000000

2008-10-01 216700.000000

2008-11-01 217800.000000

2008-12-01 219200.000000

2009-01-01 220600.000000

2009-02-01 221900.000000

2009-03-01 223100.000000

2009-04-01 224100.002516

plt.plot(rolmeanlogAftercumsum)

