Part1_Time_Series_Data_BasicPlotting

August 3, 2016

Part 1: Getting Time Series Data and Plotting

This code demonstrates how to view time series data with pandas and various methods of sampling, smoothing (rolling mean), and applying linear regression to the data.

if this tutorial doesn't cover what you are looking for, please leave a comment on the youtube video and I will try to cover what you are interested in.

https://www.youtube.com/watch?v=OwnaUVt6VVE

Importing Libraries

```
In [161]: import pandas as pd
          import pandas_datareader.data as web
          import numpy as np
          import matplotlib.pyplot as plt
          from sklearn.linear_model import LinearRegression
          %matplotlib inline
```

Getting Data and Viewing with Pandas

```
In [162]: # https://pandas-datareader.readthedocs.io/en/latest/remote_data.html
          google = web.DataReader('GOOG', data_source = 'google', start = '3/14/2009', end = '4/14/2016
          google = google.drop('Volume', axis = 1 )
          google.head()
```

Out[162]:

Date	Open	High	Low	Close
2009-03-16	162.83	164.70	159.14	159.69
2009-03-17	159.93	167.50	159.39	167.50
2009-03-18	167.24	169.83	163.86	166.38
2009-03-19	165.67	167.83	163.53	164.81
2009-03-20	164.98	166.33	163.01	164.91

Adding Column

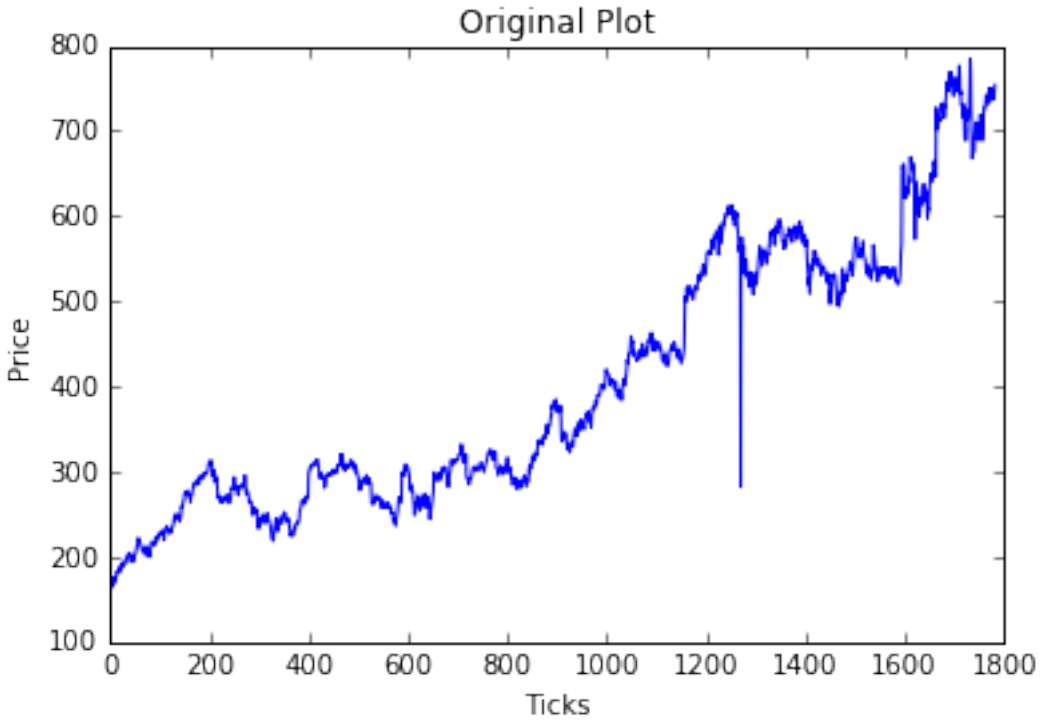
```
In [137]: google['Ticks'] = range(0,len(google.index.values))

In [138]: google.head()
```

Out[138]:

Date	Open	High	Low	Close	Ticks
2009-03-16	162.83	164.70	159.14	159.69	0
2009-03-17	159.93	167.50	159.39	167.50	1
2009-03-18	167.24	169.83	163.86	166.38	2
2009-03-19	165.67	167.83	163.53	164.81	3
2009-03-20	164.98	166.33	163.01	164.91	4

Plotting Ticks vs Open Price

```
In [139]: #very simple plotting
          fig = plt.figure(1)
          ax1 = fig.add_subplot(111)
          ax1.set_xlabel('Ticks')
          ax1.set_ylabel('Price')
          ax1.set_title('Original Plot')
          ax1.plot('Ticks', 'Open', data = google);
```



Sampling 1/10th of the Data

```
In [140]: one_tenth = google.sample(frac = .1, random_state=np.random.randint(10))

In [141]: one_tenth.head()
```

Out[141]:

Date	Open	High	Low	Close	Ticks
2015-07-24	647.00	648.17	622.52	623.56	1600
2009-12-15	296.35	297.89	295.20	296.27	191
2015-12-16	750.00	760.59	739.44	758.09	1701
2015-03-24	562.56	574.59	561.21	570.19	1515
2013-10-23	500.00	516.86	499.81	515.19	1160

Reordering Data by Ticks Value

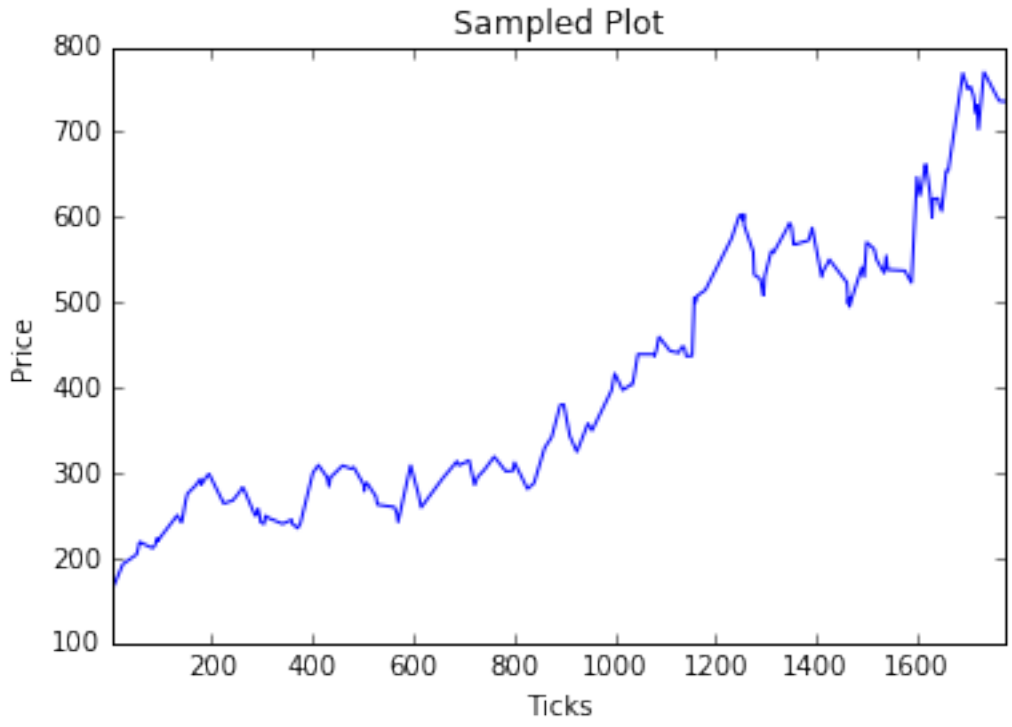
```
In [142]: # removing index name
          one_tenth.index.name = None
          one_tenth = one_tenth.sort_values(by=['Ticks'], ascending=[True])
          one_tenth.head()
```

Out[142]:

	Open	High	Low	Close	Ticks
2009-03-20	164.98	166.33	163.01	164.91	4
2009-04-20	192.88	195.13	187.76	189.46	24
2009-05-28	204.14	205.60	202.10	204.99	51
2009-06-03	212.79	216.02	211.79	215.61	55
2009-06-04	217.43	220.40	217.03	219.92	56

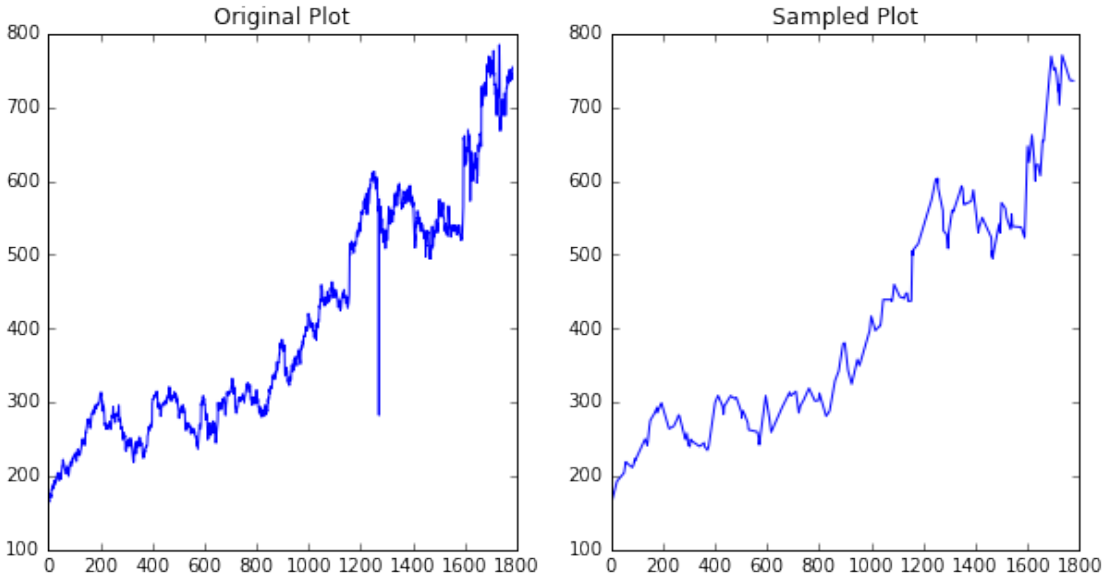
Plotting Ticks vs Open Price on Sampled Data

In [143]:
```
axes = one_tenth.plot('Ticks', 'Open', legend = False, title = 'Sampled Plot');
axes.legend = None;
axes.set_ylabel('Price');
```



Plotting Original Data vs Sampled Data (Subplot)

In [144]:
```
fig, axes = plt.subplots(nrows = 1, ncols = 2, figsize = (10,5));
axes[0].plot('Ticks', 'Open', data = google);
axes[0].set_title('Original Plot');
axes[1].plot('Ticks', 'Open', data = one_tenth);
axes[1].set_title('Sampled Plot');
```



Change the Index From Date to Ticks

```
In [145]: #google.index = google['Ticks']
          # Removing Index
          google = google.reset_index()
          google.head(3)
```

Out[145]:

	Date	Open	High	Low	Close	Ticks
0	2009-03-16	162.83	164.70	159.14	159.69	0
1	2009-03-17	159.93	167.50	159.39	167.50	1
2	2009-03-18	167.24	169.83	163.86	166.38	2

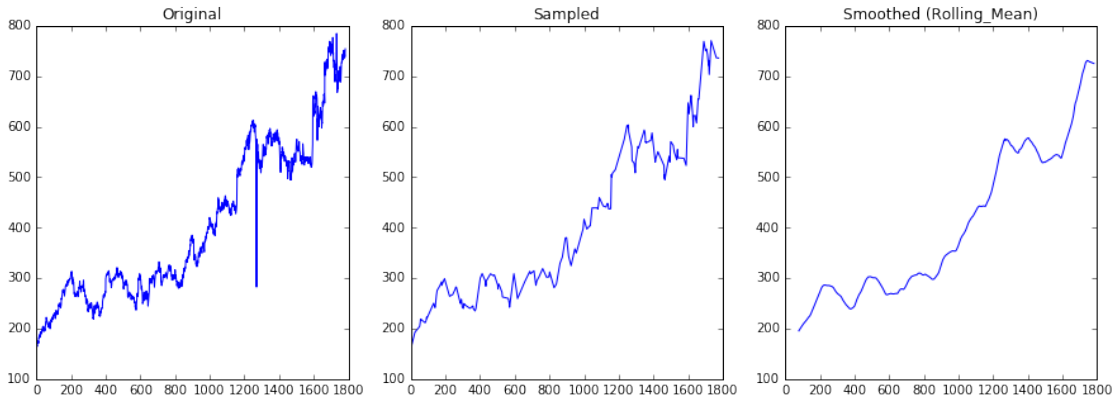
Plotting Original Data vs Sampled vs Rolling Mean Plot (Subplot)

```
In [146]: # documentation: http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.rolling.
          google['Rolling_Mean'] = google['Open'].rolling(window = 80).mean()
          google.head(5)
```

Out[146]:

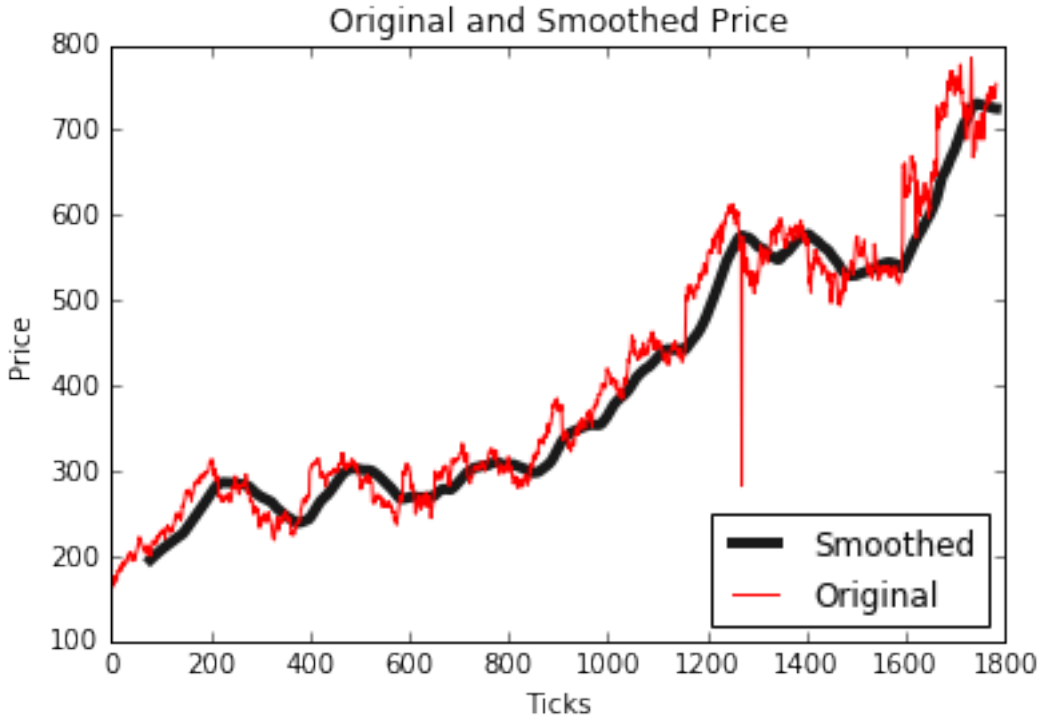
	Date	Open	High	Low	Close	Ticks	Rolling_Mean
0	2009-03-16	162.83	164.70	159.14	159.69	0	NaN
1	2009-03-17	159.93	167.50	159.39	167.50	1	NaN
2	2009-03-18	167.24	169.83	163.86	166.38	2	NaN
3	2009-03-19	165.67	167.83	163.53	164.81	3	NaN
4	2009-03-20	164.98	166.33	163.01	164.91	4	NaN

```
In [147]: fig, axes = plt.subplots(nrows = 1, ncols = 3, figsize = (15,5));
          axes[0].plot('Ticks', 'Open', data = google);
          axes[0].set_title('Original');
          axes[1].plot('Ticks', 'Open', data = one_tenth);
          axes[1].set_title('Sampled');
          axes[2].plot('Ticks', 'Rolling_Mean', data = google);
          axes[2].set_title('Smoothed (Rolling_Mean)');
```



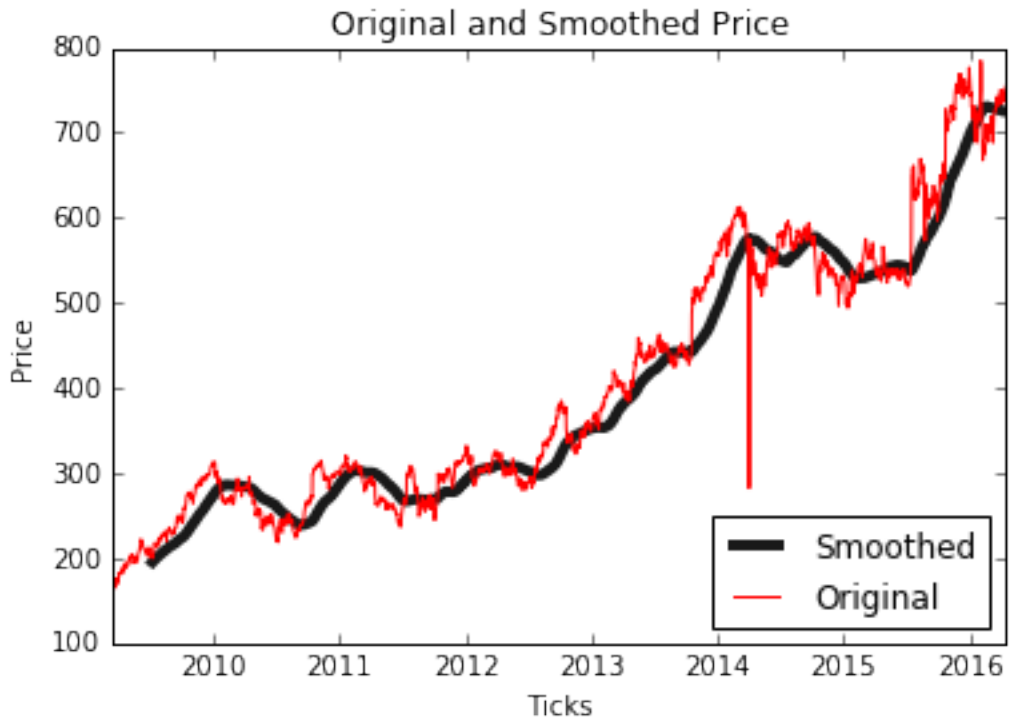
Plotting Original Data and Smoothed Data on Same Plot

```
In [148]: fig = plt.figure();
          ax = fig.add_subplot(111);
          ax.plot(google['Ticks'], google['Rolling_Mean'], color = (0,0,0), linewidth = 4, alpha = .9,
          ax.plot(google['Ticks'], google['Open'], color = (1,0,0), label = 'Original');
          ax.set_title('Original and Smoothed Price')
          ax.set_xlabel('Ticks')
          ax.set_ylabel('Price')
          ax.legend(loc='lower right');
```



Plotting Original Data and Smoothed Data on Same Plot, but with Date as x axis

In [149]:
```
# Almost exactly the code as above but with datetime
fig = plt.figure();
ax = fig.add_subplot(111);
ax.plot(google['Date'], google['Rolling_Mean'], color = (0,0,0), linewidth = 4, alpha = .9, la
ax.plot(google['Date'], google['Open'], color = (1,0,0), label = 'Original');
ax.set_title('Original and Smoothed Price')
ax.set_xlabel('Ticks')
ax.set_ylabel('Price')
ax.legend(loc='lower right');
```



In [150]:
```
# Plotting Date works best with datetime as the type.
type(google['Date'].values[0])
```

Out[150]:
```
numpy.datetime64
```

Getting Data Only from Tick 800 to 1200

In [151]:
```
filt_google = google[(google['Ticks'] >= 800) & (google['Ticks'] <= 1200)]
```

In [152]:
```
filt_google.head()
```

Out[152]:

	Date	Open	High	Low	Close	Ticks	Rolling_Mean
800	2012-05-17	316.60	318.61	310.30	311.21	800	306.568375
801	2012-05-18	312.24	315.89	298.05	299.90	801	306.865500
802	2012-05-21	299.96	307.54	299.70	306.75	802	307.043750
803	2012-05-22	306.41	306.60	297.70	300.10	803	307.310125
804	2012-05-23	300.52	304.50	298.26	304.43	804	307.457375

Linear Regression

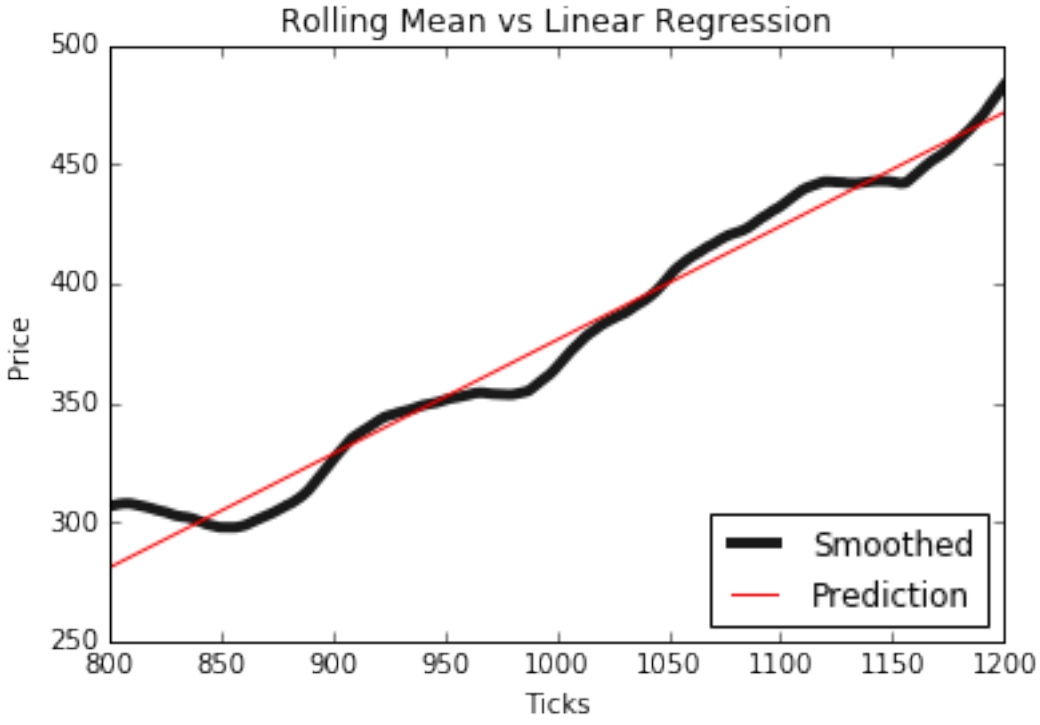
```
In [153]: from sklearn.linear_model import LinearRegression

In [154]: model = LinearRegression().fit(filt_google[['Ticks']], filt_google[['Rolling_Mean']])
          m = model.coef_[0]
          b = model.intercept_
          #equation of the line
          print 'y = ', round(m[0],2), 'x + ', round(b[0],2)

y =  0.48 x +  -100.16

In [155]: # using the equation of the line to get y values
          predictions = model.predict(filt_google[['Ticks']])
          predictions[0:5]

Out[155]: array([[ 280.99658148],
                 [ 281.47302422],
                 [ 281.94946697],
                 [ 282.42590971],
                 [ 282.90235246]])

In [156]: # making a DataFrame for the predictions
          predictions = pd.DataFrame(data = predictions, index = filt_google.index.values, columns = ['
          predictions.head()

Out[156]:            Pred
          800  280.996581
          801  281.473024
          802  281.949467
          803  282.425910
          804  282.902352
```

Joining the Two DataFrames

```
In [157]: # join and concat documentation
          # http://pandas.pydata.org/pandas-docs/stable/merging.html

          joined_df = filt_google.join(predictions, how = 'inner')
          joined_df.head()

Out[157]:           Date    Open    High     Low   Close  Ticks  Rolling_Mean  \
          800 2012-05-17  316.60  318.61  310.30  311.21    800    306.568375
          801 2012-05-18  312.24  315.89  298.05  299.90    801    306.865500
          802 2012-05-21  299.96  307.54  299.70  306.75    802    307.043750
          803 2012-05-22  306.41  306.60  297.70  300.10    803    307.310125
          804 2012-05-23  300.52  304.50  298.26  304.43    804    307.457375

                     Pred
          800  280.996581
          801  281.473024
          802  281.949467
          803  282.425910
          804  282.902352
```

Plotting the Values Based on the Equation vs the Rolling Mean

In [159]:
```
fig = plt.figure();
ax = fig.add_subplot(111);
ax.plot(joined_df['Ticks'], joined_df['Rolling_Mean'], color = (0,0,0), linewidth = 4, alpha =
ax.plot(joined_df['Ticks'], joined_df['Pred'], color = (1,0,0), label = 'Prediction');
ax.set_title('Rolling Mean vs Linear Regression')
ax.set_xlabel('Ticks')
ax.set_ylabel('Price')
ax.legend(loc='lower right');
```



In [160]:
```
# documentation
# http://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html+
import sklearn

r_squared = sklearn.metrics.r2_score(joined_df['Rolling_Mean'],joined_df['Pred'],multioutput=
r_squared
```

Out[160]: 0.97617654301753054