004-Trends_and_Cycles

November 2, 2017

```
In [1]: # author: René Kopeinig
        # script: Trends and cycles in cryptocurrency data
        # description: Applying Arima and stochastic cycle model to demonstrate
        #              the difference of trend from cycle on cryptocurrency data from Quandl
```

```
In [2]: # Add IPython-specific directive to display plots directly below the notebook cell
        %matplotlib inline
```

```
In [3]: # Import dependencies
        import os, quandl, pickle
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import statsmodels.api as sm
```

```
In [4]: # Firstly: Get data from Quandl
        # What is Quandl? It is a marketplace for financial, economic and alternative data
        # delivered in modern formats for today's analysts, including Python.

        def get_data(quandl_id):
            '''Download and cache Quandl dataseries'''
            cache_path = '{}.pkl'.format(quandl_id).replace('/','-')
            print cache_path
            try:
                f = open(cache_path, 'rb')
                df = pickle.load(f)
                print('Loaded {} from cache'.format(quandl_id))
            except (OSError, IOError) as e:
                print('Downloading {} from Quandl'.format(quandl_id))
                df = quandl.get(quandl_id, returns="pandas")
                df.to_pickle(cache_path)
                print('Cached {} at {}'.format(quandl_id, cache_path))
            return df
```

```
In [51]: # Get Bitcoin Data
         gdax_btc_eur = get_data('GDAX/EUR')
         # Starting in June 2017 until September 2017
         gdax_btc_eur = gdax_btc_eur['2017-06':'2017-09']
```

```
GDAX-EUR.pkl
Loaded GDAX/EUR from cache
```

```
In [6]: hp_cycle, hp_trend = sm.tsa.filters.hpfilter(gdax_btc_eur['Open'], lamb=129600)
```

```
In [7]: # Setting up Unobserved Components and ARIMA model
        mod_uc_arima = sm.tsa.UnobservedComponents(gdax_btc_eur['Open'], 'rwalk', autoregressive
        res_uc_arima = mod_uc_arima.fit(method='powell', disp=False)
        print(res_uc_arima.summary())
```

```
                        Unobserved Components Results
==============================================================================
Dep. Variable:                   Open   No. Observations:                  119
Model:                    random walk   Log Likelihood                -747.551
                              + AR(4)   AIC                           1507.103
Date:                Tue, 17 Oct 2017   BIC                           1523.777
Time:                        17:16:19   HQIC                          1513.874
Sample:                    06-01-2017
                         - 09-27-2017
Covariance Type:                  opg
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
sigma2.level  1.199e+04   3472.899      3.454      0.001    5187.802    1.88e+04
sigma2.ar     3652.4924   2727.949      1.339      0.181   -1694.189    8999.174
ar.L1            0.1658      0.284      0.583      0.560      -0.392       0.723
ar.L2            0.0699      0.289      0.242      0.809      -0.496       0.636
ar.L3           -0.0713      0.187     -0.381      0.703      -0.438       0.295
ar.L4           -0.5106      0.239     -2.138      0.033      -0.979      -0.042
===================================================================================
Ljung-Box (Q):                       40.87   Jarque-Bera (JB):                   53.44
Prob(Q):                              0.43   Prob(JB):                            0.00
Heteroskedasticity (H):               2.04   Skew:                               -0.56
Prob(H) (two-sided):                  0.03   Kurtosis:                            6.10
===================================================================================

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

```
In [8]: # Setting up Unobserved components with stochastic cycle (UC)
        mod_uc = sm.tsa.UnobservedComponents(gdax_btc_eur['Open'], 'rwalk',
                                             cycle=True, stochastic_cycle=True, damped_cycle=Tru
        res_uc = mod_uc.fit(method='powell', disp=False)
        res_uc = mod_uc.fit(res_uc.params, disp=False)
        print(res_uc.summary())
```

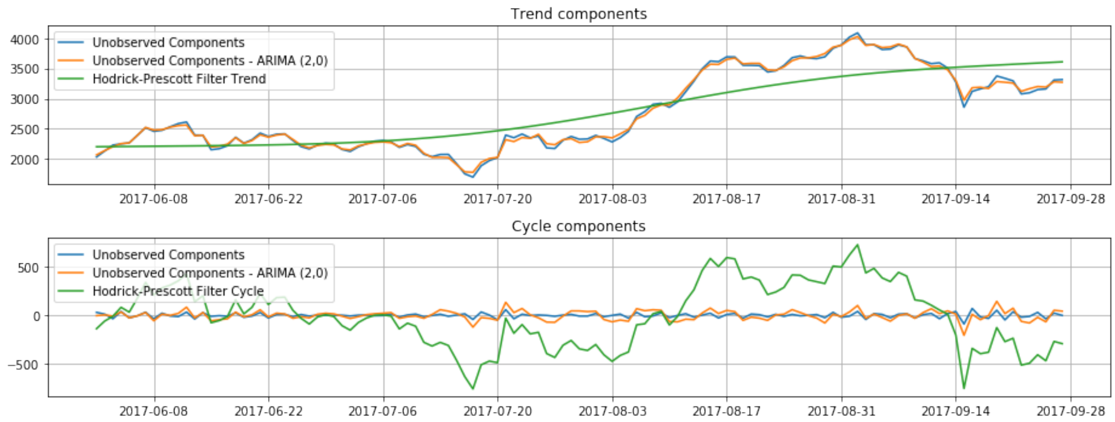
```
                            Unobserved Components Results
=====================================================================================
Dep. Variable:                              Open   No. Observations:                  119
Model:                               random walk   Log Likelihood                -736.733
                        + damped stochastic cycle   AIC                           1481.467
Date:                           Tue, 17 Oct 2017   BIC                           1492.583
Time:                                   17:16:19   HQIC                          1485.981
Sample:                               06-01-2017
                                    - 09-27-2017
Covariance Type:                             opg
==================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
sigma2.level     1.525e+04   2324.207      6.562      0.000    1.07e+04    1.98e+04
sigma2.cycle      588.3539    687.685      0.856      0.392    -759.483    1936.191
frequency.cycle     2.3496      0.181     12.989      0.000       1.995       2.704
damping.cycle       0.7782      0.203      3.831      0.000       0.380       1.176
==================================================================================
Ljung-Box (Q):                       42.03   Jarque-Bera (JB):                 41.53
Prob(Q):                              0.38   Prob(JB):                          0.00
Heteroskedasticity (H):               2.12   Skew:                             -0.44
Prob(H) (two-sided):                  0.02   Kurtosis:                          5.79
==================================================================================

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

```
In [9]: # Plot results
        fig, axes = plt.subplots(2, figsize=(13,5));
        axes[0].set(title='Trend components')
        axes[0].plot(gdax_btc_eur.index, res_uc.level.smoothed, label='Unobserved Components')
        axes[0].plot(gdax_btc_eur.index, res_uc_arima.level.smoothed, label='Unobserved Componen
        axes[0].plot(hp_trend, label='Hodrick-Prescott Filter Trend')
        axes[0].legend(loc='upper left')
        axes[0].grid()

        axes[1].set(title='Cycle components')
        axes[1].plot(gdax_btc_eur.index, res_uc.cycle.smoothed, label='Unobserved Components')
        axes[1].plot(gdax_btc_eur.index, res_uc_arima.autoregressive.smoothed, label='Unobserved
        axes[1].plot(hp_cycle, label='Hodrick-Prescott Filter Cycle')
        axes[1].legend(loc='upper left')
        axes[1].grid()

        fig.tight_layout();
```



```
In [108]: from pandas import read_csv
          from pandas import datetime
          from sklearn.metrics import mean_squared_error
          from math import sqrt
          from matplotlib import pyplot


          btc= gdax_btc_eur['Open']
          # load dataset
          # split data into train and test
          X = btc.values
          length = len(btc)
          train, test = X[1:-length], X[-length:]
          # walk-forward validation
          history = [x for x in train]
          predictions = list()
          for i in range(len(test)):
              predictions.append(history[-1])
              history.append(test[i])
          rmse = sqrt(mean_squared_error(test, predictions))
          print('RMSE: %.3f' % rmse)
```

```
    ---------------------------------------------------------------------------

    IndexError                                Traceback (most recent call last)

    <ipython-input-108-24bceccdb0c5> in <module>()
     16 predictions = list()
     17 for i in range(len(test)):
---> 18     predictions.append(history[-1])
     19     history.append(test[i])
```

```
 20 rmse = sqrt(mean_squared_error(test, predictions))


IndexError: list index out of range
```

```
In [97]: _test = pd.Series(test)
         _predictions = pd.Series(predictions)
```

```
In [98]: len(btc)
```

```
Out[98]: 119
```

```
In [102]: predicted = pd.DataFrame(_predictions)
          observed = pd.DataFrame(_test, index=btc.index)
```

```
In [106]: predicted.reset_index
          predicted.set_index(btc.index)
```

```
        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-106-e3293008ce6f> in <module>()
          1 predicted.reset_index
    ----> 2 predicted.set_index(btc.index-1)


        /home/rkopeinig/.local/lib/python2.7/site-packages/pandas/tseries/base.pyc in __sub__(se
        661                 return self._add_delta(-other)
        662             elif is_integer(other):
    --> 663                 return self.shift(-other)
        664             elif isinstance(other, (tslib.Timestamp, datetime)):
        665                 return self._sub_datelike(other)


        /home/rkopeinig/.local/lib/python2.7/site-packages/pandas/tseries/base.pyc in shift(self
        757
        758         if self.freq is None:
    --> 759             raise ValueError("Cannot shift with no freq")
        760
        761         start = self[0] + n * self.freq


        ValueError: Cannot shift with no freq
```

```
In [ ]:
```