# 003-Autocorrelation

November 2, 2017

```
In [1]: # author: René Kopeinig
        # script: Autocorrelation on cryptocurrency data
        # description: Compare autocorrelated Time-Series data of crypto currencies 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 [6]: # 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 [8]: # Get Litecoin, Ethereum and Bitcoin Data
        gdax_ltc_eur = get_data('GDAX/LTC_EUR')
        gdax_eth_eur = get_data('GDAX/ETH_EUR')
        gdax_btc_eur = get_data('GDAX/EUR')
```

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

```
In [9]: # Creating an average value form high and low
        gdax_ltc_eur['Mean'] = (gdax_ltc_eur['High']+gdax_ltc_eur['Low'])/2
        gdax_eth_eur['Mean'] = (gdax_eth_eur['High']+gdax_eth_eur['Low'])/2
        gdax_btc_eur['Mean'] = (gdax_btc_eur['High']+gdax_btc_eur['Low'])/2
```

```
In [12]: # Setting up dates
         gdax_eth_eur = gdax_eth_eur['2017-08-20':'2017-09-20']
         gdax_ltc_eur = gdax_ltc_eur['2017-08-20':'2017-09-20']
         gdax_btc_eur = gdax_btc_eur['2017-08-20':'2017-09-20']
```

```
In [14]: # Auto Correlation Function
         def auto_correlation(x):
             x = np.asarray(x)
             y = x-x.mean()
             result = np.correlate(y, y, mode='full')
             result = result[len(result)//2:]
             result /= result[0]
             return result
```

```
In [15]: # Calculate autocorrelation and transform result to Pandas Dataframe
         eth = auto_correlation(gdax_eth_eur['Mean'])
         eth = pd.DataFrame(data=eth, index=gdax_eth_eur.index, columns=['ETH'])

         btc = auto_correlation(gdax_btc_eur['Mean'])
         btc = pd.DataFrame(data=btc, index=gdax_btc_eur.index, columns=['BTC'])

         ltc = auto_correlation(gdax_ltc_eur['Mean'])
         ltc = pd.DataFrame(data=ltc, index=gdax_ltc_eur.index, columns=['LTC'])
```

```
In [16]: # Display results of autocorrelated price comparison of ETH, LTC and BTC
         eth = go.Scatter(x=eth.index, y=eth['ETH'], name='ETH')
         ltc = go.Scatter(x=ltc.index, y=ltc['LTC'], name='LTC')
         btc = go.Scatter(x=btc.index, y=btc['BTC'], name='BTC')

         data=[eth,ltc, btc]

         layout = dict(title = 'Autocorrelated Comparison of Ethereum, Litecoin and Bitcoin',
                       yaxis = dict(zeroline = False),
                       xaxis = dict(zeroline = False)
                      )
```

```
fig = dict(data=data, layout=layout)
py.iplot(fig)
```