Final Project: Tracking Cryptocurrency Price Movement - A Traders' Toolkit

Blockchain Expert: Shubham Surana

Email: ss9227@stern.nyu.edu

This project examines the cryptocurrency industry and how prices of individual currencies experience price volatility in correlation to other currencies in the market. One of the big questions I seek to answer is whether individual cryptocurrencies have a stronger correlation to the price movements of other cryptos and if that correlation has increased or decreased from the recent spike in trading in the last few months.

While bitcoin has now become mainstream, the aggregate cryptocurrency market is still an infant industry. Proponents of blockchain have hailed the technology as revolutionary and analogized it to the internet in the 1980s. Yet, given their decentralized nature, world governments have thus far struggled to pass comprehensive regulations on the industry. Some have opted for outright trading bans (China, Cambodia, Bangladesh etc.) due to fears of money-laundering and fruadulent ICOs (initial coin offerings). Because a coin's value is determined by its expected future rate of adoption, I hypothesize that as more cryptocurrencies come into further adoption, the market will rise in tandem. I expect that my calculations will show a high degree of correlations between the price movements of different cryptocurrencies and I hope that this will project can provide powerful tools for cryto traders to visualize the day-to-day movements of cryptocurrencies of their choosing.

In my analysis, I will attempt to chart the prices of bitcoin and 6 other altcoins (Ethereum, Ripple, EOS, Litecoin, Stellarlumens, and NEO). I plan on creating price charts over the past 3 years and then creating a correlation heatmap to help traders visualize the movement of different cryptocurrencies in a day and relative ot each other.

I will also display meaningul statistics based on the data in my final product.

Data Report

Overview:

The data for my project comes from a variety of cryptocurrency exchanges (Kraken (https://www.kraken.com), Coinbase (https://www.coinbase.com) and Bitstamp (https://www.bitstamp.net)). I average the prices retrieved from their APIs since cryptocurrency prices are not the same on every different exchange, unlike equities. Addiitonally, I'll also use Quandl's API to retrieve pricing data since it is considered highly accurate and can pull data from these exchanges.

The key series that I seek to retrieve are the average prices for the past two years of bitcoin and the 6 different altcoins I have aforementioned. In my analysis, I will produce visually simple, color-coded charts that show the changes in price of the 7 currencies.

Access

I will access the information using the APIs from each of the above exchanges.

Requisite Packages

These are the packages I will be required to import:

In [33]:
```
import os
import numpy as np # Used to do my calculations
import pandas as pd # Pandas dataframes will allow me to make alterations to my data
import pickle # I found out that pickle will allow me to save my data as a file so that I do not have to download it repeatedely
import quandl # Quandl will be used to import bitcoin pricing data
from datetime import datetime # Datetime will help place the necessary dates on my chart
import plotly.offline as py # I want to use plotly offline in case I need to make adjustments to the pricing data
import plotly.graph_objs as go # This will help to chart my pricing time series
import plotly.figure_factory as ff # Figure factory module creates much nicer graphs not included in plotly.js
py.init_notebook_mode(connected=True)
```

Important Note:

I'm using Plotly instead of Matplotlib in order to create a more interactive experience for my user. Again, this project is meant to serve as a toolkit for traders and I've found that charts created from Plotly make it easier to glance over specific data points.

Now it's time to retreive my bitcoin price data:

In [34]:
```
# First I want to define a function that will be able to pull prices from Quandl.
# I'll use try - except to make sure that any data that cannot be loaded is cached in another fil
e

def quandl_data(quandl_id):

    cache_path = '{}.pkl'.format(quandl_id).replace('/','-')
    try:
        f = open(cache_path, 'rb')
        df = pickle.load(f)
        print('Loaded {} from cache'.format(quandl_id))
    except (OSError) 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
```

Now I'll pull Kraken's bitcoin price data using the function defined above:

In [35]:
```
btc_usd = quandl_data('BCHARTS/KRAKENUSD')
```

```
Loaded BCHARTS/KRAKENUSD from cache
```

Using the head method, I can analyze the first 10 rows:

In [36]:
```
btc_usd.head(10)
```

Out[36]:

	Open	High	Low	Close	Volume (BTC)	Volume (Currency)	Weighted Price
Date							
2014-01-07	874.67040	892.06753	810.00000	810.00000	15.622378	13151.472844	841.835522
2014-01-08	810.00000	899.84281	788.00000	824.98287	19.182756	16097.329584	839.156269
2014-01-09	825.56345	870.00000	807.42084	841.86934	8.158335	6784.249982	831.572913
2014-01-10	839.99000	857.34056	817.00000	857.33056	8.024510	6780.220188	844.938794
2014-01-11	858.20000	918.05471	857.16554	899.84105	18.748285	16698.566929	890.671709
2014-01-12	899.96114	900.93989	833.00001	860.00000	25.429433	21880.878976	860.454846
2014-01-13	847.32152	859.99999	815.00000	835.00000	25.869127	21529.839497	832.260007
2014-01-14	835.00000	877.29300	805.00000	831.00000	31.662881	26756.278447	845.036122
2014-01-15	831.00000	864.00000	828.00000	850.00364	6.707565	5698.139372	849.509430
2014-01-16	853.00000	865.00000	824.00000	826.97077	28.602014	24229.478350	847.124912

Seems like Quandl's earliest price goes back to the beginning of 2014 when bitcoin was already at approximately 800 USD. Still, not bad considering it's current price is at $9k.

In [37]:
```
eth_usd = quandl_data('BITFINEX/ETHUSD')
```

```
Loaded BITFINEX/ETHUSD from cache
```

In [38]:
```
eth_usd.head(10)
```

Out[38]:

Date	High	Low	Mid	Last	Bid	Ask	Volume
2016-03-14	14.950	11.400	12.9325	12.925	12.925	12.940	22922.937642
2016-03-15	13.421	11.607	12.9665	12.765	12.924	13.009	18157.378517
2016-03-16	13.890	12.621	12.8200	12.800	12.800	12.840	5431.139409
2016-03-17	12.893	10.436	11.2450	11.368	11.172	11.318	25158.103084
2016-03-18	11.200	8.338	10.8155	10.750	10.751	10.880	46391.436185
2016-03-19	11.170	9.776	10.1925	10.212	10.117	10.268	21406.585692
2016-03-20	10.881	9.559	10.0955	10.070	10.055	10.136	9491.890171
2016-03-21	11.940	10.070	11.8460	11.877	11.808	11.884	10952.342211
2016-03-22	12.048	11.075	11.5140	11.572	11.478	11.550	8631.587339
2016-03-23	12.376	11.201	12.3355	12.350	12.321	12.350	7411.455061

Uh oh. Seems like Ethereum data only goes as far back as 2016. Will have to find a way to merge this with the bitcoin price data.

In [39]:
```
neo_usd = quandl_data('BITFINEX/NEOUSD')
```

```
Loaded BITFINEX/NEOUSD from cache
```

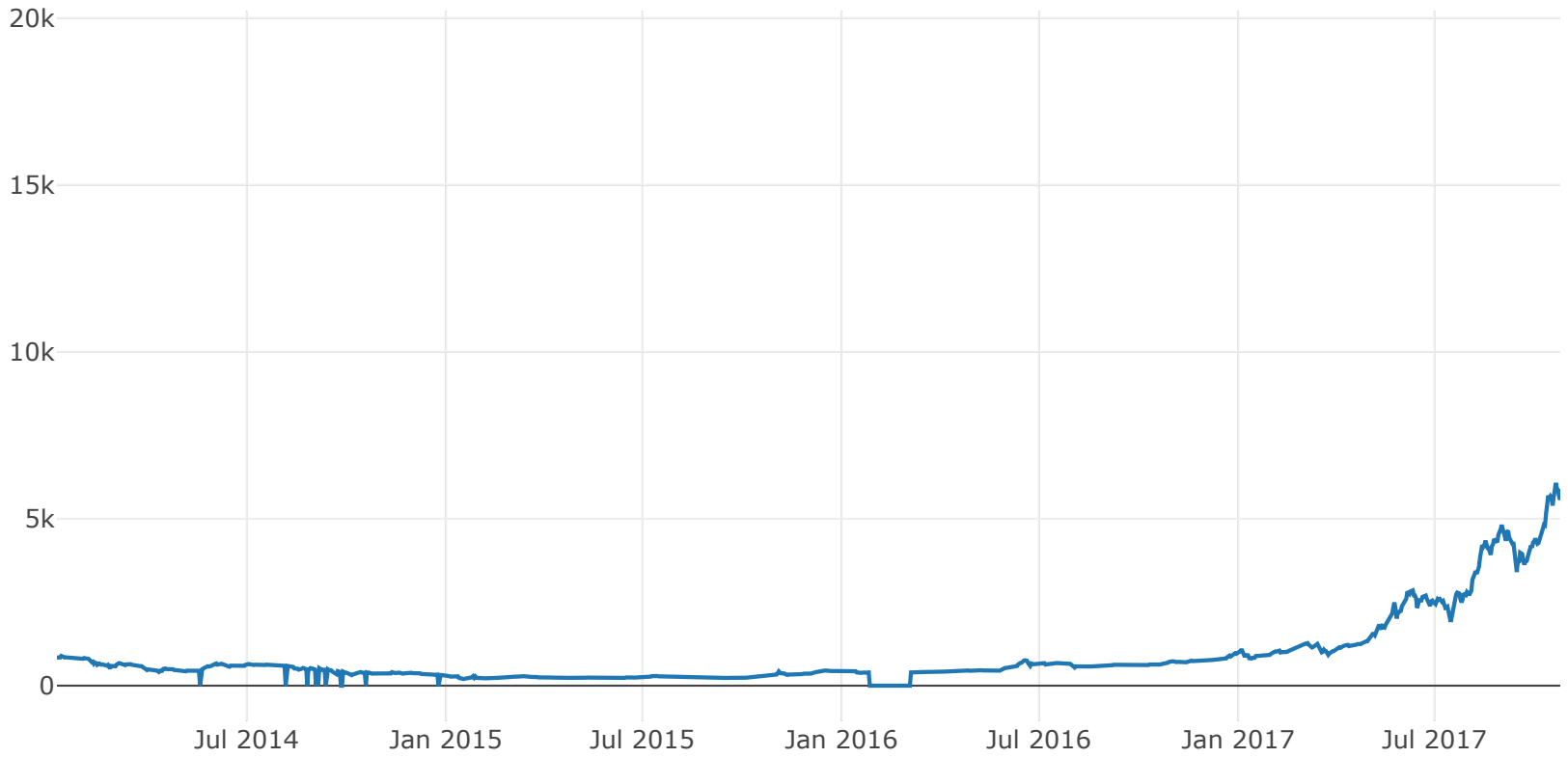
In [40]:
```
neo_usd.head(10)
```

Out[40]:

	High	Low	Mid	Last	Bid	Ask	Volume
Date							
2017-09-07	39.800	22.300	30.5525	30.411	30.415	30.690	269772.012257
2017-09-08	31.637	22.320	25.9525	25.900	25.912	25.993	291852.004274
2017-09-09	26.270	22.180	23.3955	23.351	23.358	23.433	139533.241697
2017-09-10	25.400	19.529	24.0725	24.202	23.943	24.202	175813.285869
2017-09-11	24.500	21.509	21.9105	21.880	21.873	21.948	79686.038107
2017-09-12	23.800	19.880	20.8865	20.801	20.806	20.967	148076.452223
2017-09-13	21.395	17.534	20.5725	20.507	20.442	20.703	141692.110832
2017-09-14	20.986	14.700	16.5580	16.744	16.510	16.606	167425.957702
2017-09-15	19.330	13.190	18.8075	18.860	18.755	18.860	244785.042573
2017-09-16	23.197	18.046	20.8640	20.940	20.788	20.940	178461.137134

Let's try charting some of these cryptos based on the data I've pulled.

In [41]:
```
btc_chart = go.Scatter(x=btc_usd.index, y=btc_usd['Weighted Price'])
py.iplot([btc_chart])
```



Not bad, thought there are some bumps from earlier periods where pricing information is not available. Also, notice that because this graph is charted with plot.ly, hovering over the chart will allow you to see the price at any given data point.

Okay, time to tackle those gaps in the dataset. It shows that the Kraken exchange did not have all the prices for BTC from earlier times, so instead, I'll tap into some other exchanges and pull BTC price data from them. In particular, I'll pull data from Coinbase and BitStamp.

In [60]:
```
exchanges = ['COINBASE', 'BITSTAMP']

exchange_prices = {}

for exchange in exchanges:
    exchange_code = 'BCHARTS/{}USD'.format(exchange)
    btc_exchange_df = quandl_data(exchange_code)
    exchange_prices[exchange] = btc_exchange_dfb
```

```
Loaded BCHARTS/COINBASEUSD from cache
Loaded BCHARTS/BITSTAMPUSD from cache
```

I'll have to merge these dataframes now like how we did in class. I'll also define a function to merge the dataframes since it'll make it easier for the other altcoins.

In [66]:
```
def merge_dataframes(dataframes, labels, col):

    series_set = {}
    for index in range(len(dataframes)):
        series_set[labels[index]] = dataframes[index][col]

    return pd.DataFrame(series_set)
```

Now I want to merge the dataframes together based on the numbers in the Weighted Price column.

In [69]:
```
btc_usd_datasets = merge_dataframes(list(exchange_prices.values()), list(exchange_prices.keys()),
'Weighted Price')
```

Let's take a quick look at how these dataframes are looking using the tail method:

In [70]:
```
btc_usd_datasets.tail()
```

Out[70]:

	BITSTAMP	COINBASE
Date		
2018-05-10	9234.640532	9200.444620
2018-05-11	8697.045048	8674.635184
2018-05-12	8409.414442	8425.543638
2018-05-13	8569.273096	8571.352121
2018-05-14	8595.013518	8626.122926

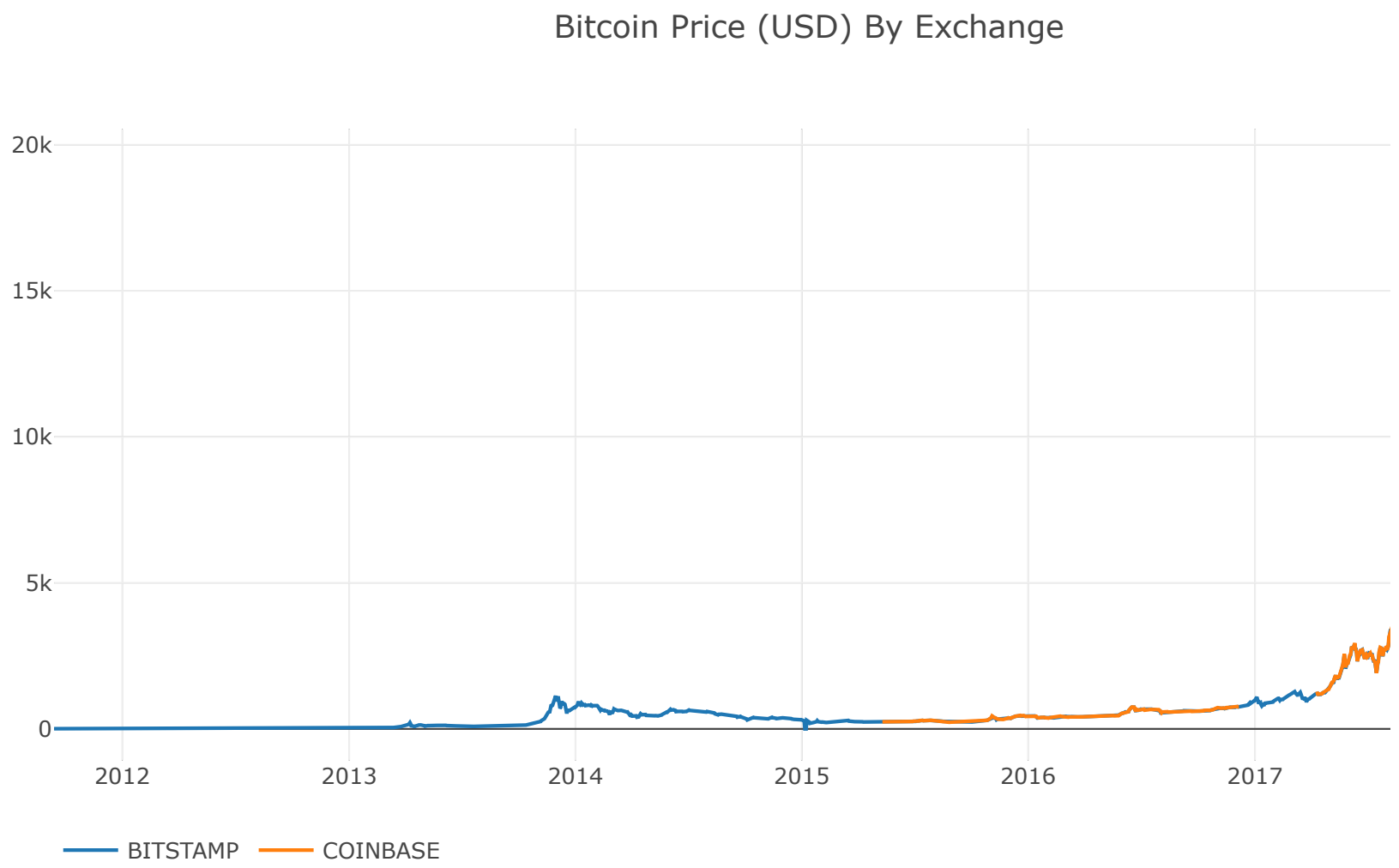
Now I want to set up a function that will allow me to make a scatter plot of the merged data frames. This part was quite difficult and I admittedely did receive some help to write the code for this function.

In [71]:
```
def df_scatter(df, title, seperate_y_axis=False, y_axis_label='', scale='linear', initial_hide=Fa
lse):

    label_arr = list(df)
    series_arr = list(map(lambda col: df[col], label_arr))

    layout = go.Layout(
        title=title,
        legend=dict(orientation="h"),
        xaxis=dict(type='date'),
        yaxis=dict(
            title=y_axis_label,
```

```
            showticklabels= not seperate_y_axis,
            type=scale
        )
    )

    y_axis_config = dict(
        overlaying='y',
        showticklabels=False,
        type=scale )

    visibility = 'visible'
    if initial_hide:
        visibility = 'legendonly'


    trace_arr = []
    for index, series in enumerate(series_arr):
        trace = go.Scatter(
            x=series.index,
            y=series,
            name=label_arr[index],
            visible=visibility
        )

        # Add seperate axis for the series
        if seperate_y_axis:
            trace['yaxis'] = 'y{}'.format(index + 1)
            layout['yaxis{}'.format(index + 1)] = y_axis_config
        trace_arr.append(trace)

    fig = go.Figure(data=trace_arr, layout=layout)
    py.iplot(fig)
```

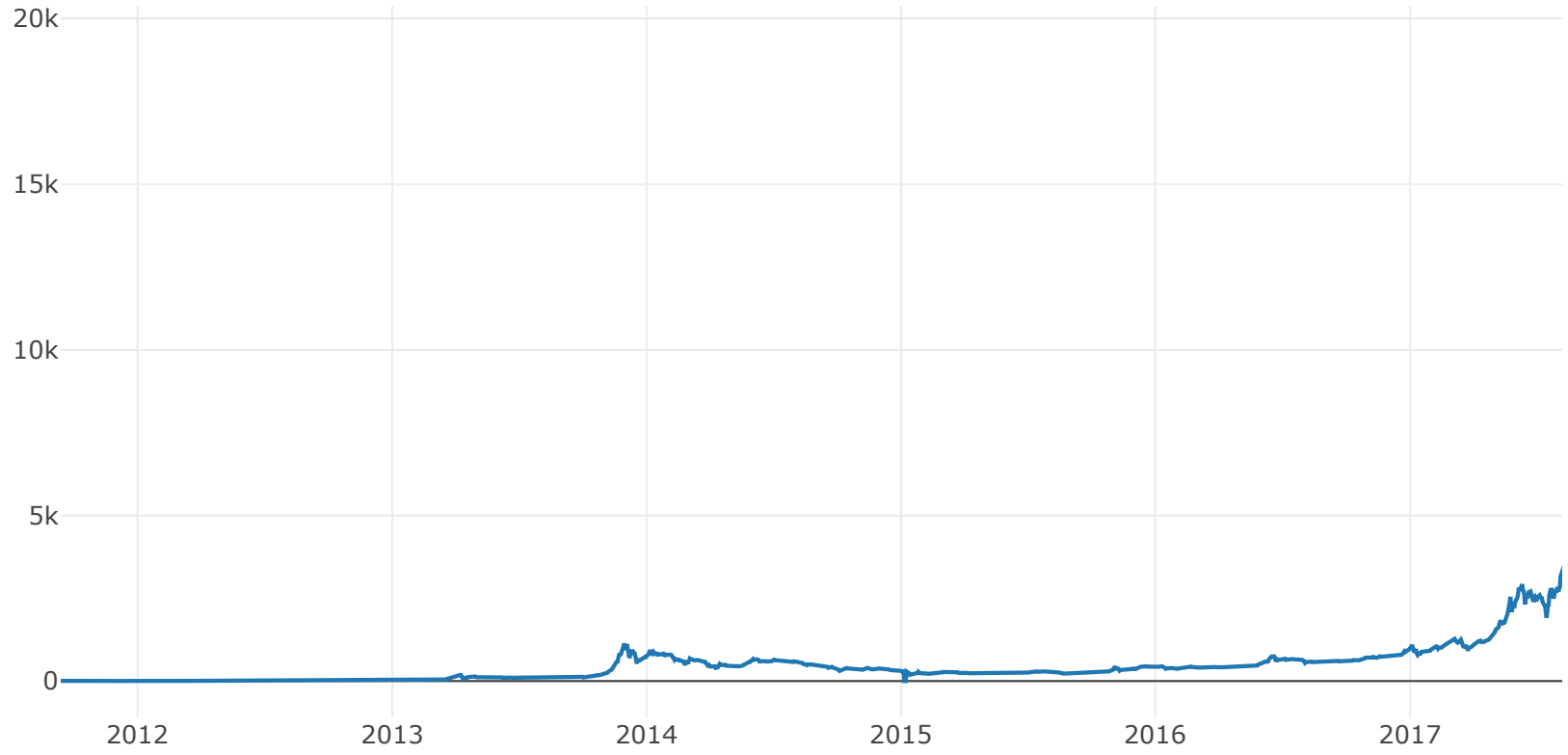
Now it's time to graph the scatter plot from the previous function using our Bitcoin dataset. Those bumps should be gone now (fingers crossed).

In [72]:
```
df_scatter(btc_usd_datasets, 'Bitcoin Price (USD) By Exchange')
```



The next logical step woud be to merge the data from the two exchanges into one dataframe and chart that as the average price.

In [75]:
```
btc_usd_datasets['avg_btc_price_usd'] = btc_usd_datasets.mean(axis=1)
btc_chart = go.Scatter(x=btc_usd_datasets.index, y=btc_usd_datasets['avg_btc_price_usd'])
py.iplot([btc_chart])
```



Looks good! Time to move on to our other altcoins.

Im going to be using the Poloniex API (https://poloniex.com/support/api/) to pull data for altcoins. I incurred a small problem here. The Poloniex API sends back the data in Java Script Object Notation. From some assistance, I found that I would have set up a function to download and cache the JSON data and then use another function to take that cached data and create a dataframe from it. The result is as follows:

In [85]:
```
def get_json_data(json_url, cache_path):
    '''Download and cache JSON data, return as a dataframe.'''
    try:
        f = open(cache_path, 'rb')
        df = pickle.load(f)
        print('Loaded {} from cache'.format(json_url))
    except (OSError, IOError) as e:
        print('Downloading {}'.format(json_url))
        df = pd.read_json(json_url)
        df.to_pickle(cache_path)
        print('Cached response at {}'.format(json_url, cache_path))
    return df
```

Now a function to take the cached data and put it into a dataframe.

In [86]:
```
base_polo_url = 'https://poloniex.com/public?command=returnChartData&currencyPair={}&start={}&end
={}&period={}'
start_date = datetime.strptime('2015-01-01', '%Y-%m-%d') # get data from the start of 2015
end_date = datetime.now() # up until today
pediod = 86400 # pull daily data (86,400 seconds per day)

def get_crypto_data(poloniex_pair):
    '''Retrieve cryptocurrency data from poloniex'''
    json_url = base_polo_url.format(poloniex_pair, start_date.timestamp(), end_date.timestamp(),
pediod)
    data_df = get_json_data(json_url, poloniex_pair)
    data_df = data_df.set_index('date')
    return data_df
```

Important Note:

Most cryptocurrencies cannot be bought directly with US dollars or other fiat currencies. Instead, traders purchase them through Bitcoin or Ethereum. Hence, these functions will pull prices in terms of Bitcoin that must be converted later on to USD.

In [95]:
```
altcoins = ['ETH','LTC','XRP','ETC','STR','DASH','XMR']

altcoin_data = {}
for altcoin in altcoins:
    coinpair = 'BTC_{}'.format(altcoin)
    crypto_price_df = get_crypto_data(coinpair)
    altcoin_data[altcoin] = crypto_price_df
```

```
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_ETH&start=142008840
0.0&end=1526354651.510528&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_LTC&start=142008840
0.0&end=1526354651.510528&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_XRP&start=142008840
0.0&end=1526354651.510528&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_ETC&start=142008840
0.0&end=1526354651.510528&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_STR&start=142008840
0.0&end=1526354651.510528&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_DASH&start=14200884
00.0&end=1526354651.510528&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_XMR&start=142008840
0.0&end=1526354651.510528&period=86400 from cache
```

Now we should have 7 dataframes with each altcoin in it's BTC pricing terms. Let's check they all came out properly by viewing each of their tails:

In [101]:
```
altcoin_data['ETH'].tail()
```

Out[101]:

	close	high	low	open	quoteVolume	volume	weightedAverage
date							
2018-05-11	0.080685	0.082200	0.078188	0.080100	16861.522020	1347.579119	0.079920
2018-05-12	0.080790	0.081273	0.076765	0.080699	28121.001767	2212.132849	0.078665
2018-05-13	0.083905	0.085099	0.079852	0.080790	13717.145187	1141.573629	0.083222
2018-05-14	0.083798	0.084642	0.082500	0.083903	12654.649756	1057.506214	0.083567
2018-05-15	0.083560	0.084204	0.082897	0.083899	937.800851	78.317675	0.083512

In [102]:
```
altcoin_data['LTC'].tail()
```

Out[102]:

	close	high	low	open	quoteVolume	volume	weightedAverage
date							
2018-05-11	0.016220	0.016584	0.016000	0.016462	22925.785026	372.237663	0.016237
2018-05-12	0.016695	0.016829	0.016000	0.016220	14634.320799	240.225313	0.016415
2018-05-13	0.016645	0.016835	0.016245	0.016718	10546.424955	174.974390	0.016591
2018-05-14	0.017000	0.017302	0.016182	0.016612	25465.460738	426.486623	0.016748
2018-05-15	0.016791	0.017122	0.016240	0.016995	14427.558248	240.920851	0.016699

In [104]:
```
altcoin_data['XRP'].tail()
```

Out[104]:

	close	high	low	open	quoteVolume	volume	weightedAverage
date							
2018-05-11	0.000081	0.000084	0.000074	0.000083	2.178301e+07	1724.011746	0.000079
2018-05-12	0.000081	0.000082	0.000077	0.000081	8.019545e+06	641.070434	0.000080
2018-05-13	0.000084	0.000086	0.000080	0.000081	7.406971e+06	616.415578	0.000083
2018-05-14	0.000084	0.000086	0.000082	0.000084	6.787215e+06	571.177930	0.000084
2018-05-15	0.000084	0.000084	0.000084	0.000084	3.049468e+05	25.614863	0.000084

In [105]:
```
altcoin_data['ETC'].tail()
```

Out[105]:

	close	high	low	open	quoteVolume	volume	weightedAverage
date							
2018-05-11	0.002101	0.002207	0.001961	0.002167	229922.333064	475.729875	0.002069
2018-05-12	0.002139	0.002150	0.002000	0.002103	129979.800847	270.347543	0.002080
2018-05-13	0.002150	0.002198	0.002110	0.002136	69385.324489	149.786843	0.002159
2018-05-14	0.002156	0.002160	0.002083	0.002150	89687.854001	190.553301	0.002125
2018-05-15	0.002154	0.002181	0.002146	0.002154	3842.871907	8.315512	0.002164

In [106]:
```
altcoin_data['STR'].tail()
```

Out[106]:

	close	high	low	open	quoteVolume	volume	weightedAverage
date							
2018-05-11	0.000037	0.000039	0.000034	0.000038	2.964081e+07	1069.745678	0.000036
2018-05-12	0.000042	0.000044	0.000036	0.000037	4.359670e+07	1753.345204	0.000040
2018-05-13	0.000043	0.000044	0.000041	0.000042	1.895475e+07	810.503757	0.000043
2018-05-14	0.000042	0.000043	0.000041	0.000043	8.254761e+06	345.884986	0.000042
2018-05-15	0.000041	0.000042	0.000041	0.000042	3.857064e+05	16.022714	0.000042

In [107]:
```
altcoin_data['DASH'].tail()
```

Out[107]:

	close	high	low	open	quoteVolume	volume	weightedAverage
date							
2018-05-11	0.046599	0.048948	0.045353	0.046200	3872.455504	182.006468	0.047000
2018-05-12	0.047604	0.047712	0.045450	0.046599	1147.072254	53.164981	0.046348
2018-05-13	0.047774	0.048391	0.046693	0.047604	1089.926062	52.021598	0.047729
2018-05-14	0.049824	0.052000	0.045702	0.047909	5976.991568	292.918680	0.049008
2018-05-15	0.049077	0.049824	0.048767	0.049824	245.787999	12.130966	0.049355

In [108]:
```
altcoin_data['XMR'].tail()
```

Out[108]:

	close	high	low	open	quoteVolume	volume	weightedAverage
date							
2018-05-11	0.023422	0.024100	0.023015	0.024001	13939.411440	328.252272	0.023548
2018-05-12	0.023715	0.023968	0.023125	0.023422	9700.392044	228.750627	0.023582
2018-05-13	0.023890	0.024564	0.023700	0.023715	7601.180020	184.039092	0.024212
2018-05-14	0.024553	0.024800	0.023522	0.023890	13249.674544	320.727481	0.024206
2018-05-15	0.024445	0.024565	0.024415	0.024553	891.873312	21.834842	0.024482

Great, now we have the rate of each coin in terms of Bitcoin. Because we also have the BItcoin pricing data, we can convert the altcoin prices into USD.

In [110]:
```
for altcoin in altcoin_data.keys():
    altcoin_data[altcoin]['price_usd'] =  altcoin_data[altcoin]['weightedAverage'] * btc_usd_data
sets['avg_btc_price_usd']
```

The previous line of code will create a new column with the USD values for each altcoin.

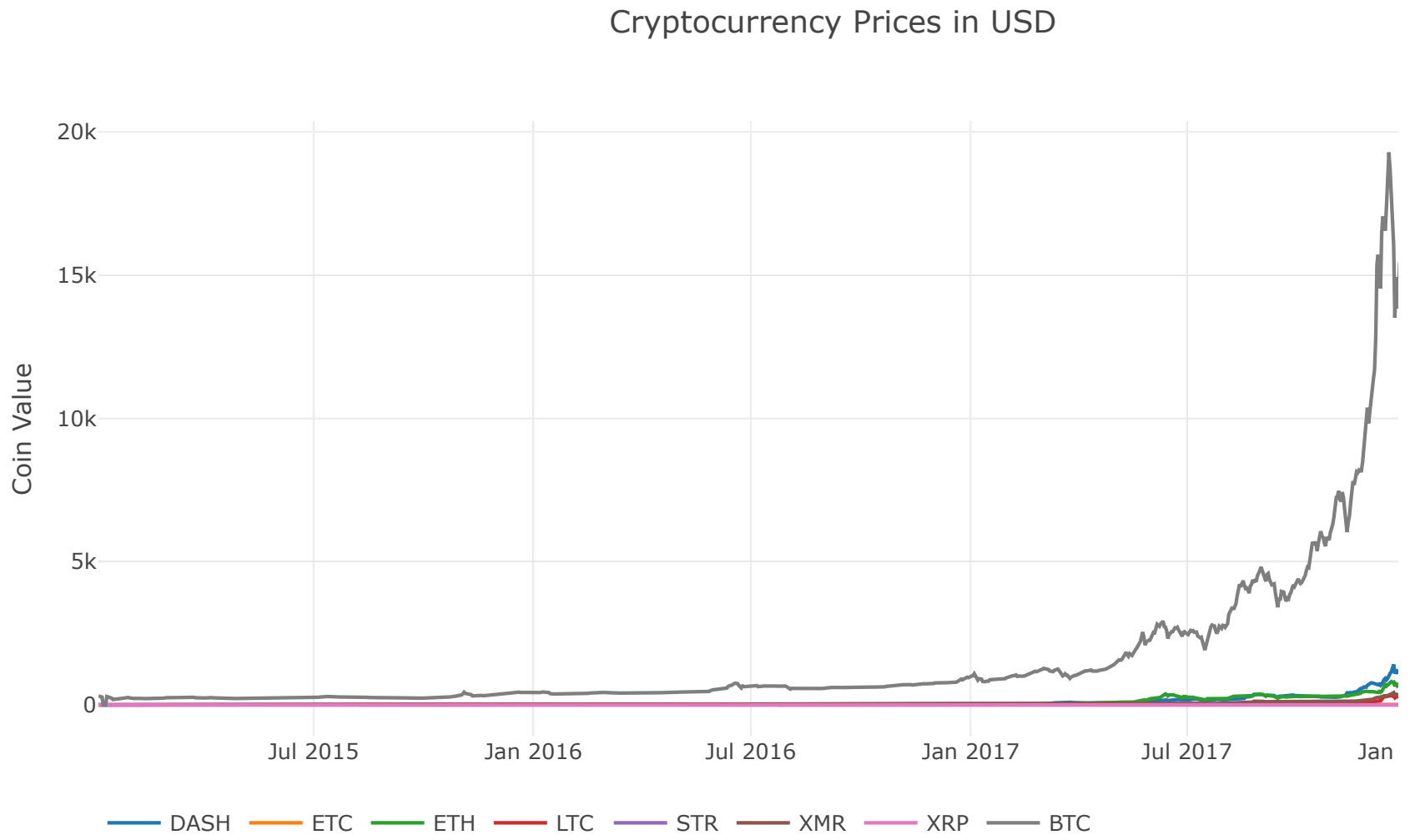
Now let's merge the dataframes to create a single dataframe with the USD prices for each altcoin:

In [111]:
```
combined_df = merge_dfs_on_column(list(altcoin_data.values()), list(altcoin_data.keys()), 'price_
usd')
```

In [115]:
```
# Also have to add our earlier BTC dataset
combined_df['BTC'] = btc_usd_datasets['avg_btc_price_usd']
```

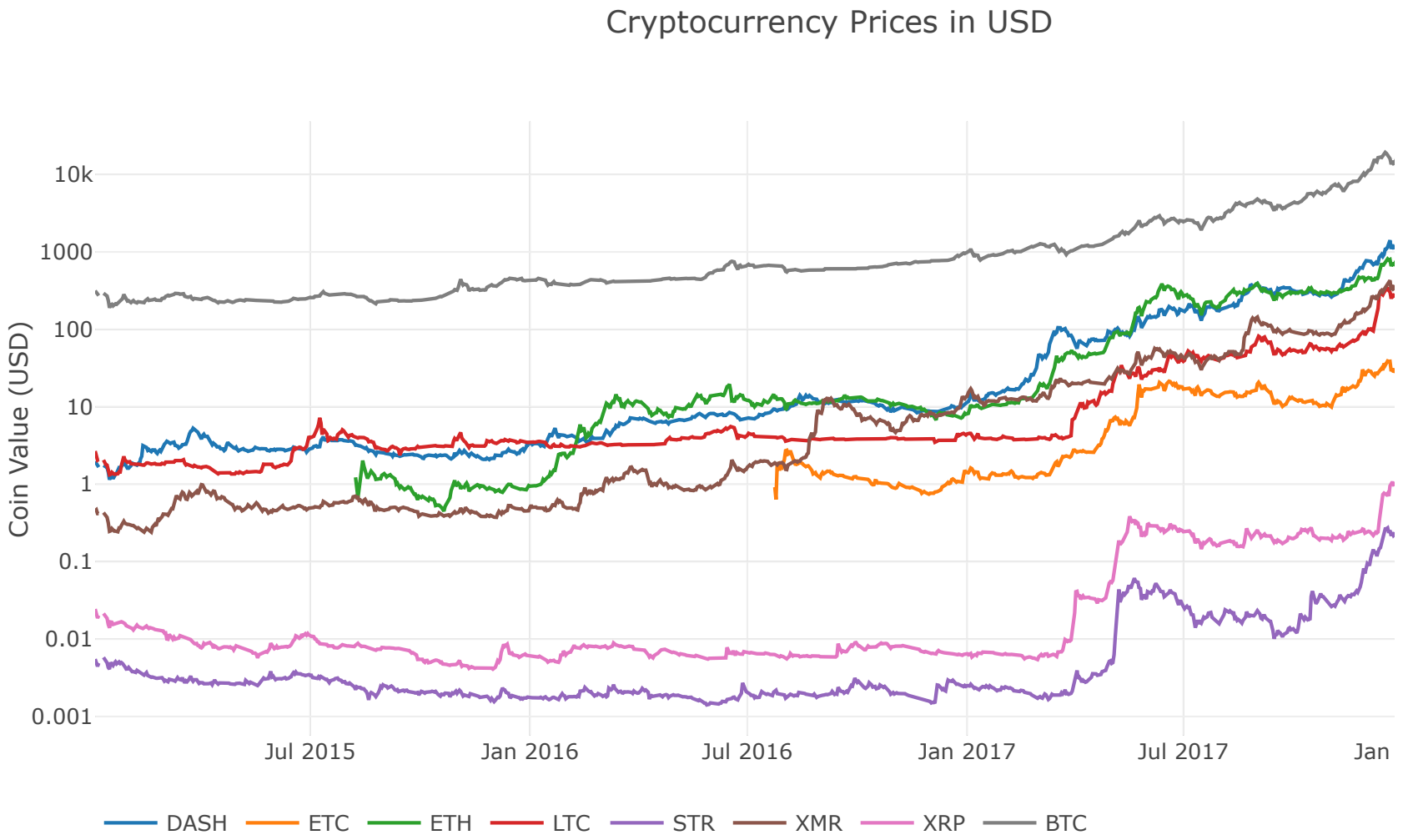
Now I'm ready to print the scatter plot that will encompass all the dataframes with USD prices that I created.

In [124]:
```
df_scatter(combined_df, 'Cryptocurrency Prices in USD', seperate_y_axis=False, y_axis_label='Coin Value')
```



This doesn't seem to good since Bitcoin's high price dominates and overshadows the prices of the other cryptocurrencies. I can try to change this by graphing on a logarithmic scale, as we discussed in class earlier.

In [125]:
```
df_scatter(combined_df, 'Cryptocurrency Prices in USD', seperate_y_axis=False, y_axis_label='Coin Value (USD)', scale='log')
```



This is way better! It's important to note the use of a logarithmic scale because while Bitcoin's price has etched into the tens of thousands, many cryptocurrencies, like Ripple, still trade for less than a dollar. Graphing on a logarithmic scale makes it more visually understandable to highlight changes in prices and also give greater account for price changes of low-prices currencies.

Okay, now that my first goal is accomplished, I also want to be able to create a correlation heatmap and compute meaningful statistics of each of these cryptocurrencies relative to each other. Traders realize that a good portfolio will usually consist of trades that have disparate correlations in the case that the market trades in a certain way. This heatmap will help the trader determine which currencies are high in correlation and that he/she may not want to hold too much of. It will also indicate which currencies have low correlations and may be a good addition to their portfolio.

I'm going to use the corr() function to and include the pct_change method to get the actual changes in percentage returns of each currency rather than their actual price since the huge price swings of bitcoin will inevitably alter the data. Let's do this for 3 years from 2016 to 2018.

In [134]:
```
combined_df_2016 = combined_df[combined_df.index.year == 2016]
combined_df_2016.pct_change().corr()
```

Out[134]:

	DASH	ETC	ETH	LTC	STR	XMR	XRP	BTC
DASH	1.000000	0.005483	0.119801	-0.016489	0.053751	0.118212	0.083971	-0.021553
ETC	0.005483	1.000000	-0.182346	-0.128905	-0.102050	-0.105059	-0.053067	-0.167863
ETH	0.119801	-0.182346	1.000000	-0.067030	0.032548	0.085570	0.082533	-0.010941
LTC	-0.016489	-0.128905	-0.067030	1.000000	0.109407	0.126444	0.049851	0.749809
STR	0.053751	-0.102050	0.032548	0.109407	1.000000	0.025409	0.317250	0.072833
XMR	0.118212	-0.105059	0.085570	0.126444	0.025409	1.000000	0.024552	0.123194
XRP	0.083971	-0.053067	0.082533	0.049851	0.317250	0.024552	1.000000	0.037478
BTC	-0.021553	-0.167863	-0.010941	0.749809	0.072833	0.123194	0.037478	1.000000

In [135]:
```
combined_df_2017 = combined_df[combined_df.index.year == 2017]
combined_df_2017.pct_change().corr()
```

Out[135]:

	DASH	ETC	ETH	LTC	STR	XMR	XRP	BTC
DASH	1.000000	0.389938	0.508660	0.342315	0.185696	0.501098	0.090502	0.311911
ETC	0.389938	1.000000	0.601750	0.482490	0.211877	0.448989	0.112351	0.417603
ETH	0.508660	0.601750	1.000000	0.438195	0.261143	0.555634	0.209824	0.410218
LTC	0.342315	0.482490	0.438195	1.000000	0.309010	0.438509	0.322565	0.421429
STR	0.185696	0.211877	0.261143	0.309010	1.000000	0.329616	0.509714	0.234268
XMR	0.501098	0.448989	0.555634	0.438509	0.329616	1.000000	0.225240	0.412321
XRP	0.090502	0.112351	0.209824	0.322565	0.509714	0.225240	1.000000	0.127016
BTC	0.311911	0.417603	0.410218	0.421429	0.234268	0.412321	0.127016	1.000000

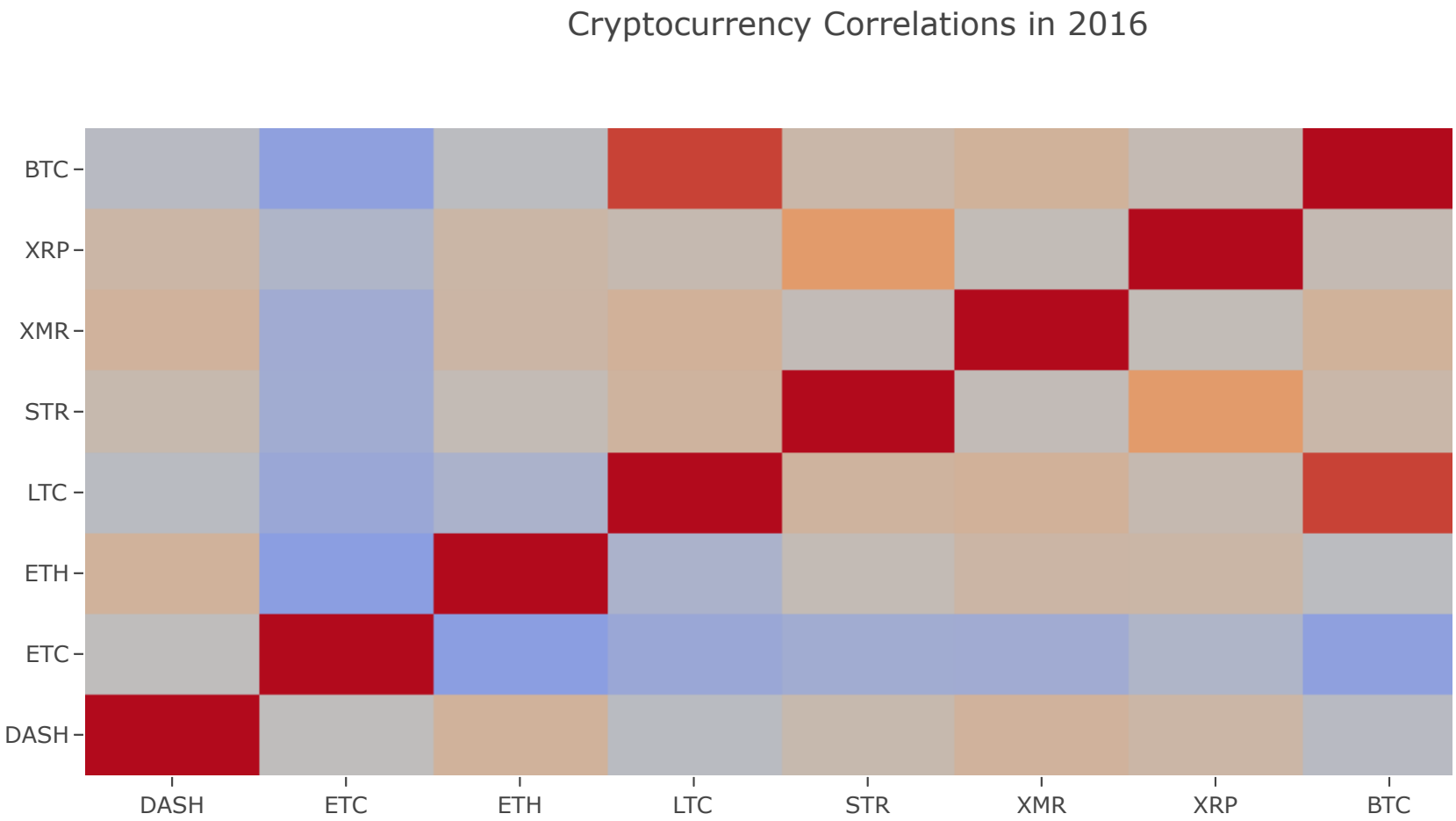
In [136]:
```
combined_df_2018 = combined_df[combined_df.index.year == 2018]
combined_df_2018.pct_change().corr()
```

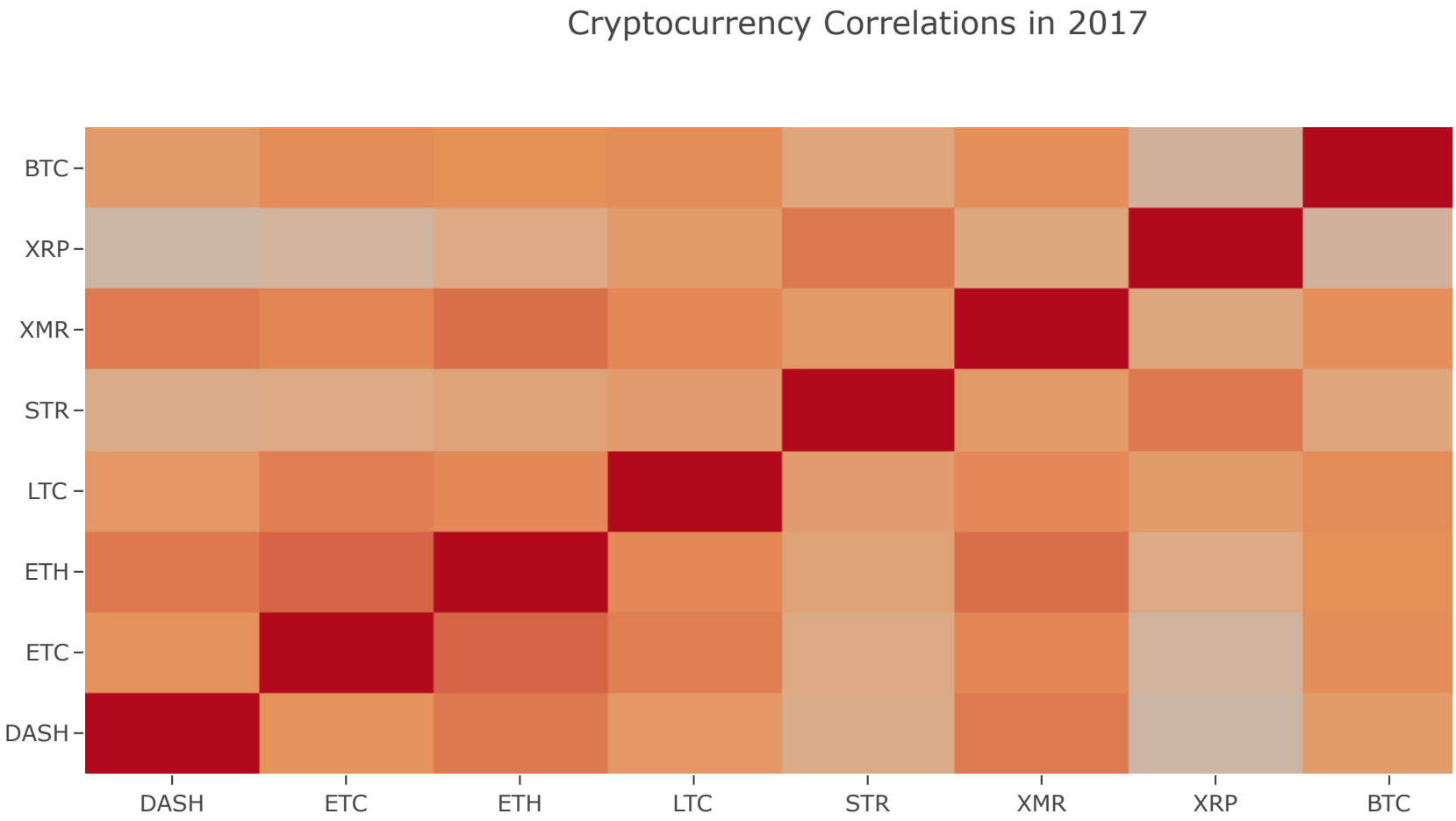
Out[136]:

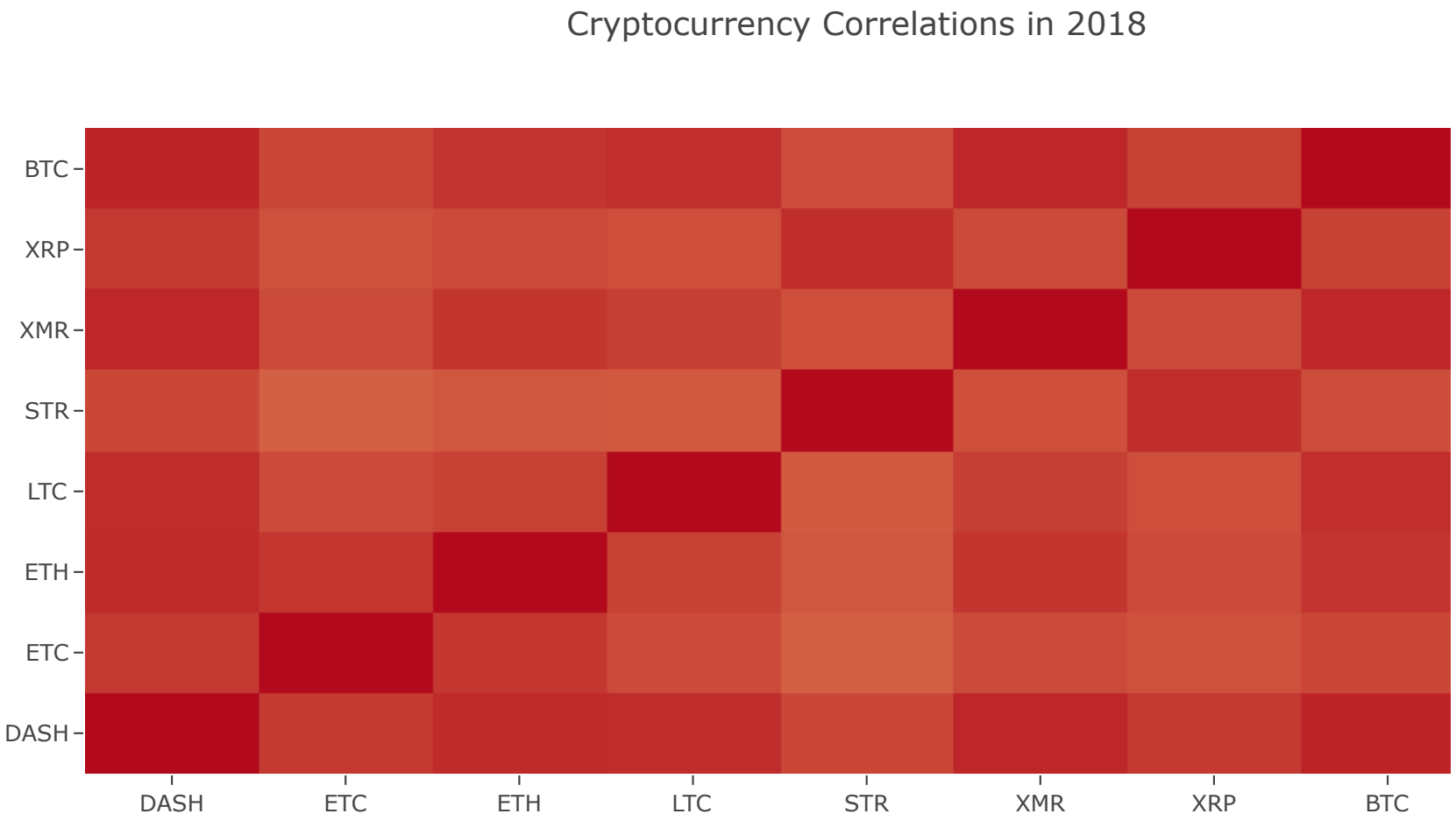
	DASH	ETC	ETH	LTC	STR	XMR	XRP	BTC
DASH	1.000000	0.787266	0.855558	0.848683	0.727598	0.871316	0.790699	0.884035
ETC	0.787266	1.000000	0.801926	0.714458	0.617159	0.710969	0.682873	0.727866
ETH	0.855558	0.801926	1.000000	0.754329	0.659086	0.804363	0.714896	0.814563
LTC	0.848683	0.714458	0.754329	1.000000	0.643882	0.764065	0.697375	0.838935
STR	0.727598	0.617159	0.659086	0.643882	1.000000	0.691198	0.841755	0.703879
XMR	0.871316	0.710969	0.804363	0.764065	0.691198	1.000000	0.711701	0.869232
XRP	0.790699	0.682873	0.714896	0.697375	0.841755	0.711701	1.000000	0.754874
BTC	0.884035	0.727866	0.814563	0.838935	0.703879	0.869232	0.754874	1.000000

In [128]:
```
def correlation_heatmap(df, title, absolute_bounds=True):
    '''Plot a correlation heatmap for the entire dataframe'''
    heatmap = go.Heatmap(
        z=df.corr(method='pearson').as_matrix(),
        x=df.columns,
        y=df.columns,
        colorbar=dict(title='Pearson Coefficient'),
    )

    layout = go.Layout(title=title)

    if absolute_bounds:
        heatmap['zmax'] = 1.0
        heatmap['zmin'] = -1.0

    fig = go.Figure(data=[heatmap], layout=layout)
    py.iplot(fig)
```

In [137]:
```
correlation_heatmap(combined_df_2016.pct_change(), "Cryptocurrency Correlations in 2016")
correlation_heatmap(combined_df_2017.pct_change(), "Cryptocurrency Correlations in 2017")
correlation_heatmap(combined_df_2018.pct_change(), "Cryptocurrency Correlations in 2018")
```


Cryptocurrency Correlations in 2016
BTC
XRP
XMR
STR
LTC
ETH
ETC
DASH
DASH
ETC
ETH
LTC
STR
XMR
XRP
BTC


Cryptocurrency Correlations in 2017
BTC
XRP
XMR
STR
LTC
ETH
ETC
DASH
DASH
ETC
ETH
LTC
STR
XMR
XRP
BTC


Cryptocurrency Correlations in 2018
BTC
XRP
XMR
STR
LTC
ETH
ETC
DASH
DASH
ETC
ETH
LTC
STR
XMR
XRP
BTC

This is incredible. If it wasn't immediately discernible from the price charts, the correlations of cryptocurrency price movement has drastically increased as each year moves forward. I personally hypothesize that this is because of the creation of many institutional investors who have ETFs (Exchange traded funds) that hold portfolios with equal weight in many cryptocurrencies. Now that the market is becoming even more mainstream, as Bitcoin, the "father of all cryptos" declines in value, the other altcoins also follow suit. Whether or not this is a good or bad thing will ultimately rest upon the trader themselves.

Well, this concludes my project. For a trader to use this toolkit, they can change the altcoins to whichever altcoins they are considering and run the code again. There are many ways to manipulate this project to suit each individual trader and I hope that the analysis that this project provides will help them.

Additionally, some interesting articles for further insight on cryptocurrency trading:

http://www.scmp.com/business/banking-finance/article/2146103/cryptocurrency-exchange-wins-new-york-state-approval-begin (http://www.scmp.com/business/banking-finance/article/2146103/cryptocurrency-exchange-wins-new-york-state-approval-begin)

https://www.cnbc.com/2018/05/04/coinbase-prepares-for-a-monster-increase-in-trading.html (https://www.cnbc.com/2018/05/04/coinbase-prepares-for-a-monster-increase-in-trading.html)