In [2]:
```
#import the required Dependencies
import os
import numpy as np
import pandas as pd
import pickle
import quandl
quandl.ApiConfig.api_key = "CwmbSAFxgKmsaVehH99K" #Insert my API key from Quandl.com\n"
from datetime import datetime
```

In [3]:
```
import plotly.offline as py
import plotly.graph_objs as go
import plotly.figure_factory as ff
py.init_notebook_mode(connected=True)
import plotly.plotly as py
```

In [4]:
```
#To assist with this data retrieval we'll define a function to download and cache datasets from Quandl.
def get_quandl_data(quandl_id):
    '''Download and cache Quandl dataseries'''
    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, 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 [5]:
```
# Pull Kraken BTC price exchange data
btc_usd_price_kraken = get_quandl_data('BCHARTS/KRAKENUSD')
```

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

In [6]:
```
# Inspect the first Five Rows of the data
btc_usd_price_kraken.tail()
```

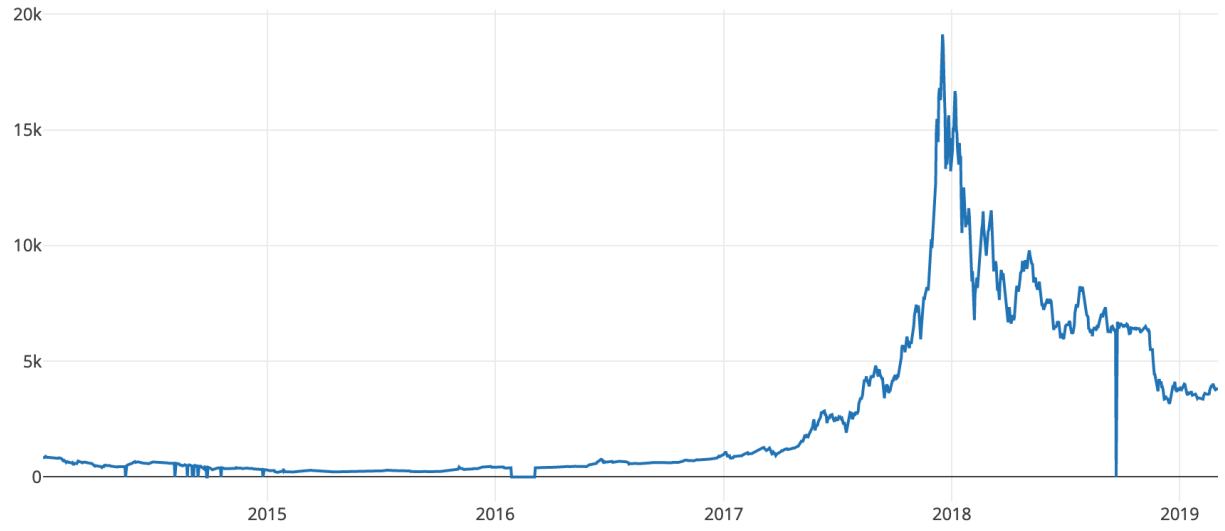
In [6]:
```
# Inspect the first Five Rows of the data
btc_usd_price_kraken.tail()
```

Out[6]:

	Open	High	Low	Close	Volume (BTC)	Volume (Currency)	Weighted Price
Date							
2019-02-27	3795.0	3827.9	3652.6	3799.1	6161.730826	2.322616e+07	3769.421906
2019-02-28	3797.8	3885.0	3756.4	3793.7	5628.741814	2.147717e+07	3815.625042
2019-03-01	3793.6	3843.0	3785.8	3804.7	2670.270560	1.017905e+07	3811.993063
2019-03-02	3804.6	3818.5	3754.3	3809.9	1766.652670	6.714834e+06	3800.879323
2019-03-03	3810.3	3822.4	3755.4	3785.7	2204.155819	8.362644e+06	3794.034743

In [7]:
```
# Chart the BTC pricing data
btc_trace = go.Scatter(x=btc_usd_price_kraken.index, y=btc_usd_price_kraken['Weighted Price'])
py.iplot([btc_trace])
```

Out[7]:



In [8]:
```
# Pull pricing data for 3 more BTC exchanges
exchanges = ['BITSTAMP','BITFLYER','OKCOIN']

exchange_data = {}

exchange_data['KRAKEN'] = btc_usd_price_kraken

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

In [8]:
```
# Pull pricing data for 3 more BTC exchanges
exchanges = ['BITSTAMP','BITFLYER','OKCOIN']

exchange_data = {}

exchange_data['KRAKEN'] = btc_usd_price_kraken

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

```
Loaded BCHARTS/BITSTAMPUSD from cache
Loaded BCHARTS/BITFLYERUSD from cache
Loaded BCHARTS/OKCOINUSD from cache
```

In [9]:
```
# Merge All Of The Pricing Data Into A Single Dataframe
def merge_dfs_on_column(dataframes, labels, col):
    '''Merge a single column of each dataframe into a new combined dataframe'''
    series_dict = {}
    for index in range(len(dataframes)):
        series_dict[labels[index]] = dataframes[index][col]

    return pd.DataFrame(series_dict)
```

In [10]:
```
# Merge the BTC price dataseries' into a single dataframe
btc_usd_datasets = merge_dfs_on_column(list(exchange_data.values()), list(exchange_data.keys()), 'Weighted Price')
```

In [11]:
```
btc_usd_datasets.tail()
```

Out[11]:

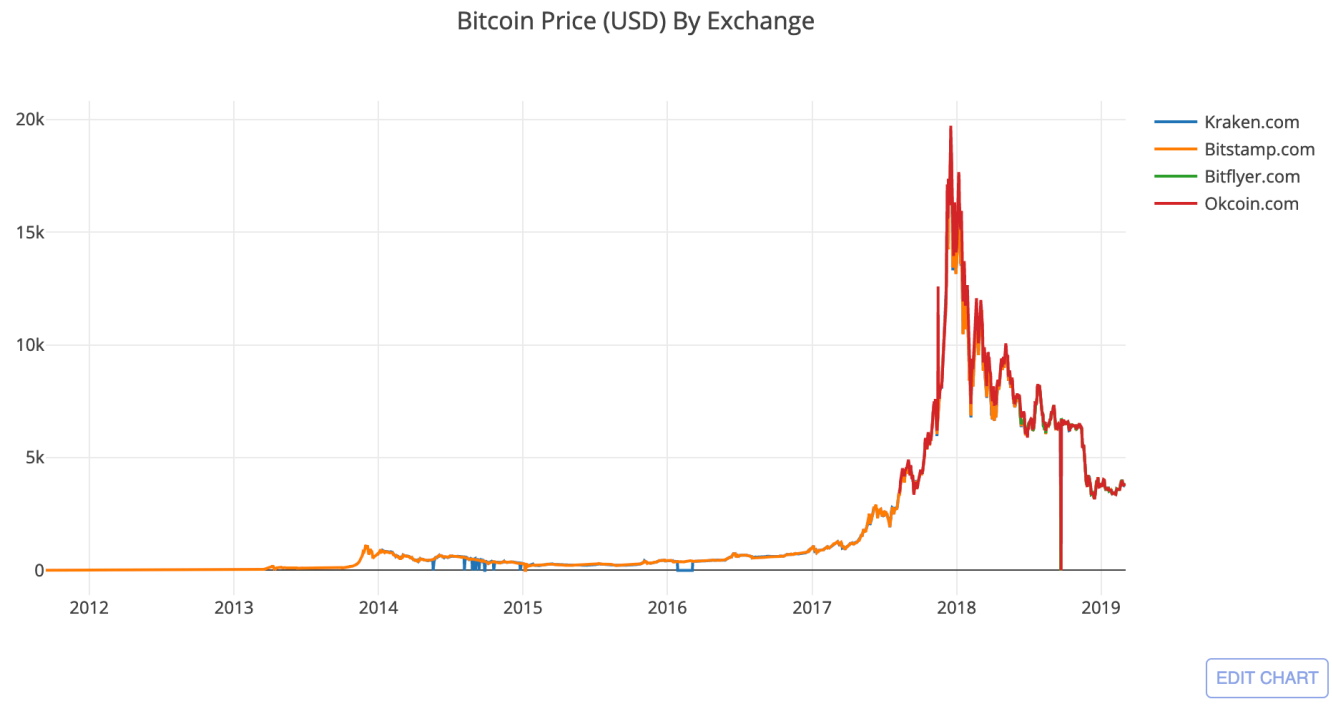
	KRAKEN	BITSTAMP	BITFLYER	OKCOIN
Date				
2019-02-28	3815.625042	3815.527051	3821.843137	3802.298307
2019-03-01	3811.993063	3816.277494	3793.724961	3813.955883
2019-03-02	3800.879323	3802.063926	3813.055542	3795.331813
2019-03-03	3794.034743	3797.084534	3800.659165	3795.386001
2019-03-04	NaN	3787.625497	3784.505000	3783.135526

In [12]:
```
#Visualize the pricing

Trace1 = go.Scatter(
            x = btc_usd_datasets.index,
            y = btc_usd_datasets.KRAKEN,
            mode = 'lines',
            name =  'Kraken.com'
    )

Trace2 = go.Scatter(
            x = btc_usd_datasets.index,
            y = btc_usd_datasets.BITSTAMP,
            mode = 'lines',
            name =  'Bitstamp.com'
    )

Trace3 = go.Scatter(
            x = btc_usd_datasets.index,
            y = btc_usd_datasets.BITFLYER,
            mode = 'lines',
            name =  'Bitflyer.com'
    )

Trace4 = go.Scatter(
            x = btc_usd_datasets.index,
            y = btc_usd_datasets.OKCOIN,
            mode = 'lines',
            name =  'Okcoin.com'
    )
```

In [13]:
```
data = [Trace1,Trace2,Trace3,Trace4]
layout = go.Layout(title = 'Bitcoin Price (USD) By Exchange')
figure = go.Figure(data = data, layout = layout)
```

In [14]:
```
py.iplot(figure)
```

Out[14]:

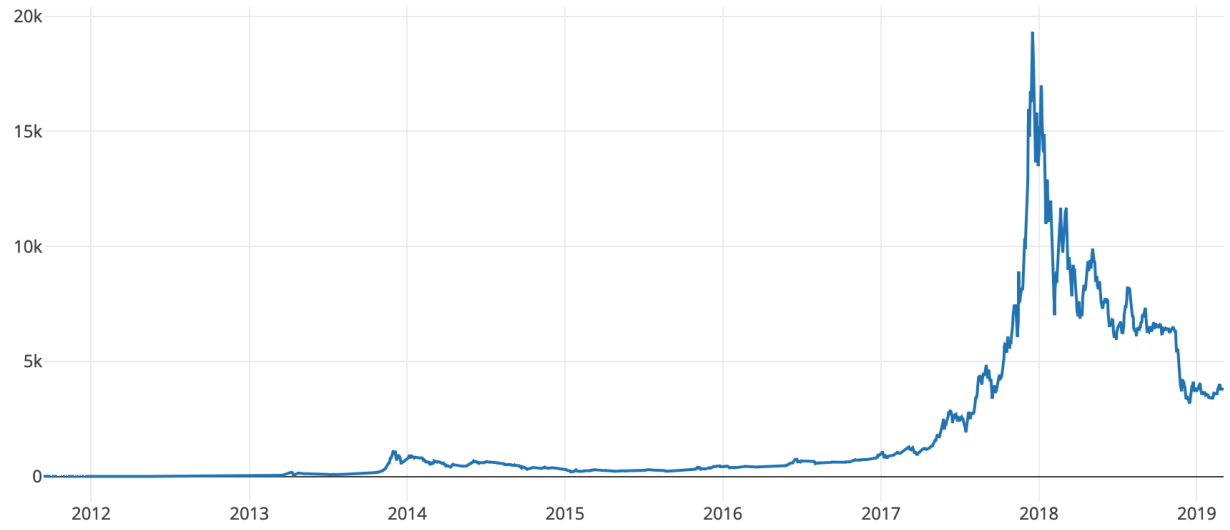


In [15]:
```
# Remove "0" values
btc_usd_datasets.replace(0, np.nan, inplace=True)
```

In [16]:
```
# Calculate the average BTC price as a new column
btc_usd_datasets['avg_btc_price_usd'] = btc_usd_datasets.mean(axis=1)
```

In [17]:
```
# Plot the average BTC price
btc_trace = go.Scatter(x=btc_usd_datasets.index, y=btc_usd_datasets['avg_btc_price_usd'])
py.iplot([btc_trace])
```

Out[17]:



EDIT CHART

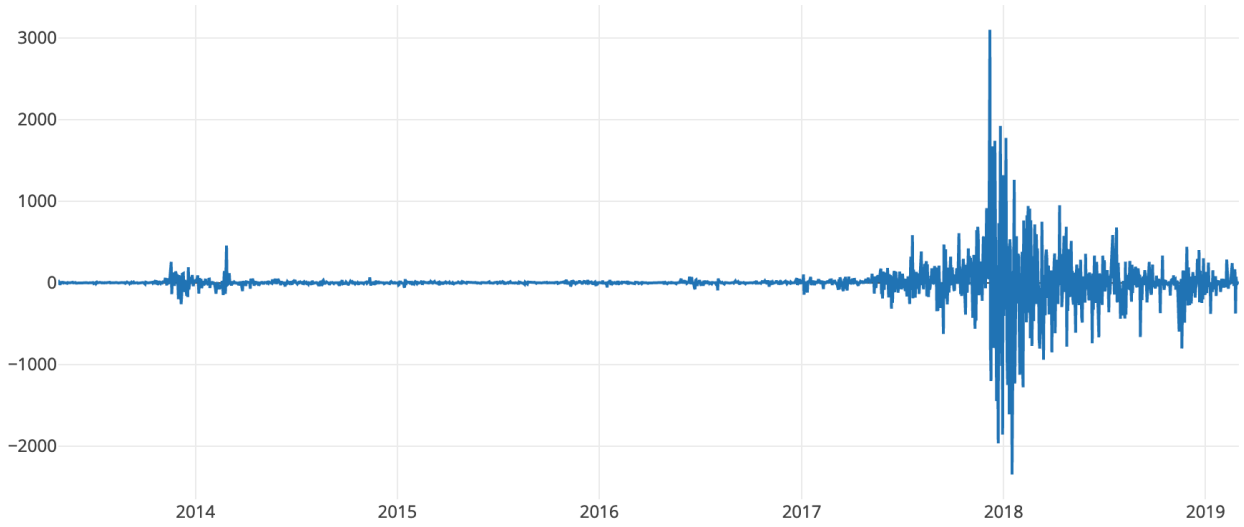
In [18]:
```
import pandas as pd
data1 = pd.read_csv("BTC.csv",parse_dates=["Date"], index_col="Date")
data1.head()
```

Out[18]:

	Open	High	Low	Close	Adj Close	Volume	DailyChange	DailyMargin
Date								
2019-03-02	3831.479980	3846.379883	3790.739990	3843.449951	3843.449951	73609812	11.969971	55.639893
2019-03-01	3823.370117	3866.120117	3822.669922	3831.479980	3831.479980	101774924	8.109863	43.450195
2019-02-28	3830.719971	3909.860107	3798.000000	3823.370117	3823.370117	170646670	-7.349854	111.860107
2019-02-27	3817.879883	3851.840088	3699.530029	3830.719971	3830.719971	175069054	12.840088	152.310059
2019-02-26	3845.510010	3852.659912	3791.899902	3817.879883	3817.879883	126868886	-27.630127	60.760010

In [19]:
```
# Plot the average BTC price
btc = go.Scatter(x=data1.index, y=data1['DailyChange'])
py.iplot([btc])
```

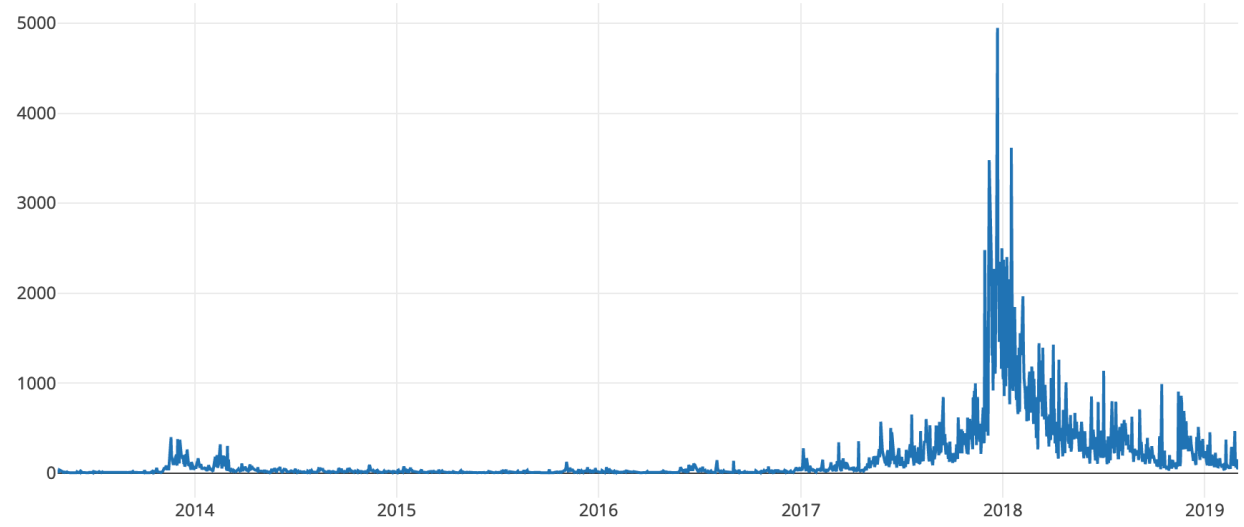
Out[19]:



EDIT CHART

In [20]:
```
# Plot the Daily Marginal Change of the BTC
btc = go.Scatter(x=data1.index, y=data1['DailyMargin'])
py.iplot([btc])
```

Out[20]:



EDIT CHART

In [21]:
```
import pandas as pd
data2 = pd.read_csv("ABCD.csv",parse_dates=["Date"], index_col="Date")
data2.tail()
```

Out[21]:

	Open	High	Low	Close	Adj Close	Volume	DailyChange	DailyMargin
Date								
2017-01-05	1135.410034	1150.630005	874.530029	989.349976	989.349976	244839289.0	-146.060058	276.099976
2017-01-04	1033.300049	1148.540039	1022.320007	1135.410034	1135.410034	170584623.0	102.109985	126.220032
2017-01-03	1017.049988	1035.469971	1006.530029	1033.300049	1033.300049	56085870.0	16.250061	28.939942
2017-01-02	995.440002	1031.680054	990.200012	1017.049988	1017.049988	66038073.0	21.609986	41.480042
2017-01-01	963.380005	1001.609985	956.099976	995.440002	995.440002	40570922.0	32.059997	45.510009

In [22]:
```
# Plot Candle Stick Plot of BTC to deptermine when its viable to trade or drop position
from plotly.tools import FigureFactory as FF
fig = FF.create_candlestick(data2.Open, data2.High, data2.Low, data2.Close, dates=data2.index)
py.iplot(fig, filename='Simple_Candlestick', validate=False)
```

Out[22]:

EDIT CHART

In [23]:
```
# Retrieve Altcoin Data from Poloniex.com
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 {} at {}'.format(json_url, cache_path))
    return df
```

In [24]:
```
base_polo_url = 'https://poloniex.com/public?command=returnChartData&currencyPair={}&start={}&end={}&period={}'
start_date = datetime.strptime('2012-01-01', '%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
```

In [25]:
```
# Download Trading Data from Poloniex
altcoins = ['ETH','LTC','XRP','ETC','STR','DASH','SC','XMR','XEM']

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=1325388600.0&end=1553207241.118
354&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_LTC&start=1325388600.0&end=1553207241.118
354&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_XRP&start=1325388600.0&end=1553207241.118
354&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_ETC&start=1325388600.0&end=1553207241.118
354&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_STR&start=1325388600.0&end=1553207241.118
354&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_DASH&start=1325388600.0&end=1553207241.11
8354&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_SC&start=1325388600.0&end=1553207241.1183
54&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_XMR&start=1325388600.0&end=1553207241.118
354&period=86400 from cache
Loaded https://poloniex.com/public?command=returnChartData&currencyPair=BTC_XEM&start=1325388600.0&end=1553207241.118
354&period=86400 from cache
```

In [26]:
```
# Show the tail of the Altcoin Data (Etherium)
altcoin_data['ETH'].tail()
```

Out[26]:

	close	high	low	open	quoteVolume	volume	weightedAverage
date							
2019-02-28	0.035465	0.035990	0.034940	0.035337	11088.199678	394.855178	0.035610
2019-03-01	0.035335	0.035791	0.035283	0.035491	6146.563879	218.159505	0.035493
2019-03-02	0.034665	0.035503	0.034354	0.035375	6077.638925	211.917755	0.034868
2019-03-03	0.034303	0.035148	0.033867	0.034670	5066.752151	175.420158	0.034622
2019-03-04	0.033839	0.034299	0.033397	0.034281	6506.653137	219.987417	0.033810

In [27]:
```
# Calculate USD Price as a new column in each altcoin dataframe
for altcoin in altcoin_data.keys():
    altcoin_data[altcoin]['price_usd'] =  altcoin_data[altcoin]['weightedAverage'] * btc_usd_datasets['avg_btc_price_u
sd']
```

In [28]:
```
# Merge USD price of each altcoin into single dataframe
combined_df = merge_dfs_on_column(list(altcoin_data.values()), list(altcoin_data.keys()), 'price_usd')
```

In [29]:
```
# Add BTC price to the dataframe
combined_df['BTC'] = btc_usd_datasets['avg_btc_price_usd']
```

In [30]:
```
# Show the tail of the Altcoin Data
combined_df.tail()
```

Out[30]:

	ETH	LTC	XRP	ETC	STR	DASH	SC	XMR	XEM	BTC
date										
2019-02-28	135.811738	45.572634	0.311361	4.233458	0.083599	81.251295	0.002365	48.045899	0.042257	3813.823384
2019-03-01	135.192101	47.150623	0.317022	4.214721	0.084141	82.470758	0.002323	47.763413	0.042127	3808.987850
2019-03-02	132.598804	48.041223	0.312783	4.275525	0.083510	81.779954	0.002358	47.800199	0.042135	3802.832651
2019-03-03	131.451780	47.825710	0.310350	4.227613	0.086681	80.473633	0.002354	48.365500	0.041955	3796.791111
2019-03-04	127.972372	46.590619	0.307273	4.141076	0.084105	79.076218	0.002347	47.659111	0.041144	3785.088675

In [31]:

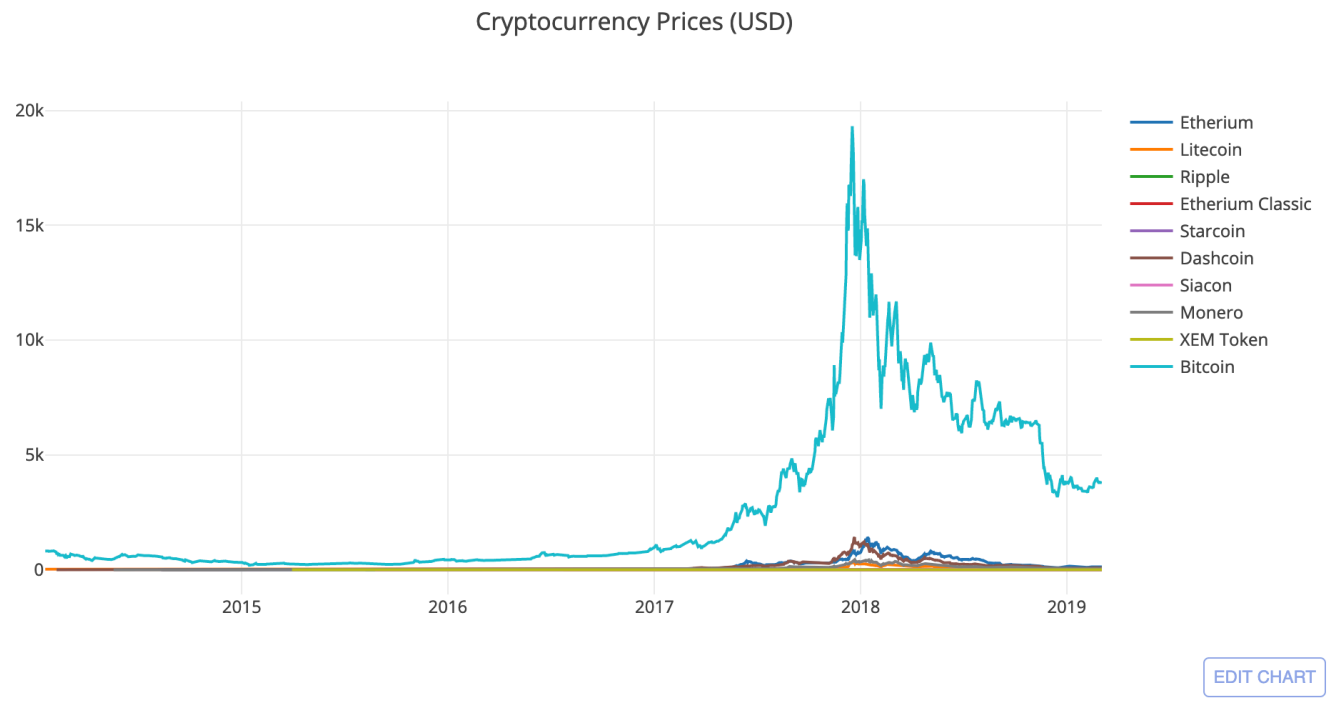
```
Trace1 = go.Scatter(
        x = combined_df.index,
        y = combined_df.ETH,
        mode = 'lines',
        name =  'Etherium'
)

Trace2 = go.Scatter(
        x = combined_df.index,
        y = combined_df.LTC,
        mode = 'lines',
        name =  'Litecoin'
)
Trace3 = go.Scatter(
        x = combined_df.index,
        y = combined_df.XRP,
        mode = 'lines',
        name =  'Ripple'
)

Trace4 = go.Scatter(
        x = combined_df.index,
        y = combined_df.ETC,
        mode = 'lines',
        name =  'Etherium Classic'
)
```

```
Trace6 = go.Scatter(
        x = combined_df.index,
        y = combined_df.DASH,
        mode = 'lines',
        name =  'Dashcoin'
)

Trace7 = go.Scatter(
        x = combined_df.index,
        y = combined_df.SC,
        mode = 'lines',
        name =  'Siacon'
)

Trace8 = go.Scatter(
        x = combined_df.index,
        y = combined_df.XMR,
        mode = 'lines',
        name =  'Monero'
)

Trace9 = go.Scatter(
        x = combined_df.index,
        y = combined_df.XEM,
        mode = 'lines',
        name =  'XEM Token'
)

Trace10 = go.Scatter(
        x = combined_df.index,
        y = combined_df.BTC,
        mode = 'lines',
        name =  'Bitcoin'
)
```

In [32]:
```
data = [Trace1,Trace2,Trace3,Trace4,Trace5,Trace6,Trace7,Trace8,Trace9,Trace10]
layout = go.Layout(title = 'Cryptocurrency Prices (USD)')
figure = go.Figure(data = data, layout = layout)
```

In [33]:
```
# Plot Cryptocurrency Prices in USD
py.iplot(figure)
```

Out[33]:



In [34]:
```
# Calculate the pearson correlation coefficients for cryptocurrencies in 2019
combined_df_2019 = combined_df[combined_df.index.year == 2019]
combined_df_2019.pct_change().corr(method='pearson')
```

Out[34]:

	ETH	LTC	XRP	ETC	STR	DASH	SC	XMR	XEM	BTC
ETH	1.000000	0.719607	0.684129	0.786109	0.700429	0.774323	0.705407	0.817768	0.563254	0.879684
LTC	0.719607	1.000000	0.580395	0.640176	0.642780	0.627805	0.623876	0.731166	0.455902	0.828460
XRP	0.684129	0.580395	1.000000	0.604901	0.737107	0.689768	0.530551	0.647041	0.421379	0.742589
ETC	0.786109	0.640176	0.604901	1.000000	0.676000	0.745485	0.692395	0.753519	0.592874	0.785987
STR	0.700429	0.642780	0.737107	0.676000	1.000000	0.659234	0.659242	0.720223	0.659349	0.813164
DASH	0.774323	0.627805	0.689768	0.745485	0.659234	1.000000	0.594047	0.817303	0.454368	0.813013
SC	0.705407	0.623876	0.530551	0.692395	0.659242	0.594047	1.000000	0.703334	0.519677	0.762007
XMR	0.817768	0.731166	0.647041	0.753519	0.720223	0.817303	0.703334	1.000000	0.528572	0.874938
XEM	0.563254	0.455902	0.421379	0.592874	0.659349	0.454368	0.519677	0.528572	1.000000	0.568753
BTC	0.879684	0.828460	0.742589	0.785987	0.813164	0.813013	0.762007	0.874938	0.568753	1.000000

In [35]:
```
# Calculate the pearson correlation coefficients for cryptocurrencies in 2018
combined_df_2018 = combined_df[combined_df.index.year == 2018]
combined_df_2018.pct_change().corr(method='pearson')
```

Out[35]:

	ETH	LTC	XRP	ETC	STR	DASH	SC	XMR	XEM	BTC
ETH	1.000000	0.812364	0.711815	0.784106	0.676943	0.797683	0.696509	0.815767	0.707075	0.815823
LTC	0.812364	1.000000	0.690836	0.714940	0.678577	0.798515	0.726552	0.800976	0.662511	0.844528
XRP	0.711815	0.690836	1.000000	0.630524	0.786955	0.675729	0.670637	0.683988	0.729476	0.701324
ETC	0.784106	0.714940	0.630524	1.000000	0.609767	0.712126	0.605670	0.703536	0.617403	0.723447
STR	0.676943	0.678577	0.786955	0.609767	1.000000	0.683934	0.690371	0.699055	0.746370	0.727613
DASH	0.797683	0.798515	0.675729	0.712126	0.683934	1.000000	0.700494	0.817633	0.683425	0.797425
SC	0.696509	0.726552	0.670637	0.605670	0.690371	0.700494	1.000000	0.705620	0.698641	0.745239
XMR	0.815767	0.800976	0.683988	0.703536	0.699055	0.817633	0.705620	1.000000	0.704067	0.856687
XEM	0.707075	0.662511	0.729476	0.617403	0.746370	0.683425	0.698641	0.704067	1.000000	0.709565
BTC	0.815823	0.844528	0.701324	0.723447	0.727613	0.797425	0.745239	0.856687	0.709565	1.000000

In [36]:
```
# Calculate the pearson correlation coefficients for cryptocurrencies in 2017
combined_df_2017 = combined_df[combined_df.index.year == 2017]
combined_df_2017.pct_change().corr(method='pearson')
```

Out[36]:

	ETH	LTC	XRP	ETC	STR	DASH	SC	XMR	XEM	BTC
ETH	1.000000	0.468134	0.232382	0.624741	0.285320	0.538365	0.406428	0.583097	0.426159	0.469922
LTC	0.468134	1.000000	0.336242	0.508436	0.327614	0.375360	0.368549	0.466687	0.404928	0.465590
XRP	0.232382	0.336242	1.000000	0.137089	0.519106	0.116197	0.263731	0.245257	0.282404	0.166660
ETC	0.624741	0.508436	0.137089	1.000000	0.235300	0.422781	0.331071	0.479225	0.349538	0.464807
STR	0.285320	0.327614	0.519106	0.235300	1.000000	0.211239	0.420419	0.348993	0.354596	0.268392
DASH	0.538365	0.375360	0.116197	0.422781	0.211239	1.000000	0.327548	0.529846	0.356093	0.376626
SC	0.406428	0.368549	0.263731	0.331071	0.420419	0.327548	1.000000	0.409682	0.354694	0.375218
XMR	0.583097	0.466687	0.245257	0.479225	0.348993	0.529846	0.409682	1.000000	0.364910	0.464736
XEM	0.426159	0.404928	0.282404	0.349538	0.354596	0.356093	0.354694	0.364910	1.000000	0.372143
BTC	0.469922	0.465590	0.166660	0.464807	0.268392	0.376626	0.375218	0.464736	0.372143	1.000000

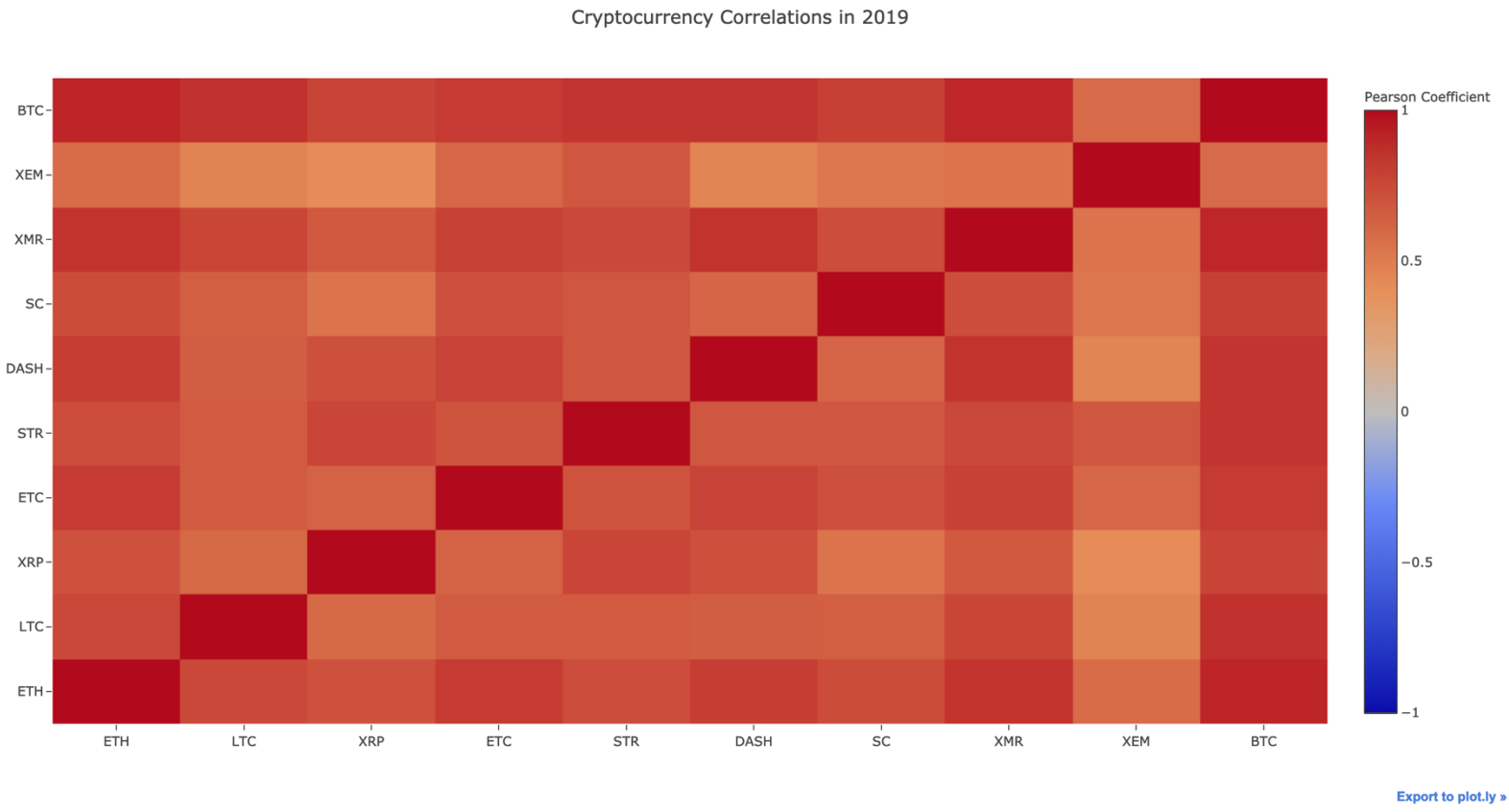
In [37]:
```
# Calculate the pearson correlation coefficients for cryptocurrencies in 2016
combined_df_2016 = combined_df[combined_df.index.year == 2016]
combined_df_2016.pct_change().corr(method='pearson')
```

Out[37]:

	ETH	LTC	XRP	ETC	STR	DASH	SC	XMR	XEM	BTC
ETH	1.000000	-0.057405	0.087995	-0.180951	0.038837	0.124465	0.168345	0.090507	0.045951	0.002103
LTC	-0.057405	1.000000	0.063289	-0.134475	0.124327	-0.002598	0.011237	0.139085	0.167943	0.756675
XRP	0.087995	0.063289	1.000000	-0.055493	0.324295	0.092328	0.020086	0.033056	0.105826	0.057021
ETC	-0.180951	-0.134475	-0.055493	1.000000	-0.103384	-0.001045	-0.009292	-0.103672	-0.081281	-0.169784
STR	0.038837	0.124327	0.324295	-0.103384	1.000000	0.064254	0.142769	0.033683	0.228720	0.093268
DASH	0.124465	-0.002598	0.092328	-0.001045	0.064254	1.000000	0.024682	0.126901	0.019492	-0.000552
SC	0.168345	0.011237	0.020086	-0.009292	0.142769	0.024682	1.000000	0.048264	0.106195	0.033859
XMR	0.090507	0.139085	0.033056	-0.103672	0.033683	0.126901	0.048264	1.000000	0.020735	0.139546
XEM	0.045951	0.167943	0.105826	-0.081281	0.228720	0.019492	0.106195	0.020735	1.000000	0.234964
BTC	0.002103	0.756675	0.057021	-0.169784	0.093268	-0.000552	0.033859	0.139546	0.234964	1.000000

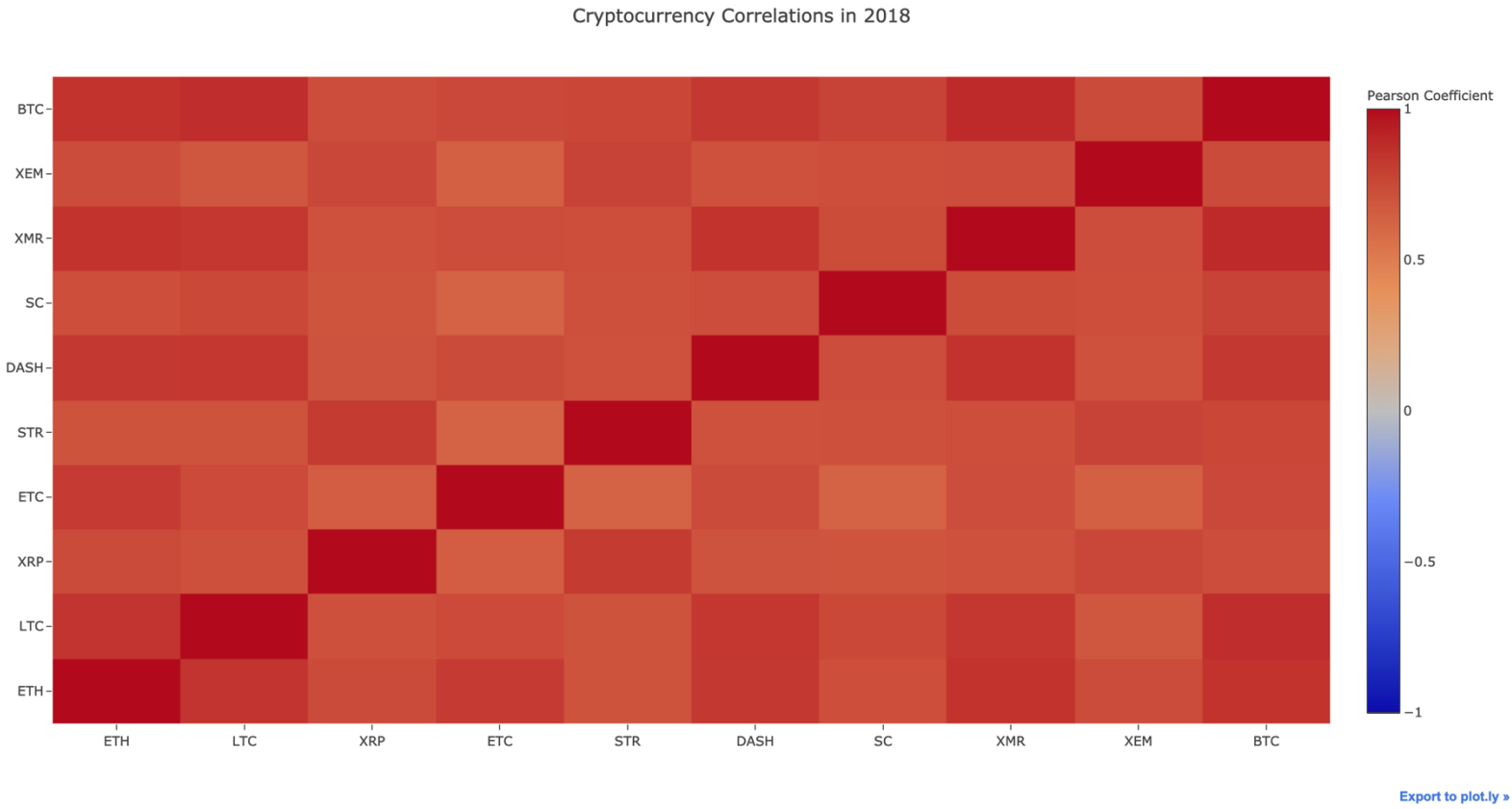
In [39]:
```
def correlation_heatmap(df, title, absolute_bounds=True):
    '''Plot a correlation heatmap for the entire dataframe'''
    heatmap = go.Heatmap(
        z=df.corr(method='pearson').as_matrix(),
        x=df.columns,
        y=df.columns,
        colorbar=dict(title='Pearson Coefficient'),
    )

    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 [40]:
```
correlation_heatmap(combined_df_2019.pct_change(), "Cryptocurrency Correlations in 2019")
```



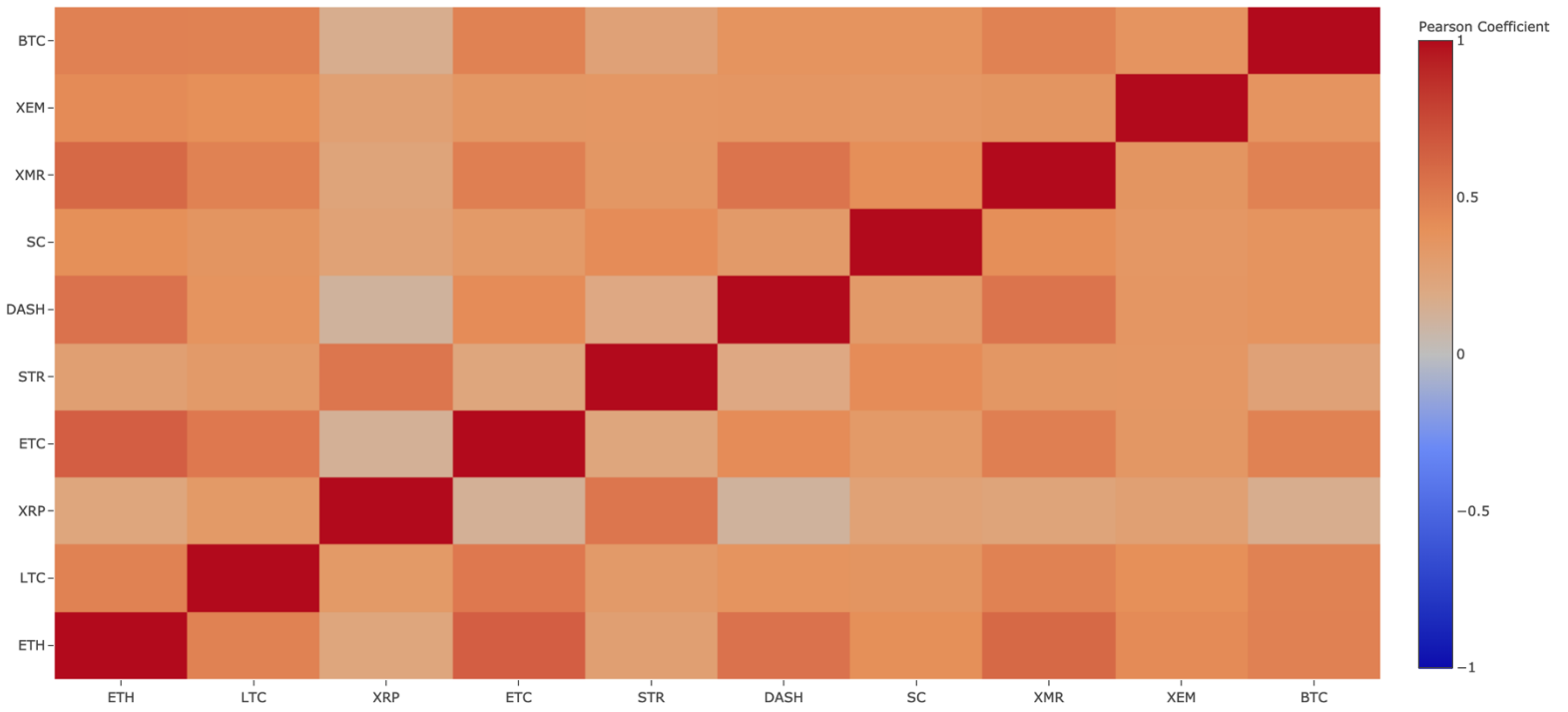
Export to plot.ly »

In [41]:
```
correlation_heatmap(combined_df_2018.pct_change(), "Cryptocurrency Correlations in 2018")
```



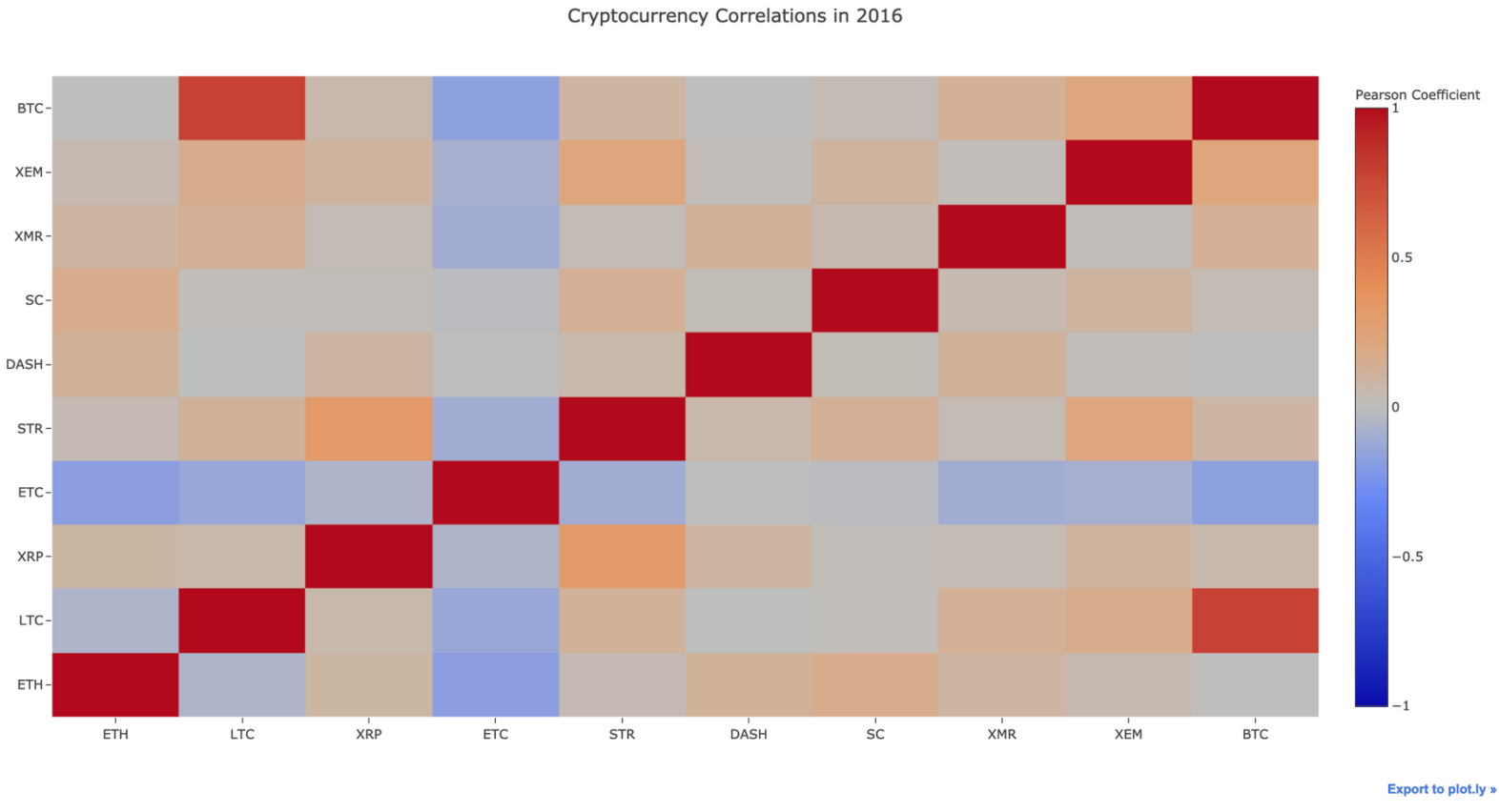
Export to plot.ly »

In [42]:
```
correlation_heatmap(combined_df_2017.pct_change(), "Cryptocurrency Correlations in 2017")
```



Export to plot.ly »

In [43]:
```
correlation_heatmap(combined_df_2016.pct_change(), "Cryptocurrency Correlations in 2016")
```



In [44]:
```
import quandl
import pandas as pd
import numpy as np
import datetime
from sklearn.linear_model import LinearRegression
from sklearn import datasets, svm, metrics
from sklearn import model_selection
from sklearn import preprocessing
from sklearn import linear_model
```

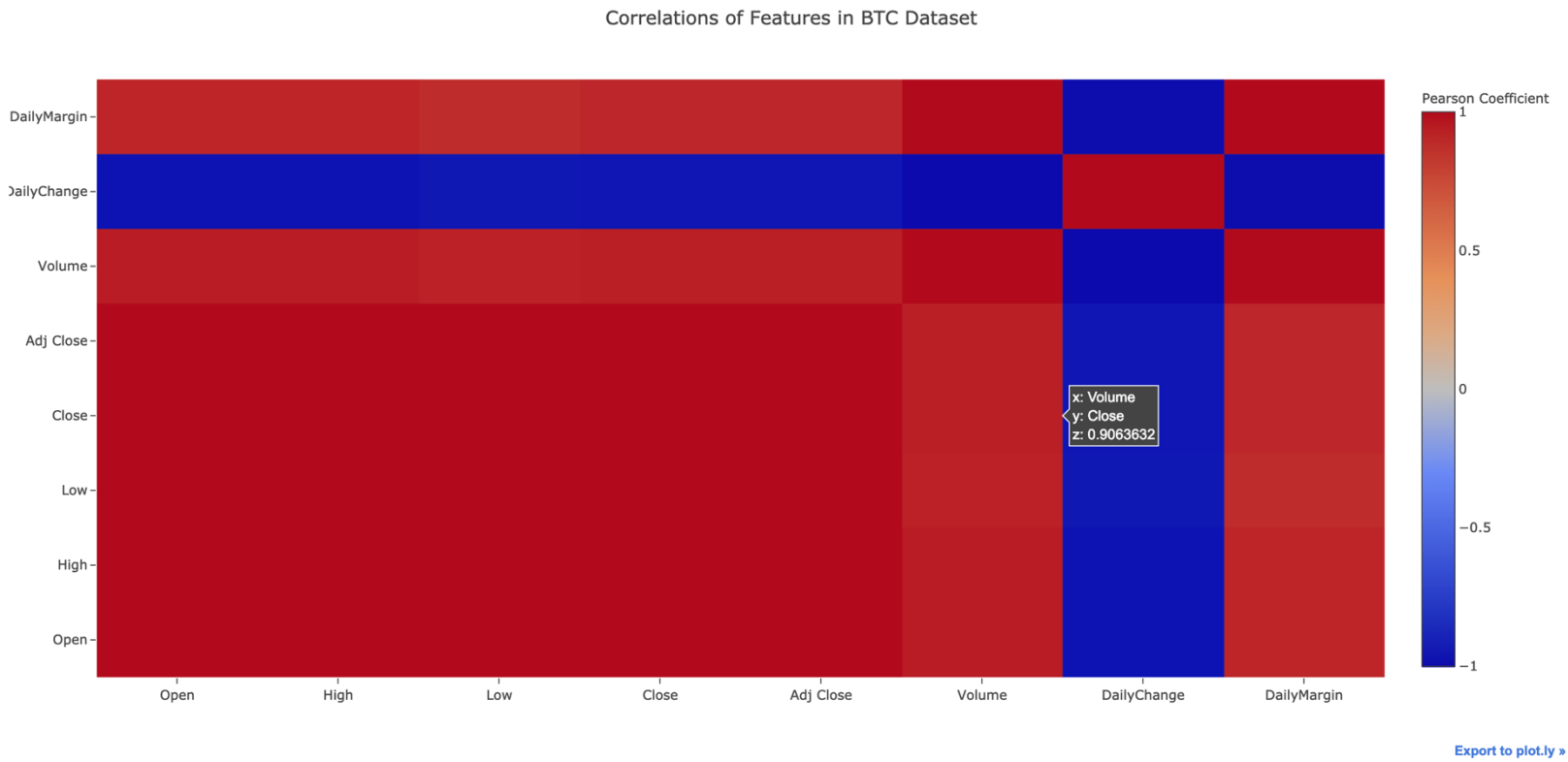
In [45]:
```
# Find the Correlation of the BTC Dataset Features
data1.corr()
```

Out[45]:

	Open	High	Low	Close	Adj Close	Volume	DailyChange	DailyMargin
Open	1.000000	0.998887	0.997833	0.997716	0.997716	0.845234	-0.034615	0.792933
High	0.998887	1.000000	0.997604	0.999103	0.999103	0.853711	0.002381	0.805316
Low	0.997833	0.997604	1.000000	0.998640	0.998640	0.820418	0.011100	0.762368
Close	0.997716	0.999103	0.998640	1.000000	1.000000	0.840308	0.032978	0.788044
Adj Close	0.997716	0.999103	0.998640	1.000000	1.000000	0.840308	0.032978	0.788044
Volume	0.845234	0.853711	0.820418	0.840308	0.840308	1.000000	-0.073421	0.955259
DailyChange	-0.034615	0.002381	0.011100	0.032978	0.032978	-0.073421	1.000000	-0.072838
DailyMargin	0.792933	0.805316	0.762368	0.788044	0.788044	0.955259	-0.072838	1.000000

In [46]:
```
df1=data1.corr()
```

In [47]:
```
correlation_heatmap(df1 , "Correlations of Features in BTC Dataset")
```



In [70]:
```
X = data1[['Volume','DailyMargin','DailyChange']]
```

In [71]:
```
y = data1[['Adj Close']]
```

In [72]:
```
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size = 0.30)
```

In [73]:
```
len(X_train)
```

Out[73]:
```
1494
```

In [74]:
```
len(X_test)
```

Out[74]:
```
641
```

In [75]:
```
# Training
clf = LinearRegression()
clf.fit(X_train,y_train)
```

Out[75]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [80]:
```
clf.coef_
```

Out[80]:
```
array([[ 7.35008366e-06, -1.84093238e+00,  2.05338874e+00]])
```

In [76]:
```
#clf.predict (X_test)
```

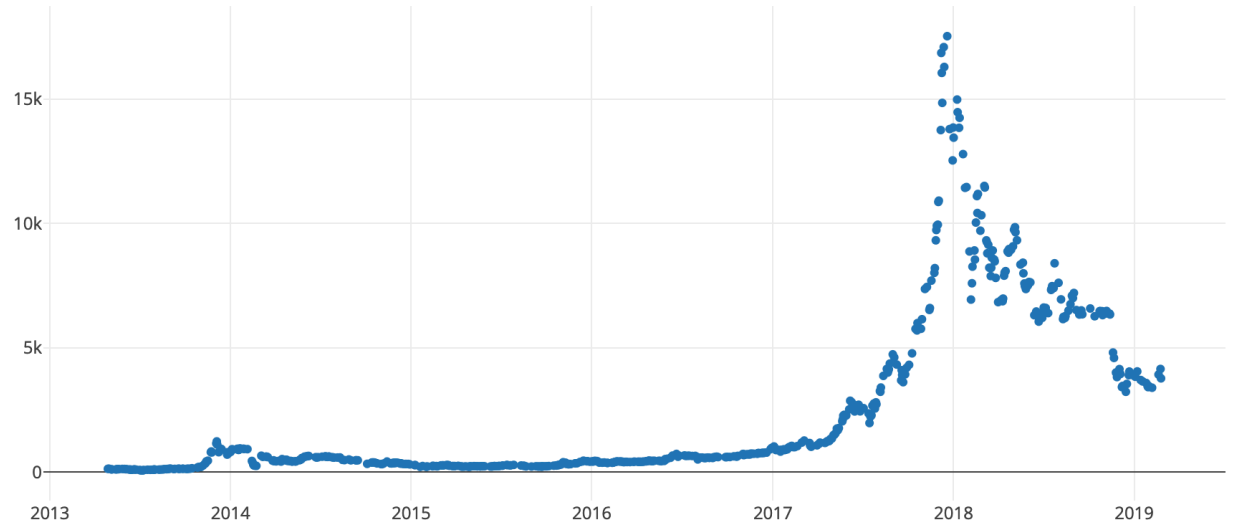
In [77]:
```
#y_test
```

In [78]:
```
# Confidence Value
confidence = clf.score(X_test, y_test)
print("Confidence Value: ", confidence * 100)
```

```
Confidence Value:  72.17195068212719
```

In [79]:
```
# Plot the average BTC price
btc = go.Scatter(x=y_test.index, y=y_test['Adj Close'], mode = 'markers')
py.iplot([btc])
```

Out[79]:



In [81]:
```
import pandas as pd
df2 = pd.read_csv("BTC_PRED.csv",parse_dates=["Date"], index_col="Date")
df2.tail()
```

Out[81]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2019-03-17	4027.010010	4030.379883	3970.969971	3998.000000	3998.000000	68146133
2019-03-18	3998.000000	4038.989990	3964.310059	3988.850098	3988.850098	109428478
2019-03-19	3988.850098	4032.040039	3972.469971	4024.139893	4024.139893	118919147
2019-03-20	4024.139893	4064.949951	3995.489990	4051.379883	4051.379883	147378040
2019-03-21	4056.770020	4082.290039	3953.530029	3992.449951	3992.449951	162040496

In [82]:
```
fig = FF.create_candlestick(df2.Open, df2.High, df2.Low, df2.Close, dates=df2.index)
py.iplot(fig, filename='Simple_Candlestick_Pred', validate=False)
```

In [59]:
```
atter(x=y_test.index, y=y_test['Adj Close'], mode = 'markers', name = 'Prdeicted Value of Trained Model')
atter(x=btc_usd_datasets.index, y=btc_usd_datasets['avg_btc_price_usd'], mode = 'lines', name = 'Real Value of Bitcoin'
```

In [60]:
```
data3 = [Trace1,Trace2]
layout = go.Layout(title = 'Predicted Values VS Real Values of Bitcoin')
figure = go.Figure(data = data3, layout = layout)
```

In [61]:
```
# Plot Predicted Values Vs Real Values
py.iplot(figure)
```

Out[61]:

In [74]:
```
X = data1[['Close','Open','Close','High']]
```

In [75]:
```
y = data1[['Adj Close']]
```

In [76]:
```
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size = 0.30)
```

In [77]:
```
# Training
clf = LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
clf.fit(X_train,y_train)
```

Out[77]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [78]:
```
clf.coef_
```

Out[78]:
```
array([[ 5.00000000e-01, -1.10824134e-16,  5.00000000e-01,
         1.41632891e-16]])
```

In [79]:
```
# Confidence Value
confidence = clf.score(X_test, y_test)
print("Confidence Value: ", confidence * 100)
```

```
Confidence Value:  100.0
```

In [118]:
```
X = data1[['DailyChange']]
```

In [119]:
```
y = data1[['Adj Close']]
```

In [120]:
```
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size = 0.30)
```

In [121]:
```
# Training
clf = LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
clf.fit(X_train,y_train)
```

Out[121]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [122]:
```
clf.coef_
```

Out[122]:
```
array([[0.67756712]])
```

In [132]:
```
clf.coef_
```

Out[132]:
```
array([[6.00756159e-06]])
```

In [133]:
```
# Confidence Value
confidence = clf.score(X_test, y_test)
print("Confidence Value: ", confidence * 100)
```

```
Confidence Value:  70.30092501822584
```