Cryptocurrencies Pricing Analysis

RIHAD VARIAWA

30-10-2018

In [0]:
```
!pip install quandl
!pip install plotly
```

```
Collecting quandl
  Downloading https://files.pythonhosted.org/packages/6f/b9/237394fb7f23fd0233bc5c39a2f1c74d4c24850932762ff51be5be20269c/Quandl-3.4.4-py2.py3-none-any.whl
Requirement already satisfied: requests>=2.7.0 in /usr/local/lib/python3.6/dist-packages (from quandl) (2.18.4)
Collecting inflection>=0.3.1 (from quandl)
  Downloading https://files.pythonhosted.org/packages/d5/35/a6eb45b4e2356fe688b21570864d4aa0d0a880ce387defe9c589112077f8/inflection-0.3.1.tar.gz
Collecting more-itertools (from quandl)
  Downloading https://files.pythonhosted.org/packages/79/b1/eace304ef66bd7d3d8b2f78cc374b73ca03bc53664d78151e9df3b3996cc/more_itertools-4.3.0-py3-none-any.whl (48kB)
    100% |████████████████████████████████| 51kB 4.3MB/s
Requirement already satisfied: numpy>=1.8 in /usr/local/lib/python3.6/dist-packages (from quandl) (1.14.6)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-packages (from quandl) (2.5.3)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from quandl) (1.11.0)
Requirement already satisfied: pandas>=0.14 in /usr/local/lib/python3.6/dist-packages (from quandl) (0.22.0)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests>=2.7.0->quandl) (3.0.4)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests>=2.7.0->quandl) (1.22)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests>=2.7.0->quandl) (2018.10.15)
Requirement already satisfied: idna<2.7,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests>=2.7.0->quandl) (2.6)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas>=0.14->quandl) (2018.6)
Building wheels for collected packages: inflection
  Running setup.py bdist_wheel for inflection ... - done
  Stored in directory: /root/.cache/pip/wheels/9f/5a/d3/6fc3bf6516d2a3eb7e18f9f28b472110b59325f3f258fe9211
Successfully built inflection
Installing collected packages: inflection, more-itertools, quandl
Successfully installed inflection-0.3.1 more-itertools-4.3.0 quandl-3.4.4
Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages (1.12.12)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from plotly) (2.18.4)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from plotly) (1.11.0)
Requirement already satisfied: pytz in /usr/local/lib/python3.6/dist-packages (from plotly) (2018.6)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests->plotly) (1.22)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests->plotly) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests->plotly) (2018.10.15)
Requirement already satisfied: idna<2.7,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests->plotly) (2.6)
```

In [0]:
```python
import os
import numpy as np
import pandas as pd
import pickle
import quandl
from datetime import datetime
```

In [0]:
```python
#Enable offline mode
import plotly.offline as py
import plotly.graph_objs as go
from plotly.tools import FigureFactory as FF
py.init_notebook_mode(connected=True)
```

In [0]:
```python
#Let's get bitcoin pricing data using Quandl's bitcoin API
def get_quandl_data(quandl_id):
  cashe_path = '{}.pkl'.format(quandl_id).replace('/','-')
  try:
    f = open(cashe_path, 'rb')
    df = pickle.load(f)
    print('Loaded {} from cashe'.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(cashe_path)
    print('Cached {} as {}'.format(quandl_id, cashe_path))
  return df
```

In [0]:
```python
#Let's get pricing data from Kraken bitcoin exchange
btc_usd_price_kraken = get_quandl_data('BCHARTS/KRAKENUSD')
```

```
Downloading BCHARTS/KRAKENUSD from Quandl
Cached BCHARTS/KRAKENUSD as BCHARTS-KRAKENUSD.pkl
```

In [0]:
```python
#Let's inspect the first 5 rows
btc_usd_price_kraken.head()
```

Out[0]:

Date	Open	High	Low	Close	Volume (BTC)	Volume (Currency)	Weighted Price
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

In [0]:
```python
#Let's visualise the data
btc_trace = go.Scatter(x=btc_usd_price_kraken.index, y=btc_usd_price_kraken['Weighted Price'])
py.iplot([btc_trace])
```

Export to plot.ly »

In [0]:
```python
#Let's extract more data from 3 more exchanges to calc an aggregate bitcoin price index
exchanges = ['COINBASE', 'BITSTAMP', 'ITBIT']
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
```

```
Downloading BCHARTS/COINBASEUSD from Quandl
Cached BCHARTS/COINBASEUSD as BCHARTS-COINBASEUSD.pkl
Downloading BCHARTS/BITSTAMPUSD from Quandl
Cached BCHARTS/BITSTAMPUSD as BCHARTS-BITSTAMPUSD.pkl
Downloading BCHARTS/ITBITUSD from Quandl
Cached BCHARTS/ITBITUSD as BCHARTS-ITBITUSD.pkl
```

In [0]:
```python
#Let's merge all pricing data into a single df
def merge_dfs_on_column(dataframes, labels, col):
  series_dict = {}
  for index in range(len(dataframes)):
    series_dict[labels[index]] = dataframes[index][col]

  return pd.DataFrame(series_dict)
```

In [0]:
```python
#Let's merge all df on their 'Weighted Price' col
btc_usd_datasets = merge_dfs_on_column(list(exchange_data.values()), list(exchange_data.keys()), 'Weighted Price')
```

In [0]:
```python
#Let's preview last 5 rows
btc_usd_datasets.tail()
```

Out[0]:

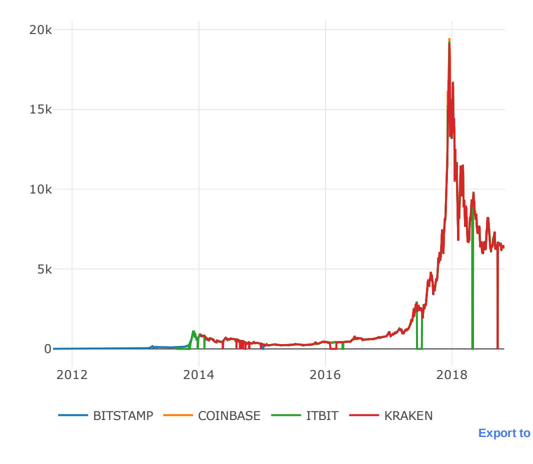
Date	BITSTAMP	COINBASE	ITBIT	KRAKEN
2018-10-25	6397.527049	6400.245236	6399.338522	6398.123113
2018-10-26	6409.240536	6403.984360	6398.979986	6404.132246
2018-10-27	6404.827196	6405.738944	6400.116561	6405.367996
2018-10-28	6402.355831	6401.989052	6402.783307	6410.208619
2018-10-29	6298.770984	6300.846994	6281.930929	6306.690844

Prices seem to be as expected. Similiar across all ranges, but with slight variations based on the demand and supply of each individual bitcoin exchange.

In [0]:
```python
#Let's visualise how these pricing datasets compare
def df_scatter(df, title, seperate_y_axis=False, y_axis_label='', scale='linear', initial_hide=False):

    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'

#Form trace for each series
    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)
```

In [0]:
```python
#Let's preview our btc exchange prices
df_scatter(btc_usd_datasets, 'Bitcoin Price (USD) By Exchange')
```

Bitcoin Price (USD) By Exchange



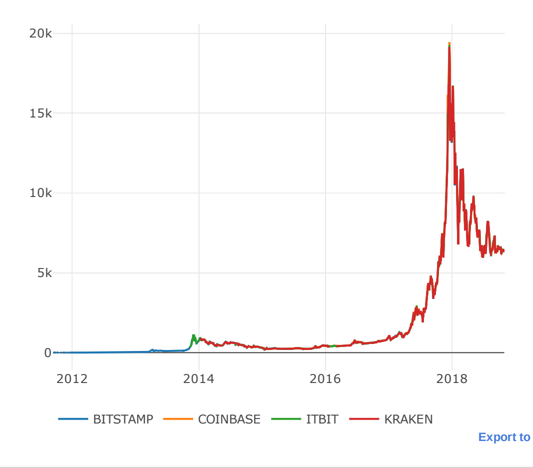
Export to plot.ly »

We notice that, although the four series follow roughly the same path, there are various irregulaties in each that we'll want to get rid of.

In [0]:
```python
#Let's remove all zero values from the df, since btc pricing has never been equal to zero in the timeframe we are reviewing
btc_usd_datasets.replace(0, np.nan, inplace=True)
```

In [0]:
```python
#Let's re-chart the df to see a cleaner chart without down-spikes
df_scatter(btc_usd_datasets, 'Bitcoin Price (USD) By Exchange')
```

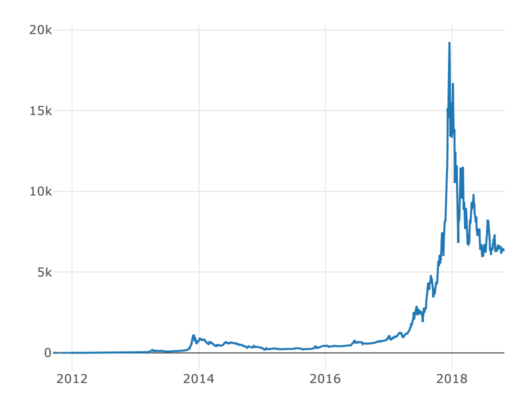
Bitcoin Price (USD) By Exchange



Export to plot.ly »

In [0]:
```python
#Let's calc a new col containing the average btc price across all exchanges
btc_usd_datasets['avg_btc_price_usd'] = btc_usd_datasets.mean(axis=1)
```

In [0]:
```python
#Let's chart this col to make sure it previews ok
btc_trace = go.Scatter(x=btc_usd_datasets.index, y=btc_usd_datasets['avg_btc_price_usd'])
py.iplot([btc_trace])
```



Export to plot.ly »

In [0]:

Great!

In [0]:
```python
#Let's retrieve data on altcoins, (non-btc currencies) using 2 helper functions to download and cashe JSON data from this API
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 [0]:
```python
#Let's define a function that will generate Poloniex API HTTP requests and subsequently call our new function to save the resulting data
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
```

This function will take a cryptocurrency pair string (such as 'BTC_ETH') and return a dataframe containing the historical exchange rate of the two currencies.

Most altcoins cannot be bought directly with USD; to acquire these coins individuals often buy Bitcoins and then trade the Bitcoins for altcoins on cryptocurrency exchanges.

Hence, we'll download the exchange rate to BTC for each coin, and then we'll use our existing BTC pricing data to convert this value to USD.

We'll download exchange data for nine of the top cryptocurrencies.

Ethereum, Litecoin, Ripple, Ethereum Classic, Stellar, Dash, Siacoin, Monero, and NEM.

In [0]:
```python
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
```

```
Downloading https://poloniex.com/public?command=returnChartData&currencyPair=BTC_ETH&start=1420070400.0&end=1540889085.14896&period=86400
Cached https://poloniex.com/public?command=returnChartData&currencyPair=BTC_ETH&start=1420070400.0&end=1540889085.14896&period=86400 at BTC_ETH
Downloading https://poloniex.com/public?command=returnChartData&currencyPair=BTC_LTC&start=1420070400.0&end=1540889085.14896&period=86400
Cached https://poloniex.com/public?command=returnChartData&currencyPair=BTC_LTC&start=1420070400.0&end=1540889085.14896&period=86400 at BTC_LTC
Downloading https://poloniex.com/public?command=returnChartData&currencyPair=BTC_XRP&start=1420070400.0&end=1540889085.14896&period=86400
Cached https://poloniex.com/public?command=returnChartData&currencyPair=BTC_XRP&start=1420070400.0&end=1540889085.14896&period=86400 at BTC_XRP
Downloading https://poloniex.com/public?command=returnChartData&currencyPair=BTC_ETC&start=1420070400.0&end=1540889085.14896&period=86400
Cached https://poloniex.com/public?command=returnChartData&currencyPair=BTC_ETC&start=1420070400.0&end=1540889085.14896&period=86400 at BTC_ETC
Downloading https://poloniex.com/public?command=returnChartData&currencyPair=BTC_STR&start=1420070400.0&end=1540889085.14896&period=86400
Cached https://poloniex.com/public?command=returnChartData&currencyPair=BTC_STR&start=1420070400.0&end=1540889085.14896&period=86400 at BTC_STR
Downloading https://poloniex.com/public?command=returnChartData&currencyPair=BTC_DASH&start=1420070400.0&end=1540889085.14896&period=86400
Cached https://poloniex.com/public?command=returnChartData&currencyPair=BTC_DASH&start=1420070400.0&end=1540889085.14896&period=86400 at BTC_DASH
Downloading https://poloniex.com/public?command=returnChartData&currencyPair=BTC_SC&start=1420070400.0&end=1540889085.14896&period=86400
Cached https://poloniex.com/public?command=returnChartData&currencyPair=BTC_SC&start=1420070400.0&end=1540889085.14896&period=86400 at BTC_SC
Downloading https://poloniex.com/public?command=returnChartData&currencyPair=BTC_XMR&start=1420070400.0&end=1540889085.14896&period=86400
Cached https://poloniex.com/public?command=returnChartData&currencyPair=BTC_XMR&start=1420070400.0&end=1540889085.14896&period=86400 at BTC_XMR
Downloading https://poloniex.com/public?command=returnChartData&currencyPair=BTC_XEM&start=1420070400.0&end=1540889085.14896&period=86400
Cached https://poloniex.com/public?command=returnChartData&currencyPair=BTC_XEM&start=1420070400.0&end=1540889085.14896&period=86400 at BTC_XEM
```

Now we have a dictionary with 9 dataframes, each containing the historical daily average exchange prices between the altcoin and Bitcoin.

In [0]:
```python
#Let's preview the last 5 rows for the Ethereum price table
altcoin_data['ETH'].tail()
```

Out[0]:

date	close	high	low	open	quoteVolume	volume	weightedAverage
2018-10-26	0.031345	0.031675	0.031145	0.031225	5385.568022	168.798551	0.031343
2018-10-27	0.031434	0.031450	0.031303	0.031345	3191.663487	100.133646	0.031373
2018-10-28	0.031630	0.031651	0.031391	0.031434	3145.438601	99.212066	0.031542
2018-10-29	0.031030	0.031811	0.030867	0.031629	8618.249836	269.317134	0.031250
2018-10-30	0.031060	0.031096	0.030980	0.031020	1249.165185	38.757839	0.031027

In [0]:
```python
#Let's combine this btc altcoin exchange prices with our btc pricing index to directly calc historical USD values for each altcoin
for altcoin in altcoin_data.keys():
    altcoin_data[altcoin]['price_usd'] =  altcoin_data[altcoin]['weightedAverage'] * btc_usd_datasets['avg_btc_price_usd']
```

Here, we've created a new column in each altcoin dataframe with the USD prices for that coin.

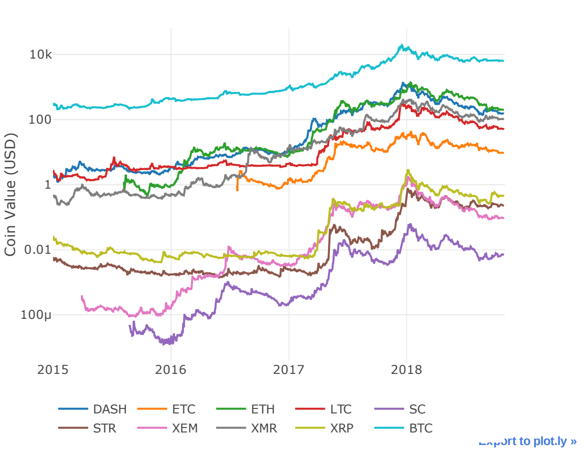
In [0]:
```python
#Let's create a combined df of USD prices for each cryptocurrency
combined_df = merge_dfs_on_column(list(altcoin_data.values()), list(altcoin_data.keys()), 'price_usd')
```

In [0]:
```python
#Let's add btc prices as a final col to the combined df
combined_df['BTC'] = btc_usd_datasets['avg_btc_price_usd']
```

Now we should have a single dataframe containing daily USD prices for the ten cryptocurrencies that we're examining.

In [0]:
```python
#Let's chart all currencies against each other
df_scatter(combined_df, 'Cryptocurrency Prices (USD)', seperate_y_axis=False, y_axis_label='Coin Value (USD)', scale='log')
```

Cryptocurrency Prices (USD)



Export to plot.ly »

FANTASTIC!

This graph provides a pretty solid "big picture" of how the exchange rates for each currency have varied over the past few years.

In [0]:
```python
#Let's perform Correlation Analysis, since cryptocurrency exchange prices, despite their different values and volatility, look slightly correlated
#Testing my correlation hypothesis using Pandas, computes a Pearson correlation coefficient for each col in the df against each other col
#Note, computing correlations directly on non-stationary time series can give bias correlation values
combined_df_2016 = combined_df[combined_df.index.year == 2016]
combined_df_2016.pct_change().corr(method='pearson')
```

Out[0]:

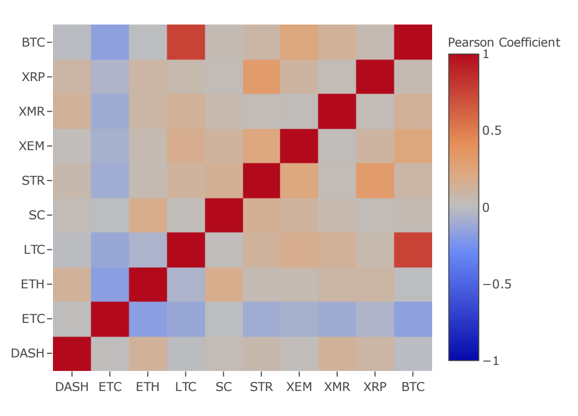
	DASH	ETC	ETH	LTC	SC	STR	XEM	XMR	
DASH	1.000000	0.003992	0.122695	-0.012194	0.026602	0.058083	0.014571	0.121537	0.0
ETC	0.003992	1.000000	-0.181991	-0.131079	-0.008066	-0.102654	-0.080938	-0.105898	-0.0
ETH	0.122695	-0.181991	1.000000	-0.064652	0.169642	0.035093	0.043205	0.087216	0.0
LTC	-0.012194	-0.131079	-0.064652	1.000000	0.012253	0.113523	0.160667	0.129475	0.0
SC	0.026602	-0.008066	0.169642	0.012253	1.000000	0.143252	0.106153	0.047910	0.0
STR	0.058083	-0.102654	0.035093	0.113523	0.143252	1.000000	0.225132	0.027998	0.3
XEM	0.014571	-0.080938	0.043205	0.160667	0.106153	0.225132	1.000000	0.016438	0.1
XMR	0.121537	-0.105898	0.087216	0.129475	0.047910	0.027998	0.016438	1.000000	0.0
XRP	0.088657	-0.054095	0.085630	0.053712	0.021098	0.320116	0.101326	0.027649	1.0
BTC	-0.014040	-0.170538	-0.006502	0.750174	0.035116	0.079075	0.227674	0.127520	0.0

These correlation coefficients are all over the place. Coefficients close to 1 or -1 mean that the series' are strongly correlated or inversely correlated respectively, and coefficients close to zero mean that the values are not correlated, and fluctuate independently of each other.

In [0]:
```python
#Let's visualise these results
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 [0]:
```python
correlation_heatmap(combined_df_2016.pct_change(), "Cryptocurrency Correlations in 2016")
```

Cryptocurrency Correlations in 2016



Export to plot.ly »

Here, the dark red values represent strong correlations (note that each currency is, obviously, strongly correlated with itself), and the dark blue values represent strong inverse correlations. All of the light blue/orange/gray/tan colors in-between represent varying degrees of weak/non-existent correlations.

Essentially, it shows that there was little statistically significant linkage between how the prices of different cryptocurrencies fluctuated during 2016.

In [0]:
```python
#Let's test the hypothesis thst cryptos have become more correlated in recent months
combined_df_2017 = combined_df[combined_df.index.year == 2017]
combined_df_2017.pct_change().corr(method='pearson')
```

Out[0]:

	DASH	ETC	ETH	LTC	SC	STR	XEM	XMR	XRP
DASH	1.000000	0.387555	0.506911	0.340153	0.291424	0.183038	0.325968	0.498418	0.091146
ETC	0.387555	1.000000	0.601437	0.482062	0.298406	0.210387	0.321852	0.447398	0.114780
ETH	0.506911	0.601437	1.000000	0.437609	0.373078	0.259399	0.399200	0.554632	0.212350
LTC	0.340153	0.482062	0.437609	1.000000	0.339144	0.307589	0.379088	0.437204	0.323905
SC	0.291424	0.298406	0.373078	0.339144	1.000000	0.402966	0.331350	0.378644	0.243872
STR	0.183038	0.210387	0.259399	0.307589	0.402966	1.000000	0.339502	0.327488	0.509828
XEM	0.325968	0.321852	0.399200	0.379088	0.331350	0.339502	1.000000	0.336076	0.268168
XMR	0.498418	0.447398	0.554632	0.437204	0.378644	0.327488	0.336076	1.000000	0.226636
XRP	0.091146	0.114780	0.212350	0.323905	0.243872	0.509828	0.268168	0.226636	1.000000
BTC	0.307095	0.416562	0.410771	0.420645	0.325318	0.230957	0.329431	0.409183	0.131469

These are somewhat more significant correlation coefficients. Strong enough to use as the sole basis for an investment? Certainly not.

In [0]:
```python
correlation_heatmap(combined_df_2017.pct_change(), "Cryptocurrency Correlations in 2017")
```

Cryptocurrency Correlations in 2017