Cryptocurrency 2018 Bear Market Analysis

- Author: Vanessa Liu (vanessaliu124@gmail.com)
- Date: Feb 14, 2018

Project Setup

In [61]:
```
#import the required dependencies
import requests
import datetime
import pandas as pd
import pickle
import matplotlib.pyplot as plt
import plotly.offline as py
import plotly.graph_objs as go
import plotly.figure_factory as ff
py.init_notebook_mode(connected=True)

%matplotlib inline
plt.style.use('fivethirtyeight')

# import Plotly and enable the offline mode.
import plotly.offline as py
import plotly.graph_objs as go
import plotly.figure_factory as ff
py.init_notebook_mode(connected=True)
```

In [58]:
```
# Pretty print the JSON
import uuid
from IPython.display import display_javascript, display_html, display
import json

class RenderJSON(object):
    def __init__(self, json_data):
        if isinstance(json_data, dict):
            self.json_str = json.dumps(json_data)
        else:
            self.json_str = json_data
        self.uuid = str(uuid.uuid4())

    def _ipython_display_(self):
        display_html('<div id="{}" style="height: 600px; width:100%;"></div
>'.format(self.uuid), raw=True)
        display_javascript("""
        require(["https://rawgit.com/caldwell/renderjson/master/renderjson.
js"], function() {
        document.getElementById('%s').appendChild(renderjson(%s))
        });
        """ % (self.uuid, self.json_str), raw=True)
```

In [81]:
```
# daily_price_historical function gives all the historical daily price of a cr
yptocurrency
# this price is updated every day at 8pm
# (price in USD in this study )
def daily_price_historical(symbol, comparison_symbol, limit=1, aggregate=1, ex
change='', allData='true'):
    url = 'https://min-api.cryptocompare.com/data/histoday?fsym={}&tsym={}&lim
it={}&aggregate={}&allData={}'\
            .format(symbol.upper(), comparison_symbol.upper(), limit, aggregat
e, allData)
    if exchange:
        url += '&e={}'.format(exchange)
    page = requests.get(url)
    data = page.json()['Data']
    df = pd.DataFrame(data)
    df['date'] = [datetime.datetime.fromtimestamp(d).date() for d in df.time]
    return df
```

In [82]:
```
# for example, historical daily price of Bitcoin(BTC) in USD
daily_price_historical('BTC','USD')
```

Out[82]:

	close	high	low	open	time	volumefrom	
0	0.04951	0.04951	0.04951	0.04951	1279324800	20.00	9.9
1	0.08584	0.08585	0.05941	0.04951	1279411200	75.01	5.0
2	0.08080	0.09307	0.07723	0.08584	1279497600	574.00	4.9
3	0.07474	0.08181	0.07426	0.08080	1279584000	262.00	2.0
4	0.07921	0.07921	0.06634	0.07474	1279670400	575.00	4.2
5	0.05050	0.08181	0.05050	0.07921	1279756800	2160.00	1.2
6	0.06262	0.06767	0.05050	0.05050	1279843200	2402.50	1.4
7	0.05454	0.06161	0.05049	0.06262	1279929600	496.32	2.6
8	0.05050	0.05941	0.05050	0.05454	1280016000	1551.48	8.5
9	0.05600	0.05600	0.05000	0.05050	1280102400	877.00	4.6
10	0.06000	0.06050	0.05300	0.05600	1280188800	3373.69	1.9
11	0.05890	0.06200	0.05400	0.06000	1280275200	4390.29	2.5
12	0.06990	0.06990	0.05710	0.05890	1280361600	8058.49	5.2
13	0.06270	0.06980	0.05820	0.06990	1280448000	3020.85	1.9
14	0.06785	0.06889	0.05600	0.06270	1280534400	4022.25	2.4
15	0.06110	0.06500	0.06000	0.06785	1280620800	2601.00	1.6
16	0.06000	0.06330	0.06000	0.06110	1280707200	3599.00	2.2
17	0.06000	0.06500	0.05900	0.06000	1280793600	9821.46	6.0
18	0.05700	0.06231	0.05700	0.06000	1280880000	3494.00	2.1

	close	high	low	open	time	volumefrom	
19	0.06100	0.06100	0.05800	0.05700	1280966400	5034.07	3.0
20	0.06230	0.06240	0.06070	0.06100	1281052800	1395.00	8.5
21	0.05900	0.06220	0.05900	0.06230	1281139200	2619.00	1.5
22	0.06090	0.06100	0.05900	0.05900	1281225600	2201.00	1.3
23	0.07100	0.07350	0.05930	0.06090	1281312000	13631.09	8.8
24	0.07000	0.07090	0.06651	0.07100	1281398400	1310.39	8.8
25	0.06700	0.07541	0.06000	0.07000	1281484800	14061.18	1.0
26	0.07000	0.07000	0.06141	0.06700	1281571200	2062.31	1.3
27	0.06450	0.06800	0.06450	0.07000	1281657600	3591.77	2.3
28	0.06700	0.06950	0.06450	0.06450	1281744000	4404.20	2.9
29	0.06529	0.06700	0.06500	0.06700	1281830400	4462.87	2.9
...	...	...	...	...	...	...	...
2740	11282.49000	13648.84000	10032.69000	13634.60000	1516060800	325702.79	3.8
2741	11162.70000	11736.30000	9205.38000	11282.49000	1516147200	348631.94	3.6
2742	11175.52000	12018.43000	10642.33000	11162.70000	1516233600	204918.02	2.3
2743	11521.76000	11780.49000	10867.18000	11175.52000	1516320000	110885.87	1.2
2744	12783.94000	13031.04000	11502.11000	11521.82000	1516406400	119084.84	1.4
2745	11549.93000	12787.35000	11101.73000	12783.54000	1516492800	130427.64	1.5
2746	10814.52000	11913.74000	10067.76000	11549.98000	1516579200	165723.08	1.8

	close	high	low	open	time	volumefrom	
2747	10858.23000	11388.52000	9980.50000	10814.52000	1516665600	158868.59	1.7
2748	11429.02000	11531.60000	10506.55000	10853.78000	1516752000	115804.82	1.2
2749	11175.87000	11741.92000	10930.34000	11428.11000	1516838400	93317.76	1.0
2750	11104.20000	11656.54000	10346.86000	11175.87000	1516924800	142350.84	1.5
2751	11459.71000	11638.69000	10879.20000	11104.34000	1517011200	90873.81	1.0
2752	11767.74000	12064.19000	11407.94000	11460.39000	1517097600	87917.22	1.0
2753	11233.95000	11860.29000	11089.52000	11767.74000	1517184000	80713.70	9.2
2754	10107.26000	11263.70000	9871.21000	11234.32000	1517270400	164072.93	1.7
2755	10226.86000	10377.96000	9698.13000	10107.40000	1517356800	122260.49	1.2
2756	9114.72000	10280.84000	8726.95000	10226.86000	1517443200	208918.80	1.9
2757	8870.82000	9147.93000	7786.20000	9114.73000	1517529600	322596.22	2.7
2758	9251.27000	9504.37000	8194.68000	8872.87000	1517616000	139226.07	1.2
2759	8218.05000	9400.99000	7889.83000	9251.27000	1517702400	164609.06	1.4
2760	6937.08000	8391.29000	6627.31000	8218.05000	1517788800	341828.54	2.5
2761	7701.25000	7932.38000	5968.36000	6936.43000	1517875200	495883.24	3.3
2762	7592.72000	8572.68000	7208.86000	7701.25000	1517961600	271450.37	2.1
2763	8260.69000	8643.94000	7590.48000	7593.78000	1518048000	193040.33	1.5
2764	8696.83000	8743.20000	7775.36000	8259.26000	1518134400	162279.68	1.3
2765	8569.29000	9081.49000	8176.25000	8696.83000	1518220800	155616.78	1.3

	close	high	low	open	time	volumefrom	
2766	8084.61000	8573.35000	7862.31000	8569.32000	1518307200	123293.84	1.0
2767	8911.27000	8997.34000	8084.41000	8084.61000	1518393600	124923.98	1.0
2768	8544.69000	8955.15000	8379.35000	8911.17000	1518480000	98632.88	8.5
2769	9268.66000	9382.37000	8542.98000	8544.69000	1518566400	116219.67	1.0

2770 rows × 8 columns

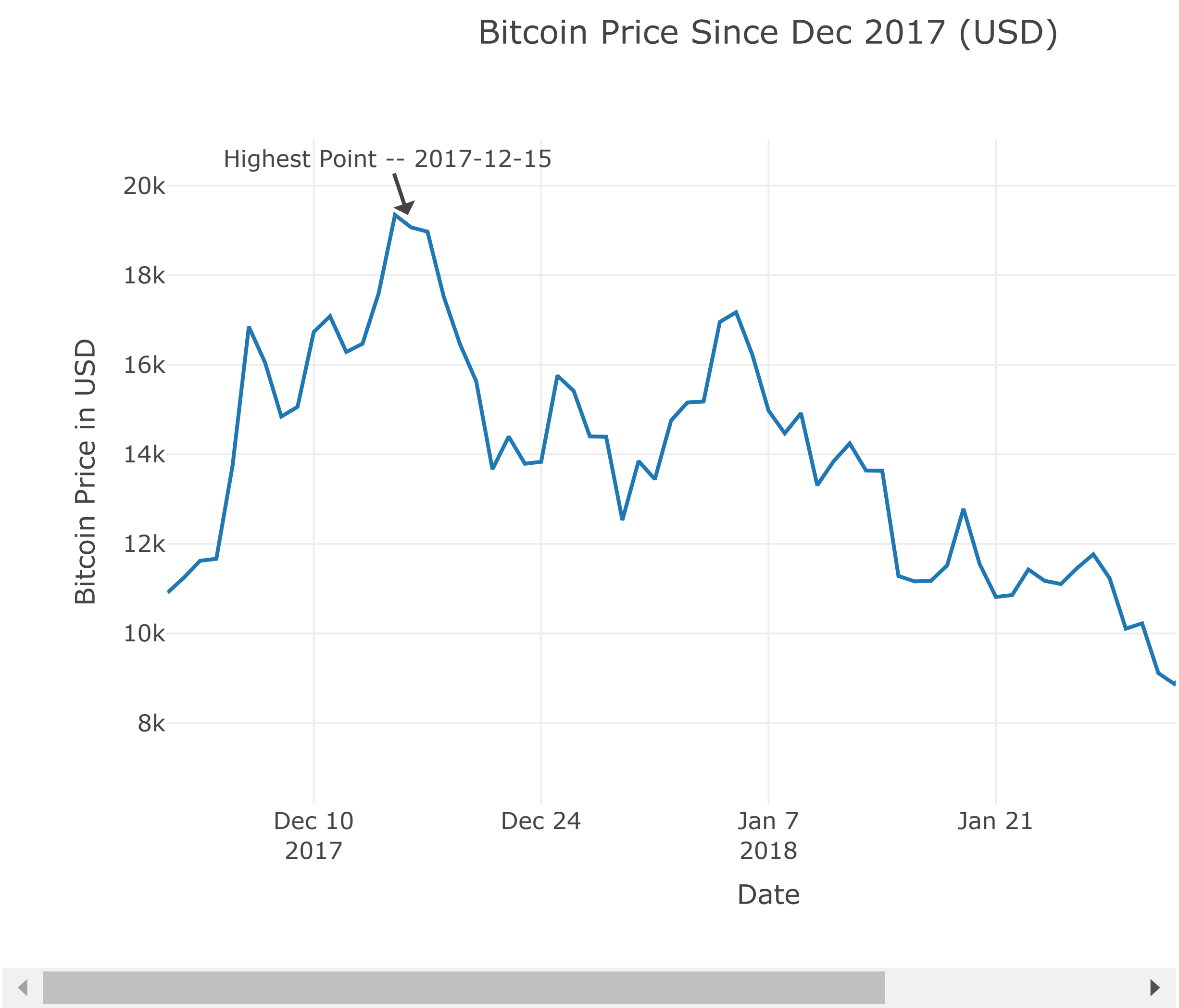
In [88]:
```
# This step pull Bitcoin data since Bear market, which is after 2017-12-01
btcprice = daily_price_historical('BTC','USD')
btcpriceb = btcprice[(btcprice['date'] > datetime.date(2017,11,30))]

btcpriceb.head(30)
```

Out[88]:

	close	high	low	open	time	volumefrom	volumeto	
2695	10912.73	11175.23	10715.55	10861.47	1512172800	86825.51	9.504742e+08	20 12
2696	11246.21	11851.09	10578.43	10912.72	1512259200	122125.70	1.380012e+09	20 12
2697	11623.91	11624.63	10917.81	11244.20	1512345600	93173.90	1.057859e+09	20 12
2698	11667.13	11901.87	11486.13	11624.37	1512432000	89687.21	1.048839e+09	20 12
2699	13749.57	13843.20	11661.76	11667.13	1512518400	191576.66	2.437038e+09	20 12
2700	16850.31	16879.26	13401.61	13750.09	1512604800	297108.66	4.510225e+09	20 12
2701	16047.61	17294.85	13906.10	16867.98	1512691200	286762.02	4.546015e+09	20 12
2702	14843.42	16313.18	13151.47	16048.18	1512777600	181979.81	2.699876e+09	20 12
2703	15059.60	15783.20	13031.00	14839.98	1512864000	201620.09	2.904038e+09	20 12
2704	16732.47	17399.18	15024.56	15060.45	1512950400	159724.56	2.634268e+09	20 12
2705	17083.90	17560.65	16254.53	16733.29	1513036800	132846.57	2.246139e+09	20 12
2706	16286.82	17267.96	15669.86	17083.90	1513123200	155407.35	2.576056e+09	20 12
2707	16467.91	16941.08	16023.64	16286.82	1513209600	107918.03	1.773814e+09	20 12
2708	17604.85	17987.03	16442.20	16467.91	1513296000	153651.15	2.682351e+09	20 12
2709	19345.49	19587.70	17318.54	17594.08	1513382400	112173.97	2.078806e+09	20 12
2710	19065.71	19870.62	18750.91	19346.60	1513468800	117408.38	2.264650e+09	20 12
2711	18972.32	19221.10	18114.42	19065.71	1513555200	139251.39	2.597510e+09	20 12
2712	17523.70	19021.97	16812.80	18971.19	1513641600	174543.37	3.136709e+09	20 12
2713	16461.97	17813.60	15642.69	17521.73	1513728000	227676.13	3.791753e+09	20 12

	close	high	low	open	time	volumefrom	volumeto	c
2714	15632.12	17301.83	14952.98	16461.09	1513814400	163735.02	2.619295e+09	20 12
2715	13664.97	15823.72	10875.71	15632.12	1513900800	466980.60	6.245732e+09	20 12
2716	14396.46	15493.23	13356.07	13664.97	1513987200	170169.39	2.491903e+09	20 12
2717	13789.95	14413.72	12166.45	14396.63	1514073600	182417.29	2.428438e+09	20 12
2718	13833.49	14467.43	13010.71	13789.95	1514160000	107475.98	1.487888e+09	20 12
2719	15756.56	16094.67	13748.49	13830.19	1514246400	143135.26	2.198577e+09	20 12
2720	15416.64	16514.59	14534.66	15757.02	1514332800	138705.28	2.162831e+09	20 12
2721	14398.70	15505.51	13466.07	15416.34	1514419200	170366.63	2.425913e+09	20 12
2722	14392.57	15109.81	13951.08	14398.45	1514505600	118874.63	1.733584e+09	20 12
2723	12531.52	14461.46	11962.09	14392.14	1514592000	182065.44	2.387311e+09	20 12
2724	13850.40	14241.82	12359.43	12532.38	1514678400	111270.55	1.492142e+09	20 12

In [89]:
```
# Chart the BTC pricing data since Bear market
btc_trace = go.Scatter(x=btcpriceb['date'], y=btcpriceb['close'])
data_trace=go.Data([btc_trace])
layout=go.Layout(title="Bitcoin Price Since Dec 2017 (USD)", xaxis={'title':'D
ate'}, yaxis={'title':'Bitcoin Price in USD'})
layout.update(dict(annotations=[go.Annotation(text="Highest Point -- 2017-12-1
5", x="2017-12-15 19:00:00", y="19345.49")]))
#layout.update(dict(annotations=[go.Annotation(text="Lowest Point -- 2018-02-0
4", x="2018-02-04 19:00:00", y="6937.08")]))
figure=go.Figure(data=data_trace,layout=layout)
py.iplot(figure)
```



we can see that the price dropped significantly from almost 20k (peak at 2017-12-15)
to below 7k (bottom on 2018-02-04)
for my data selection I will select data since Dec 2017,
then we include the 2018 bear market data and a little bit of bull market ata (pre Dec 2017)

Coins selection

In the analysis i will choose the mainstream coins and small coins, list below:

Mainstream Coins

- BTC -- Bitcoin
- ETH -- Ethereum
- LTC -- Litecoin
- XRP -- Ripple
- ETC -- Ethereum Classic

Non - Mainstream Coins

- XLM -- Stellar
- INK
- ELF
- XRB
- INS
- SRN
- BCD
- DBC
- BCPT

In [90]:

```
# getting all pricin data of coins
coins = ['BTC','ETH','LTC','XRP','XLM','XRB','BCD','BCPT','ZCL','LSK','OMG']
# 'DBC','ELF','INK','INS','SRN','ETC',
coin_data = {}
for coin in coins:
    crypto_price_df = daily_price_historical(coin, 'USD')
    coin_data[coin] = crypto_price_df[(crypto_price_df['date'] > datetime.date
(2017,11,30))].set_index('date')

# pricing data of Bitcoin
coin_data['BTC'].tail()
```

Out[90]:

	close	high	low	open	time	volumefrom	volumeto
date							
2018-02-09	8569.29	9081.49	8176.25	8696.83	1518220800	155616.78	1.348923e+09
2018-02-10	8084.61	8573.35	7862.31	8569.32	1518307200	123293.84	1.013772e+09
2018-02-11	8911.27	8997.34	8084.41	8084.61	1518393600	124923.98	1.085922e+09
2018-02-12	8544.69	8955.15	8379.35	8911.17	1518480000	98632.88	8.533204e+08
2018-02-13	9268.66	9382.37	8542.98	8544.69	1518566400	116219.67	1.056246e+09

In [98]:
```
# In this study we only use daily close data for pricing analysis
# This step merge daily close price of all coins

def merge_dfs_on_column(dataframes, labels, col):
    '''Merge a single column of each dataframe into a new combined datafram
e'''
    series_dict = {}
    for index in range(len(dataframes)):
        series_dict[labels[index]] = dataframes[index][col]

    return pd.DataFrame(series_dict)

combined_df = merge_dfs_on_column(list(coin_data.values()), list(coin_data.key
s()), 'close')
combined_df.head()
```

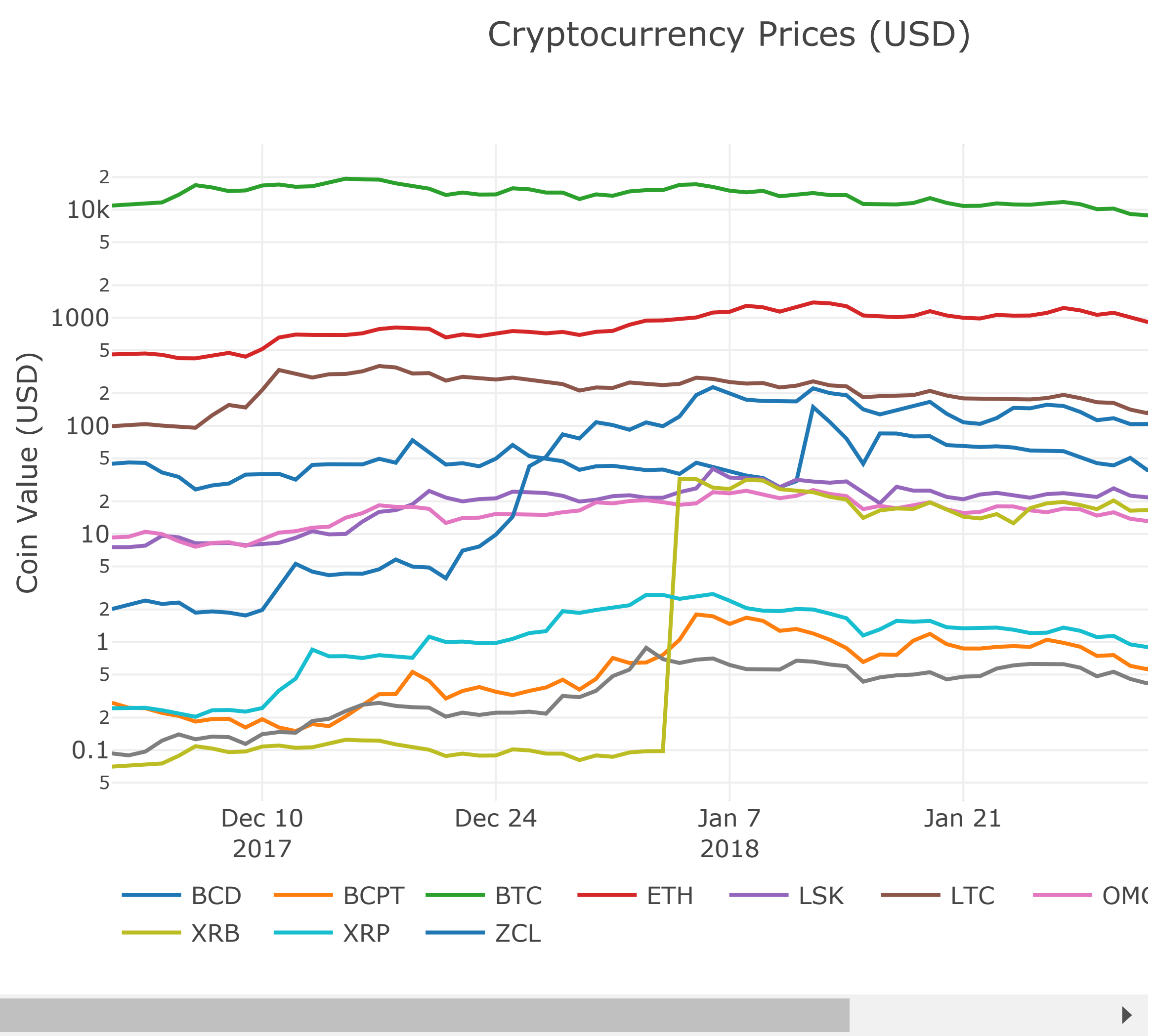
Out[98]:

	BCD	BCPT	BTC	ETH	LSK	LTC	OMG	XLM	XRB	XRP	ZCL
date											
2017-12-01	44.59	0.2743	10912.73	457.96	7.55	99.32	9.28	0.09321	0.07039	0.2441	2.02
2017-12-02	45.84	0.2467	11246.21	462.81	7.55	100.70	9.42	0.08946	0.07254	0.2449	2.22
2017-12-03	45.46	0.2441	11623.91	466.93	7.80	103.87	10.48	0.09701	0.07497	0.2462	2.42
2017-12-04	37.02	0.2209	11667.13	453.96	9.61	100.49	9.97	0.12250	0.07525	0.2337	2.25
2017-12-05	33.69	0.2073	13749.57	422.48	9.30	98.97	8.56	0.13980	0.08868	0.2182	2.32

In [95]:
```
# saving the result to excel since the cryptocompare API is unstable sometimes
# combined_df.to_excel('coinprices2.xlsx')
```

In [99]:
```
# function to make a neat plot of coins price

def df_scatter(df, title, seperate_y_axis=False, y_axis_label='', scale='linea
r', initial_hide=False):
    '''Generate a scatter plot of the entire dataframe'''
    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 [100]:
```
df_scatter(combined_df, 'Cryptocurrency Prices (USD)', seperate_y_axis=False,
y_axis_label='Coin Value (USD)', scale='log')
```



I'm pretty happy with this plot, it plot nicely of coin prices in USD

However, the price does not flutuate much and its difficult to see overall trend.
Later I will use a min max scaler to scale all coin prices between 0 to 1.
This can also help us better understand the correlation and how sensitive coin prices are

However, before doing that, lets see the pearson correlation of the prices first

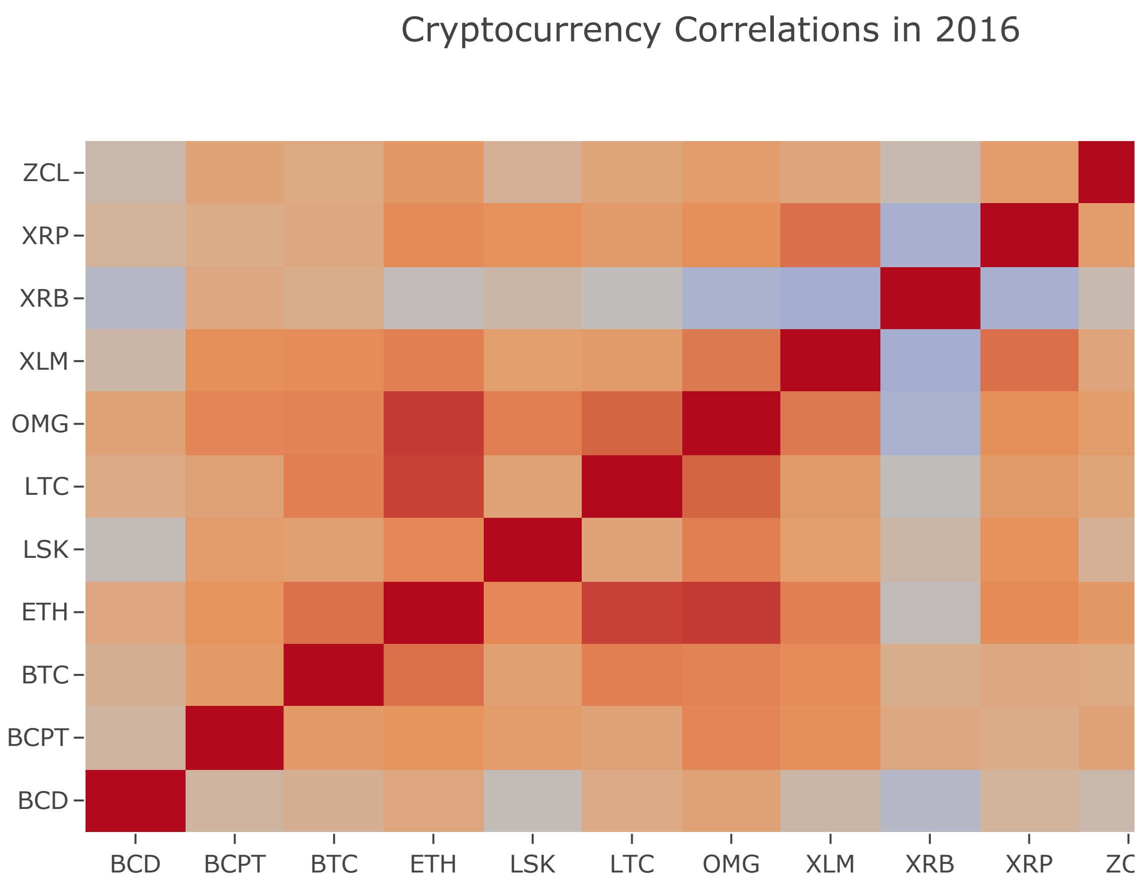
In [102]:
```
combined_df.pct_change().corr(method='pearson')
```

Out[102]:

	BCD	BCPT	BTC	ETH	LSK	LTC	OMG	XLM
BCD	1.000000	0.105505	0.147676	0.229645	0.019818	0.199238	0.264402	0.084720
BCPT	0.105505	1.000000	0.333882	0.379094	0.316747	0.267289	0.450353	0.410483
BTC	0.147676	0.333882	1.000000	0.543854	0.285025	0.479254	0.459421	0.415910
ETH	0.229645	0.379094	0.543854	1.000000	0.444152	0.753897	0.789921	0.478554
LSK	0.019818	0.316747	0.285025	0.444152	1.000000	0.256409	0.486154	0.297973
LTC	0.199238	0.267289	0.479254	0.753897	0.256409	1.000000	0.601014	0.324560
OMG	0.264402	0.450353	0.459421	0.789921	0.486154	0.601014	1.000000	0.504738
XLM	0.084720	0.410483	0.415910	0.478554	0.297973	0.324560	0.504738	1.000000
XRB	-0.032425	0.216246	0.180996	0.017901	0.085950	0.010138	-0.073945	-0.088838
XRP	0.124707	0.184859	0.213949	0.423388	0.402816	0.325668	0.407156	0.545251
ZCL	0.067581	0.256657	0.209760	0.344826	0.139394	0.241565	0.323604	0.238797

In [103]:
```
# Heatmap visualization to more clearly see the correlation,
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 [104]:
```
correlation_heatmap(combined_df.pct_change(), "Cryptocurrency Correlations in
 2016")
```



From the correlation matrix we can see that most coins are either posititively correlated with each other or slightly(potentially not statistically significant) negatively correlated.
Negative correlations could be significant between a small coin and other coins.
Mainstream coins are all positively correlatied with each other

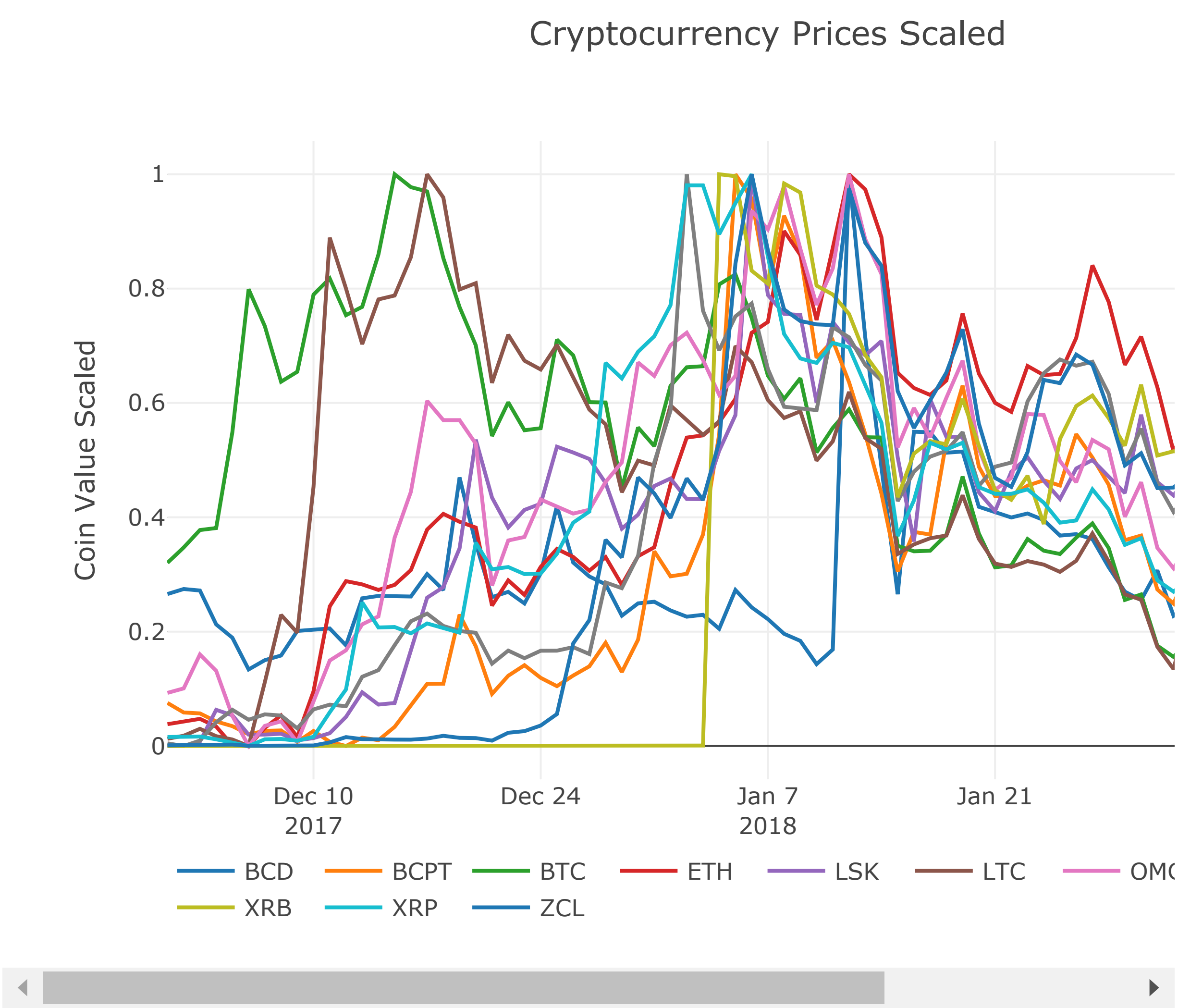
In [111]:
```
# import MinMaxScaler to scale price from 0 to 1
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [116]:
```
# Transform the combined_df to combined_df_scaled, which scale prices from 0 t
o 1
combined_df_scaled = pd.DataFrame(scaler.fit_transform(combined_df), columns=c
ombined_df.columns, index = combined_df.index)
combined_df_scaled.head()
```

Out[116]:

	BCD	BCPT	BTC	ETH	LSK	LTC	OMG	XLM	X
date									
2017-12-01	0.265816	0.075333	0.320400	0.038190	0.000000	0.012695	0.092945	0.004715	0.
2017-12-02	0.274583	0.058606	0.347275	0.043222	0.000000	0.017973	0.100784	0.000000	0.
2017-12-03	0.271918	0.057030	0.377714	0.047496	0.007676	0.030094	0.160134	0.009493	0.
2017-12-04	0.212723	0.042970	0.381197	0.034040	0.063248	0.017170	0.131579	0.041542	0.
2017-12-05	0.189367	0.034727	0.549022	0.001380	0.053730	0.011357	0.052632	0.063294	0.

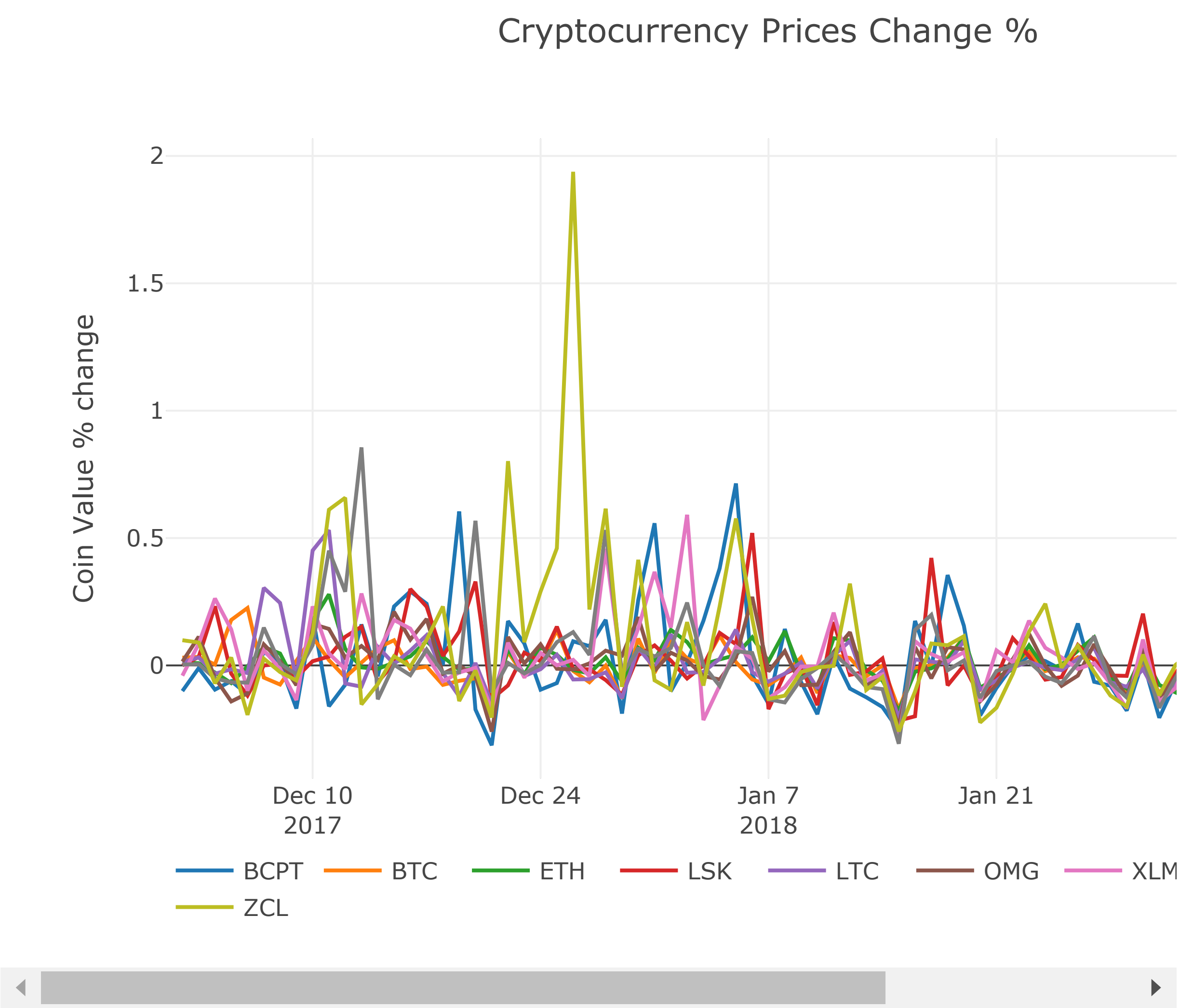
In [114]:
```
# Plot it
df_scatter(combined_df_scaled, 'Cryptocurrency Prices Scaled', seperate_y_axis
=False, y_axis_label='Coin Value Scaled')
```



I'm pretty happy with the result,

However its still a little difficult to the sensitivity since price moves all the time
It would be bette if we can try the plot of daily price percentage change

In [117]:
```
#combined_df.pct_change()
# I dropped certain coins since i found them are outliers and affect the scale
 (y axix) too much
# I will talk about them separately later
df_scatter(combined_df.pct_change().drop('XRB', 1).drop('BCD', 1),
           'Cryptocurrency Prices Change %', seperate_y_axis=False, y_axis_lab
el='Coin Value % change', )
```



I'm happy with this plot!

We can tell a lot from this plot.

- First, we can easily tell that there are almost no lag of coin prices of non-mainstream coins.
  Expecially when there is lots of fluctuations
  For example at the significant ups/downs: Dec 9, Dec 21, Jan 15, Feb 4
  Possible reason is possibly because cryptocurrency can trade instantly so basically lead time in trading coins if people think cryptocurrency is not doing good
- Second, we can see the most volatile coins -- BCPT, XRP, LSK, ZCL -- are all non-mainstream coins.

Future Studies:

- Data wise:
  import price of all coins and use PCA to cluster them into groups
  import some google trend data, twitter data and stock price
- Analysis wise:
  Classify non mainstream coins into scam/non-scam and analize them separately

Scratch area

Coin List

In [ ]:
```
def coin_list():
    url = 'https://www.cryptocompare.com/api/data/coinlist/'
    page = requests.get(url)
    data = page.json()['Data']
    return data
```

In [ ]:

In [ ]:
```
coins = ['BTC','ETH','LTC','XRP','ETC','XLM','INK','ELF','XRB','INS','SRN','BC
D','DBC','BCPT']
#
coin_data = {}
for coin in coins:
    crypto_price_df = daily_price_historical(coin, 'USD')
    coin_data[coin] = crypto_price_df[(crypto_price_df['timestamp'] > '2017-12
-15')]

coin_data['BTC'].tail()
```

In [ ]:
```
coin_data['BTC']
```

In [ ]:
```
import statsmodels.api as sm
```

In [ ]:
```
from statsmodels.tsa.api import VAR, DynamicVAR
```

In [ ]:
```
mdata = sm.datasets.macrodata.load_pandas().data
mdata
```

In [ ]:

In [ ]:
```
# Chart the BTC pricing data since Bear market
btc_trace = go.Scatter(x=btcpriceb['timestamp'], y=btcpriceb['close'])
data=go.Data([btc_trace])
layout=go.Layout(title="Bitcoin Price Since Dec 2017 (USD)", xaxis={'title'
:'Date'}, yaxis={'title':'Bitcoin Price in USD'})
figure=go.Figure(data=data,layout=layout)
py.iplot(figure)

#we can see that the price dropped significantly from almost 20k (peak at D
ec 17th) to below 8k
```

In [ ]:
```
def price(symbol, comparison_symbols=['USD'], exchange=''):
    url = 'https://min-api.cryptocompare.com/data/price?fsym={}&tsyms={}'\
            .format(symbol.upper(), ','.join(comparison_symbols).upper())
    if exchange:
        url += '&e={}'.format(exchange)
    page = requests.get(url)
    data = page.json()
    return data
```

In [ ]:
```
coins = ['BTC','ETH','LTC','XRP','ETC','XLM','INK','ELF','XRB','INS','SRN','BC
D','DBC','BCPT']
#
coin_data = {}
for coin in coins:
    crypto_price_df = daily_price_historical(coin, 'USD')
#    coin_data[coin] = crypto_price_df.set_index('timestamp')
#    coin_data[coin] = crypto_price_df[(crypto_price_df['timestamp'] > '2017-1
2-15')]
    coin_data[coin] = crypto_price_df
#coin_data['BTC'].tail()
crypto_price_df
```

Useful links:

- CryptoCompare API Quick Start
  This is a very useful link for the cryptocompare API, which is where I got data from this study
  https://github.com/agalea91/cryptocompare-api/blob/master/CryptoCompare.API.2017.08.ipynb (https://github.com/agalea91/cryptocompare-api/blob/master/CryptoCompare.API.2017.08.ipynb)

- Cryptocurrency Analysis Python
  https://github.com/triestpa/Cryptocurrency-Analysis-Python (https://github.com/triestpa/Cryptocurrency-Analysis-Python)

- CryptoAsset Portfolios: Identifying Highly Correlated Cryptocurrencies using PCA
  http://www.quantatrisk.com/2017/03/31/cryptocurrency-portfolio-correlation-pca-python/ (http://www.quantatrisk.com/2017/03/31/cryptocurrency-portfolio-correlation-pca-python/)

  Future studies
- Useful link of tsa lag and correlation
  https://stackoverflow.com/questions/25320773/time-series-correlation-and-lag-time (https://stackoverflow.com/questions/25320773/time-series-correlation-and-lag-time)