FINM 33150 Final Project

HTML version is included as well, one of the graphs has some trouble while displaying

Title:

$$Pair\ Trading\ Using\ Distance\ Method\ in\ Crypto\ Market$$

1. Introduction

Asset Class Choosing: Why Crypto? - Cryptocurrency Market Overview

- Cryptocurrency is known as a digital currency that operates using a decentralized system, meaning that it is free of limitations neither central issuing nor authority regulations. Relying on blockchain and other cryptographic technologies to authenticate transactions, it ensures buyers and sellers with adequate security, transparency, and immutability of transactions.
- Aside from the extent of financial freedom and privacy that it provides people with, it is also an attractive investment for investors because of the high returns due to its volatility. This could come from various factors including the market demand and rationality, government regulatory policies, or potential security concerns.
- In reality, the cryptocurrency market has exhibited significant growth and volatility since its inception, with a current market capitalization of over 2 trillion. Currently, Bitcoin is the largest cryptocurrency by market capitalization, which has surpassed the 50,000 mark for the first time in several months but also experienced several significant dips in its value. Other major cryptocurrencies including Ethereum, Binance Coin, and Cardano also had significant gains in value. Simultaneously, there were also concerns about the sustainability of the market's growth.
- While high volatility could indicate severe risks for investors, with proper analysis of the underlying technical patterns of particular cryptocurrencies and strategic intra-day trading, we might be able to generate returns from these volatile movements. In this project, we will not discuss in detail the commonly known properties of cryptocurrencies and the mechanics behind them, but will instead focus on analyzing the crypto-market performance and the profitability of quantitative trading strategies associated with cryptocurrencies, namely, pair trading using the Distance Method.

Literature Review

- We investigated the statistical arbitrage methods commonly used in the cryptocurrency market to establish a quantitative trading strategy. These include mean reversion, pairs trading, and momentum strategies.

- In mean reversion strategies, traders assume that the price of an asset will eventually return to its expected value and try to identify occurrences where the price of a particular cryptocurrency deviates from its historical average. On the other hand, pairs trading involves buying one cryptocurrency and simultaneously selling another similar one, with the anticipation that the relative differences of prices of the two assets will converge eventually. In momentum strategies for cryptocurrency trading, traders look for strong price movement patterns in a singular direction and trades on its continuing growth.
- Pair trading presents as a simple but profitable strategy throughout a long time in different studies and empirical tests. Following Kolmogorov and Ramazanov (2019)'s research, we discovered common pair trading strategies including the distance method, cointegration method, and copula method. The "Distance Method" stands out as a very flexible option for trading cryptocurrencies.
- Gatev et al. (2006) invented this simple pair trading strategy called the Distance Method. He discovered its remarkable profits over an extended period, specifically, throughout 196 years of US stock market data. The Distance Method quantifies the relative "distance" between two cryptocurrencies based on statistical measures, and chooses to buy or hold one of the assets whenever the distance exceeds a predetermined threshold. This brought our attention to this old-fashioned but popular strategy and its historical profitability as examined in previous research.
- On the contrary, according to Do and Faff (2012), the profitability of pairs trading has decreased due to fewer arbitrage opportunities in the market, evidenced by the rise in the number of the cryptocurrencies pairs that never converge as anticipated. They studied pairs trading profitability across a range of US equity sectors, but found out that the profitability for pairs trading decreased as trading costs were incorporated. The study discovered that one could attain potentially higher profits in less liquid and less efficient markets, which somewhat aligns with the Efficient Market Hypothesis

Motivation on Our Strategy

- While some disputes centering around the robustness of different pair trading strategies exist, in the paper "Pairs Trading in Cryptocurrency Markets" by Jacobs and Weber (2015), the authors investigated the application of pairs trading strategy in cryptocurrency markets and analyzed its profitability using a sample of 10 major cryptocurrencies. They found out that pairs trading strategies in fact could be profitable and were robust to transaction costs. Although the results varied for different cryptocurrencies and strategies, the highest returns were observed for pairs involving Bitcoin.
- The truth of profitability cannot be generalized for the cryptocurrency market, which is why we will experiment with the traditional "Distance Method", following the customized procedures. We will build up the strategy starting with 10 pairs of cryptocurrencies and make selections based on some established criterion.
- Therefore, we will reference previous research papers on pair trading strategies including "Pairs Trading in Cryptocurrency Markets" by Miroslav Fil and Ladislav Kristoufek (2018). We replicate their methodologies or decision making to a certain extent. Replications and discretionary changes will either be documented or justified.

2. Strategy Introduction

Pair Trading Strategy using Distance Method

- Begin with the original dataset. First, we gathered data from 10 cryptocurrencies including Bitcoin that have been actively traded since 2019 with a 5-minute frequency from Binance to form 10 pairs with Bitcoin.
- We calculate and rank pair wise SSD (sum of squared deviations). Here, we will use the cryptocurrencies' closing price and more importantly, the SSD is between the normalized logarithmic price series of the paired assets.
- The sum of squared deviations between two normalized logarithmic price series of assets i and j, simply labeled as P here:

$$SSD_{ij} = \sum_t (P_{it} - P_{jt})^2$$

- The trading mechanics could be summarized as follows:
  - Compute the ratio of the close prices of the cryptocurrencies for each pair
  - Using Exponentially Weighted EW calculations to determine the upper and lower bound to use for mean reversion trading
- Below is how the upper and lower bond is defined:

$$UpperLimit = EWMA_{PriceRatio} + threshold * EWMSTD_{PriceRatio}$$
$$LowerLimit = EWMA_{PriceRatio} - threshold * EWMSTD_{PriceRatio}$$

- Define the spread as Long BTC and Short the other coin. Long the spread when the ratio is below the lower limit; Short the spread when the ratio exceeds the upper limit.
- Weights for each cryptocurrencies are determined by finding the tangency weights and are updated periodically

3. Get Cryptodata From Binance

Introducing the cryptocurrencies we have selected for the final project:

Cryptocurrency's Name	Symbol	Brief Introduction
Bitcoin	BTC	Bitcoin is the coin people generally reference when they talk about digital currency.
Ethereum	ETH	The system allows you to use ether (the currency) to perform a number of functions.
BNB	BNB	BNB is the cryptocurrency issued by Binance, among the largest crypto exchanges in the world.
Cardano	ADA	Created by the co-founder of Ethereum, Cardano also uses smart contracts.
Polygon	MATIC	Focused on being accessible to creating digital apps and scales up the Ethereum cryptocurrency.
Dogecoin	DOGE	Originally created as a joke after the run-up in Bitcoin.
Solana	SOL	Newer cryptocurrency and it touts its speed at completing transactions.

Cryptocurrency's Name	Symbol	Brief Introduction
Polkadot	DOT	It connected the technology of blockchain from many different cryptocurrencies.
Litecoin	LTC	Litecoins are generated faster than Bitcoin, but Bitcoin is worth more.

Most functions are included in the Utils.py file

In [77]:
```python

import ccxt
import pandas as pd
import numpy as np
import urllib
import json
from typing import List
import datetime as dt
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import warnings
from functools import reduce
from plotly.subplots import make_subplots
import plotly.graph_objects as go
warnings.filterwarnings('ignore')
import math
from scipy.stats.mstats import gmean
#using functions from utils.py file
import utils as utils
import statsmodels.api as sm

```

Notice: Since Binance has location restriction, the United States users do not have accessibility of the data directly. We obtain the data using the following function in a different IP location and stored selected cryptocurrencies in csv files with frequency hourly. Functions are displayed since we have used it.

Here, we load CSV files, here, we displayed BTC as an example

In [7]:
```python
folder = '/Users/dennishua/Downloads/Hua_Lyu_Ge_Li_final/Crypto Da
#folder = 'C:\\Users\\Owner\\Desktop\\qts project\\'
```

In [8]:
```python
BTC_Data = utils.load_and_clean_data(folder+'BTCUSDT_5m_2019-01-01
```

In [9]:
```
BTC_Data
```

Out[9]:

	Open	High	Low	Close	Volume	Log_close	5-Min Return
Time							
2019-01-01 00:00:00	3701.23	3703.72	3695.00	3696.32	85.572181	8.215093	NaN
2019-01-01 00:05:00	3696.30	3697.24	3689.88	3692.34	62.296581	8.214016	-0.001077
2019-01-01 00:10:00	3692.34	3698.93	3692.34	3697.31	43.105333	8.215361	0.001346
2019-01-01 00:15:00	3697.91	3698.75	3693.00	3693.00	48.551084	8.214194	-0.001166
2019-01-01 00:20:00	3693.44	3695.98	3690.92	3692.18	47.706443	8.213972	-0.000222
...	...	...	...	...	...	...	...
2023-02-02 00:55:00	24200.02	24214.25	24152.65	24189.85	1394.042710	10.093688	-0.000420
2023-02-02 01:00:00	24188.09	24209.91	24117.00	24132.98	1542.742520	10.091335	-0.002351
2023-02-02 01:05:00	24132.98	24136.27	24053.57	24093.79	1979.277790	10.089709	-0.001624
2023-02-02 01:10:00	24093.79	24099.72	24045.29	24063.89	1318.924240	10.088468	-0.001241
2023-02-02 01:15:00	24062.36	24107.00	24062.26	24096.10	1499.251360	10.089805	0.001339

429198 rows × 7 columns

In [10]:
```
utils.load_and_clean_data(folder+'BTCUSDT_5m_2019-01-01_2023-02-01
```

Out[10]:

	Open	High	Low	Close	Volume	Log_close	5-Min Return
Time							
2019-01-01 00:00:00	3701.23	3703.72	3695.00	3696.32	85.572181	8.215093	NaN
2019-01-01 00:05:00	3696.30	3697.24	3689.88	3692.34	62.296581	8.214016	-0.001077
2019-01-01 00:10:00	3692.34	3698.93	3692.34	3697.31	43.105333	8.215361	0.001346
2019-01-01 00:15:00	3697.91	3698.75	3693.00	3693.00	48.551084	8.214194	-0.001166
2019-01-01 00:20:00	3693.44	3695.98	3690.92	3692.18	47.706443	8.213972	-0.000222
...	...	...	...	...	...	...	...
2023-02-02 00:55:00	24200.02	24214.25	24152.65	24189.85	1394.042710	10.093688	-0.000420
2023-02-02 01:00:00	24188.09	24209.91	24117.00	24132.98	1542.742520	10.091335	-0.002351
2023-02-02 01:05:00	24132.98	24136.27	24053.57	24093.79	1979.277790	10.089709	-0.001624
2023-02-02 01:10:00	24093.79	24099.72	24045.29	24063.89	1318.924240	10.088468	-0.001241
2023-02-02 01:15:00	24062.36	24107.00	24062.26	24096.10	1499.251360	10.089805	0.001339

429198 rows × 7 columns

Then, we collected the 5-minute return from the 10 different cryptocurrencies and put them in one dataframe. We use 2019 data for the initial exploratory analysis. We have chosen 5-min data as it offers an appropriate frequency for the strategy without significant simulation time.

In [11]:
```
label_list = ['BTC','ADA', 'BNB', 'DOGE', 'DOT', 'ETH', 'LINK', 'L
mlist = []

for coin in label_list:
    filepath = folder+f'{coin}USDT_5m_2019-01-01_2023-02-01.csv.xz
    df = utils.load_and_clean_data(filepath)
    df.rename(columns = {'5-Min Return' : f'{coin} 5-Min Return'},
    df_2019 = df.loc[:'2019-12-31'].copy()
    mlist.append(df_2019[f'{coin} 5-Min Return'])

```

In [12]:
```
Crypto_Returns = reduce(lambda left, right: pd.merge(left,right,le
Crypto_Returns
```

Out[12]:

Time	BTC 5-Min Return	ADA 5-Min Return	BNB 5-Min Return	DOGE 5-Min Return	DOT 5-Min Return	ETH 5-Min Return	LINK 5-Min Return	LTC 5-Min Return	MA
2019-01-01 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2019-01-01 00:05:00	-0.001077	-0.000992	-0.000049	NaN	NaN	0.000152	NaN	-0.001005	
2019-01-01 00:10:00	0.001346	-0.001985	0.000836	NaN	NaN	0.000152	NaN	0.000670	
2019-01-01 00:15:00	-0.001166	-0.002735	-0.004259	NaN	NaN	-0.000076	NaN	-0.000670	
2019-01-01 00:20:00	-0.000222	-0.000499	-0.002319	NaN	NaN	-0.000152	NaN	0.000000	
...	...	...	...	...	...	...	...	...	
2019-12-31 23:35:00	0.000323	0.000304	-0.000131	0.00005	NaN	0.000542	0.000227	0.000000	0.
2019-12-31 23:40:00	0.000496	0.000304	0.001065	0.00000	NaN	0.000464	0.000906	0.000000	-0.
2019-12-31 23:45:00	-0.000089	-0.000912	0.000751	0.00000	NaN	0.000232	0.000283	0.002425	0.
2019-12-31 23:50:00	0.001284	0.000913	0.000910	0.00000	NaN	0.000155	0.001131	0.000726	0.
2019-12-31 23:55:00	-0.001222	-0.000912	-0.001921	0.00000	NaN	-0.001237	-0.001582	-0.001692	0.

104768 rows × 10 columns

Correlation Analysis and Graphs

In [15]:

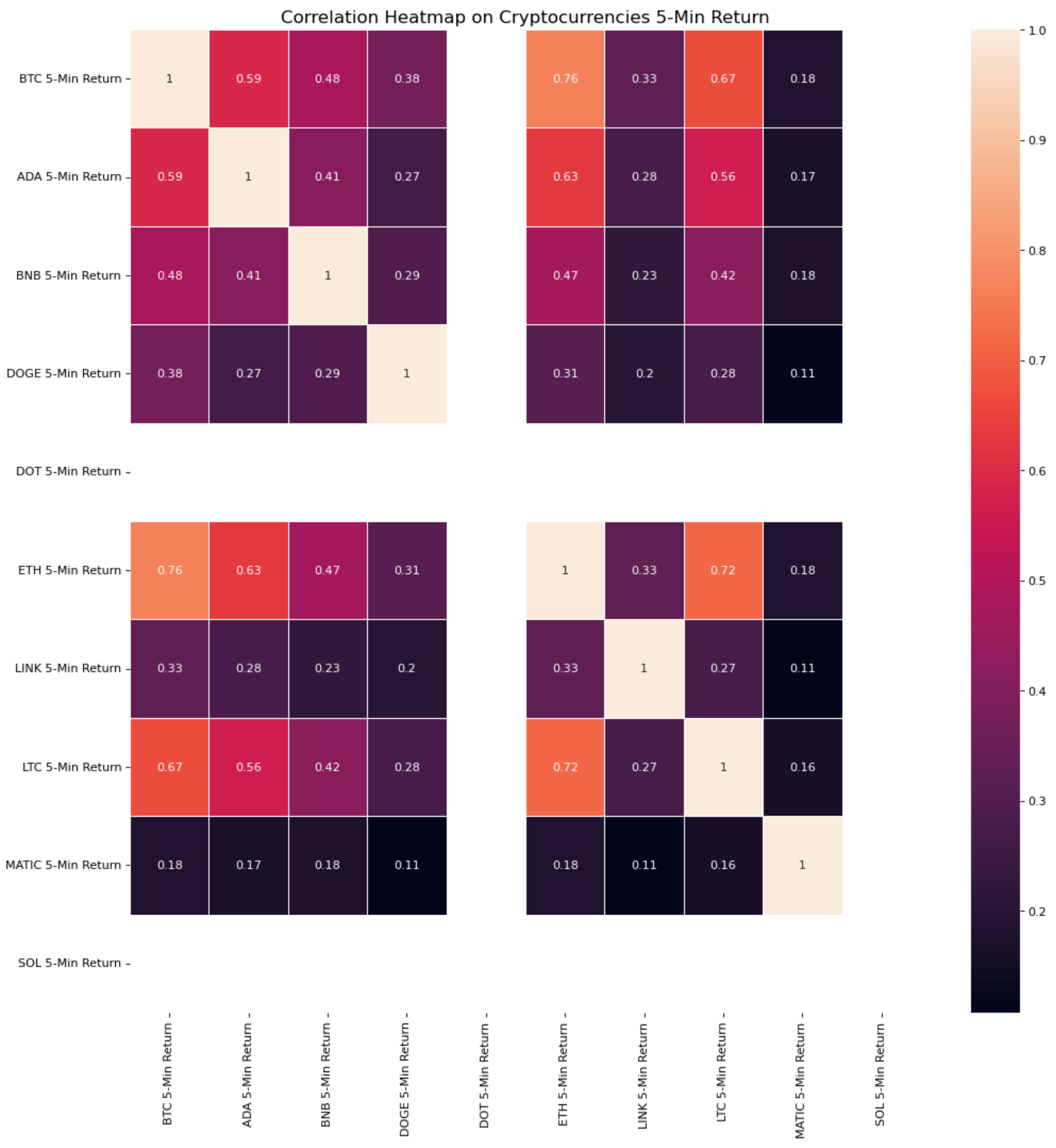
```
Correlation_Matrix = Crypto_Returns.corr()
Correlation_Matrix
```

Out[15]:

	BTC 5-Min Return	ADA 5-Min Return	BNB 5-Min Return	DOGE 5-Min Return	DOT 5-Min Return	ETH 5-Min Return	LINK 5-Min Return	LTC 5-Min Return	MATI 5-Mi Retur
BTC 5-Min Return	1.000000	0.590603	0.480231	0.375354	NaN	0.764214	0.327256	0.667529	0.18311
ADA 5-Min Return	0.590603	1.000000	0.412738	0.267798	NaN	0.632689	0.275094	0.561885	0.16982
BNB 5-Min Return	0.480231	0.412738	1.000000	0.293417	NaN	0.473559	0.225311	0.415766	0.17645
DOGE 5-Min Return	0.375354	0.267798	0.293417	1.000000	NaN	0.312396	0.200506	0.276515	0.10934
DOT 5-Min Return	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
ETH 5-Min Return	0.764214	0.632689	0.473559	0.312396	NaN	1.000000	0.327309	0.716076	0.18336
LINK 5-Min Return	0.327256	0.275094	0.225311	0.200506	NaN	0.327309	1.000000	0.273439	0.10760
LTC 5-Min Return	0.667529	0.561885	0.415766	0.276515	NaN	0.716076	0.273439	1.000000	0.16325
MATIC 5-Min Return	0.183119	0.169822	0.176451	0.109347	NaN	0.183364	0.107608	0.163252	1.00000
SOL 5-Min Return	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

In [16]:
```
plt.figure(figsize = (15,15))
corr_heatmap = sns.heatmap(Correlation_Matrix, annot=True, linewic
corr_heatmap.set_title('Correlation Heatmap on Cryptocurrencies 5-
```

Out[16]:
```
Text(0.5, 1.0, 'Correlation Heatmap on Cryptocurrencies 5-Min Retur
n')
```



	BTC 5-Min Return	ADA 5-Min Return	BNB 5-Min Return	DOGE 5-Min Return	DOT 5-Min Return	ETH 5-Min Return	LINK 5-Min Return	LTC 5-Min Return	MATIC 5-Min Return	SOL 5-Min Return
BTC 5-Min Return	1	0.59	0.48	0.38		0.76	0.33	0.67	0.18	
ADA 5-Min Return	0.59	1	0.41	0.27		0.63	0.28	0.56	0.17	
BNB 5-Min Return	0.48	0.41	1	0.29		0.47	0.23	0.42	0.18	
DOGE 5-Min Return	0.38	0.27	0.29	1		0.31	0.2	0.28	0.11	
DOT 5-Min Return										
ETH 5-Min Return	0.76	0.63	0.47	0.31		1	0.33	0.72	0.18	
LINK 5-Min Return	0.33	0.28	0.23	0.2		0.33	1	0.27	0.11	
LTC 5-Min Return	0.67	0.56	0.42	0.28		0.72	0.27	1	0.16	
MATIC 5-Min Return	0.18	0.17	0.18	0.11		0.18	0.11	0.16	1	
SOL 5-Min Return										

Discussion:

- From the heatmap, we visualize that the 5-Min return of BTC in percentage has a fairly high correlation with mostly all the other cryptocurrencies. This is not quite surprising since BTC is a relative stable and mature cryptocurrency. As we have discussed in the introduction section, one could use the trend of BTC as the overall trend in the cryptocurrency market since the market does not truly depend on other macro factors.

General graphs and analysis

In [17]:
```
label_list = ['BTC','ADA', 'BNB', 'DOGE', 'DOT', 'ETH', 'LINK', 'L
plist = []

for coin in label_list:
    filepath =folder+f'{coin}USDT_5m_2019-01-01_2023-02-01.csv.xz'
    df = utils.load_and_clean_data(filepath)
    df.rename(columns = {'Close' : f'{coin} Close Price',
                        'Volume' : f'{coin} Volume'}, inplace = T
    df_2019 = df.loc[:'2019-12-31'].copy()
    plist.append(df[[f'{coin} Close Price', f'{coin} Volume']])

```

In [18]:
```
Crypto_Price = reduce(lambda left, right: pd.merge(left,right,left
Crypto_Price
```

Out[18]:

	BTC Close Price	BTC Volume	ADA Close Price	ADA Volume	BNB Close Price	BNB Volume	DOGE Close Price	DOGE Volume
Time								
2019-01-01 00:00:00	3696.32	85.572181	0.04034	312256.3	6.1002	6778.020	NaN	NaN
2019-01-01 00:05:00	3692.34	62.296581	0.04030	603576.0	6.0999	3911.350	NaN	NaN
2019-01-01 00:10:00	3697.31	43.105333	0.04022	467300.9	6.1050	9809.520	NaN	NaN
2019-01-01 00:15:00	3693.00	48.551084	0.04011	295711.0	6.0790	6634.950	NaN	NaN
2019-01-01 00:20:00	3692.18	47.706443	0.04009	260797.1	6.0649	5590.580	NaN	NaN
...	...	...	...	...	...	...	...	...
2023-02-02 00:55:00	24189.85	1394.042710	0.40480	660590.7	322.2000	2719.618	0.09568	11166439.0
2023-02-02 01:00:00	24132.98	1542.742520	0.40240	1332611.3	321.4000	4186.234	0.09515	11096546.0
2023-02-02 01:05:00	24093.79	1979.277790	0.40150	1429480.2	320.8000	2080.526	0.09511	12611147.0
2023-02-02 01:10:00	24063.89	1318.924240	0.40110	750456.1	320.2000	1467.380	0.09523	4961271.0
2023-02-02 01:15:00	24096.10	1499.251360	0.40180	690697.2	321.2000	2713.905	0.09519	8939882.0

429198 rows × 20 columns

In [19]:
```
downsampled = Crypto_Price.resample('1H').first()[:'2021-01-01']
```

In [20]:
```
fig = make_subplots(rows=5, cols=2, shared_xaxes=True, vertical_sp
                    subplot_titles=[f"{coin} Close Price" for coir
                    )
fig.update_layout(height=1000, width=1000, title={ "text": "Select
fig.update_xaxes(title_text='Date')
fig.update_yaxes(title_text='Price in USD')
for i, coin in enumerate(label_list):
    fig.add_trace(go.Scatter(x=downsampled.index, y=downsampled[f'
fig.show()
```

In [21]:
```
downsampled_return = Crypto_Returns.resample('1H').first()[:'2021-
```

In [22]:
```
fig = make_subplots(rows=5, cols=2, shared_xaxes=True, vertical_sp
                    subplot_titles=[f"{coin} Close Price" for coin
                    )
fig.update_layout(height=1000, width=1000, title={ "text": "Select
fig.update_xaxes(title_text='Date')
fig.update_yaxes(title_text='Price in USD')
for i, coin in enumerate(label_list):
    fig.add_trace(go.Scatter(x=downsampled_return.index, y=downsam
fig.show()
```

Discussion:

- In the cryptocurrencies' close price graphs, we have a slightly zoom-in version of the graph (the second one) since BTC's price is way higher than other cryptocurrencies in terms of USD. Comparing the first graph and the second one, we could say that the other cryptocurrencies kinda following the BTC's trend.
- The hourly return graph won't tell much about the relationships between those currencies. However, we could spot some outliners easily from the hourly return graph, such as MATIC and LINK.

4. Pre-step Before Simulation

Right now, We collected ten cryptocurrencies data until Jan. 2020. However, the closed price for DOT and SOL won't be available until the beginning of 2021. Therefore, we didn't put these two currencies under consideration for now. We pick the dataframe since Aug 2019 to compute the other 7 pairs of cryptos' ssd and compare their sum of Squared Deviations to selected 5 pairs which minimize the sum of Squared Deviations.

- The rationale behind this is that pairs with smaller SSD are likely to have a more stable and predictable relationship, which can provide a better foundation for executing a profitable trading strategy.
- When two assets have a stable and predictable relationship, it becomes easier to identify when the relationship deviates from the norm, and when it is likely to return to its usual pattern. This allows pair traders to take advantage of the expected mean reversion by buying the underperforming asset and selling the overperforming asset.
- Furthermore, pairs with smaller SSD may also have lower risk and higher liquidity, as they are less likely to experience sudden and large price movements that can lead to losses. This can make it easier to manage risk and execute trades with a higher degree of confidence.
- Recall the SSD formula:

$$SSD_{ij} = \sum_t (P_{it} - P_{jt})^2$$

In [23]:
```
Crypto_Price_since_2019_8 = Crypto_Price.loc['2019-08-01':'2020-01
Crypto_Price_since_2019_8
```

Out[23]:

Time	BTC Close Price	BTC Volume	ADA Close Price	ADA Volume	BNB Close Price	BNB Volume	DOGE Close Price	DOGE Volume	E Clo Pr
2019-08-01 00:00:00	10095.98	323.926682	0.06002	351559.1	27.7208	10308.56	0.002861	85437.0	218
2019-08-01 00:05:00	10112.75	615.980183	0.06012	1394932.6	27.7919	13035.49	0.002868	364102.0	218
2019-08-01 00:10:00	10092.59	224.986658	0.05988	295469.1	27.6948	7669.26	0.002861	5263.0	217
2019-08-01 00:15:00	10082.57	236.746094	0.05986	349744.0	27.7091	5617.17	0.002861	0.0	217
2019-08-01 00:20:00	10067.79	100.909774	0.05979	234422.4	27.6887	1710.79	0.002867	64892.0	217
...	...	...	...	...	...	...	...	...	
2020-01-01 23:35:00	7190.50	28.946975	0.03341	23926.1	13.6740	2111.99	0.002024	0.0	130
2020-01-01 23:40:00	7193.80	43.722730	0.03347	56287.7	13.6858	1768.71	0.002024	0.0	130
2020-01-01 23:45:00	7198.30	27.344353	0.03347	897.6	13.6849	1394.38	0.002024	8989.0	130
2020-01-01 23:50:00	7201.00	41.540170	0.03350	49634.7	13.6890	10343.37	0.002024	16781.0	130
2020-01-01 23:55:00	7200.85	37.603650	0.03348	11833.0	13.7184	3605.16	0.002024	0.0	130

44204 rows × 16 columns

In [24]:
```
eth_ssd = utils.compute_ssd(Crypto_Price_since_2019_8,'eth', use_l
ada_ssd = utils.compute_ssd(Crypto_Price_since_2019_8,'ada', use_l
bnb_ssd = utils.compute_ssd(Crypto_Price_since_2019_8,'bnb', use_l
matic_ssd = utils.compute_ssd(Crypto_Price_since_2019_8,'matic', u
doge_ssd = utils.compute_ssd(Crypto_Price_since_2019_8,'doge', use
#sol_ssd = computessd(Crypto_Price_since_2019_8,'sol')=, use_log=T
#dot_ssd = computessd(Crypto_Price_since_2019_8,'dot, use_log=True
ltc_ssd = utils.compute_ssd(Crypto_Price_since_2019_8,'ltc', use_l
link_ssd = utils.compute_ssd(Crypto_Price_since_2019_8,'link', use
```

In [25]:
```
ssds = {
    'ETH/BTC':eth_ssd,
    'ADA/BTC':ada_ssd,
    'BNB/BTC':bnb_ssd,
    'MATIC/BTC':matic_ssd,
    'DOGE/BTC':doge_ssd,
    #'SOL/BTC':sol_ssd,
    #'DOT/BTC':dot_ssd,
    'LTC/BTC':ltc_ssd,
    'LINK/BTC':link_ssd,
}
ssd_pd = pd.DataFrame(ssds.items(), columns=['Pair', 'SSD'])
ssd_pd = ssd_pd.set_index('Pair').sort_values(by = 'SSD')
ssd_pd
```

Out[25]:

	SSD
Pair	
LTC/BTC	3847.395609
BNB/BTC	6011.913598
ADA/BTC	6988.220252
ETH/BTC	12479.299030
DOGE/BTC	18634.689689
LINK/BTC	84867.498896
MATIC/BTC	117571.472436

By comparing the SSD we calculated, we can tell during this period, five pairs formed by LTC/BTC, BNB/BTC, ETH/BTC, ADA/BTC and DOGE/BTC have minimized sum of Squared Deviations. At this point, we would like use these five pairs to create the trading portfolio.

In [26]:
```
utils.adf_cointegration_test(Crypto_Price_since_2019_8, 'LTC', 'BT
```

```
Series are not individually stationary. Cointegration test may not b
e valid.
Cointegration test passed. The series are cointegrated.
```

Out[26]:
```
array([2.58776246e-06, 5.20234412e-01])
```

In [27]:
```
utils.adf_cointegration_test(Crypto_Price_since_2019_8, 'BNB', 'BT
```

```
Series are not individually stationary. Cointegration test may not b
e valid.
Cointegration test passed. The series are cointegrated.
```

Out[27]:
```
array([5.46944362e-07, 5.14773455e-01])
```

In [28]:
```
utils.adf_cointegration_test(Crypto_Price_since_2019_8, 'ADA', 'BT
```

```
Series are not individually stationary. Cointegration test may not b
e valid.
Cointegration test passed. The series are cointegrated.
```

Out[28]:
```
array([-1.52147110e-06,  4.63715847e-01])
```

In [29]:
```
utils.adf_cointegration_test(Crypto_Price_since_2019_8, 'ETH', 'BT
```

```
Series are not individually stationary. Cointegration test may not b
e valid.
Cointegration test passed. The series are cointegrated.
```

Out[29]:
```
array([3.00939768e-07, 6.85196042e-01])
```

In [30]:
```
utils.adf_cointegration_test(Crypto_Price_since_2019_8, 'DOGE', 'E
```

```
Series are not individually stationary. Cointegration test may not b
e valid.
Cointegration test passed. The series are cointegrated.
```

Out[30]:
```
array([-5.69245786e-06,  2.49497642e-01])
```

Graph of Pair LTC/BTC

In [125]:
```
LTC_BTC = Crypto_Returns[['BTC 5-Min Return', 'LTC 5-Min Return']]
LTC_BTC['BTC CumSum'] = LTC_BTC['BTC 5-Min Return'].cumsum()
LTC_BTC['LTC CumSum'] = LTC_BTC['LTC 5-Min Return'].cumsum()
LTC_BTC = LTC_BTC.loc['2019-08-01':]
LTC_BTC = LTC_BTC[['BTC CumSum', 'LTC CumSum']]
fig = px.line(LTC_BTC, x=LTC_BTC.index, y=LTC_BTC.columns, labels=
fig.show()
```

Graph of Pair BNB/BTC

In [126]:
```
BNB_BTC = Crypto_Returns[['BTC 5-Min Return', 'BNB 5-Min Return']]
BNB_BTC['BTC CumSum'] = BNB_BTC['BTC 5-Min Return'].cumsum()
BNB_BTC['BNB CumSum'] = BNB_BTC['BNB 5-Min Return'].cumsum()
BNB_BTC = BNB_BTC.loc['2019-08-01':]
BNB_BTC = BNB_BTC[['BTC CumSum', 'BNB CumSum']]
fig = px.line(BNB_BTC, x=BNB_BTC.index, y=BNB_BTC.columns, labels=
fig.show()
```

Graph of Pair ETH/BTC

In [127]:
```
ETH_BTC = Crypto_Returns[['BTC 5-Min Return', 'ETH 5-Min Return']]
ETH_BTC['BTC CumSum'] = ETH_BTC['BTC 5-Min Return'].cumsum()
ETH_BTC['ETH CumSum'] = ETH_BTC['ETH 5-Min Return'].cumsum()
ETH_BTC = ETH_BTC.loc['2019-08-01':]
ETH_BTC = ETH_BTC[['BTC CumSum', 'ETH CumSum']]
fig = px.line(ETH_BTC, x=ETH_BTC.index, y=ETH_BTC.columns, labels=
fig.show()
```

Graph of Pair ADA/BTC

In [128]:
```
ADA_BTC = Crypto_Returns[['BTC 5-Min Return', 'ADA 5-Min Return']]
ADA_BTC['BTC CumSum'] = ADA_BTC['BTC 5-Min Return'].cumsum()
ADA_BTC['ADA CumSum'] = ADA_BTC['ADA 5-Min Return'].cumsum()
ADA_BTC = ADA_BTC.loc['2019-08-01':]
ADA_BTC = ADA_BTC[['BTC CumSum', 'ADA CumSum']]
fig = px.line(ADA_BTC, x=ADA_BTC.index, y=ADA_BTC.columns, labels=
fig.show()
```

Graph of Pair DOGE/BTC

In [129]:
```
DOGE_BTC = Crypto_Returns[['BTC 5-Min Return', 'DOGE 5-Min Return'
DOGE_BTC['BTC CumSum'] = DOGE_BTC['BTC 5-Min Return'].cumsum()
DOGE_BTC['DOGE CumSum'] = DOGE_BTC['DOGE 5-Min Return'].cumsum()
DOGE_BTC = DOGE_BTC.loc['2019-08-01':]
DOGE_BTC = DOGE_BTC[['BTC CumSum', 'DOGE CumSum']]
fig = px.line(DOGE_BTC, x=DOGE_BTC.index, y=DOGE_BTC.columns, labe
fig.show()
```

Discussion:

- From the above five graphs, we are able to conclude that the pairs we have selected are closely related in terms of moving trend on cumulative returns. Trading signals will be explored in the next steps and more stats will be displayed later.

5. Rolling Window and Optimal Stop Loss

- After we decided the pairs for each trading periods, we will further investigate the correlation between those pairs and optimize their weights in our portfolio. Then, we will run each pair with distance method pair trading strategy where long signal will be created when price exceed the 2 standard deviation and short signal will be created when price is down below -2 standard deviation. The position will be closed while the price is close to the mean of normalized prices.

- Here, for all rolling window and stop loss discussion, we will use the data from 2019 as

In [36]:
```
Crypto_Price_2019 = Crypto_Price.loc[:'2019-12-31'].copy()
```

Rolling Window

Calculating the rolling average on BTC / LTC pair ratio. With three different rolling windows, we could see the graph displayed below.

In [37]:
```
PR_LTC = pd.DataFrame(Crypto_Price_2019['BTC Close Price']/Crypto_
```

In [38]:
```
PR_LTC['5h window'] = PR_LTC['BTC vs LTC'].rolling(60).mean()
PR_LTC['20h window'] = PR_LTC['BTC vs LTC'].rolling(240).mean()
PR_LTC['500h window'] = PR_LTC['BTC vs LTC'].rolling(6000).mean()
fig = px.line(PR_LTC, x=PR_LTC.index, y=PR_LTC.columns, labels={"v
fig.show()
```

In [39]:
```
PR_LTC.dropna()
```

Out[39]:

Time	BTC vs LTC	5h window	20h window	500h window
2019-01-21 19:55:00	115.095238	115.041213	114.961957	113.230024
2019-01-21 20:00:00	114.967080	115.040681	114.960111	113.228554
2019-01-21 20:05:00	114.892869	115.042390	114.957710	113.227073
2019-01-21 20:10:00	114.974267	115.044232	114.956333	113.225592
2019-01-21 20:15:00	115.030975	115.041470	114.954684	113.224130
...	...	...	...	...
2019-12-31 23:35:00	174.390398	174.065091	172.196494	173.381127
2019-12-31 23:40:00	174.476964	174.079284	172.208646	173.382863
2019-12-31 23:45:00	174.039429	174.081812	172.218933	173.384555
2019-12-31 23:50:00	174.136572	174.083112	172.230047	173.386248
2019-12-31 23:55:00	174.218644	174.083208	172.241803	173.387979

98769 rows × 4 columns

In [40]:
```
test = PR_LTC[['BTC vs LTC','5h window']]
test['5h std'] = test['BTC vs LTC'].rolling(5).std()
test['signal1'] = np.where(PR_LTC['BTC vs LTC']<PR_LTC['5h window'
test['signal2'] = np.where(PR_LTC['BTC vs LTC']>PR_LTC['5h window'
test['signal'] = test['signal1'] + test['signal2']
test['position'] = test['signal'].cumsum()
test
```

Out[40]:

Time	BTC vs LTC	5h window	5h std	signal1	signal2	signal	position
2019-01-01 00:00:00	123.788346	NaN	NaN	0	0	0	0
2019-01-01 00:05:00	123.779417	NaN	NaN	0	0	0	0
2019-01-01 00:10:00	123.862982	NaN	NaN	0	0	0	0
2019-01-01 00:15:00	123.801542	NaN	NaN	0	0	0	0
2019-01-01 00:20:00	123.774053	NaN	0.036034	0	0	0	0
...	...	...	...	...	...	...	...
2019-12-31 23:35:00	174.390398	174.065091	0.215461	0	-1	-1	-5835
2019-12-31 23:40:00	174.476964	174.079284	0.235710	0	-1	-1	-5836
2019-12-31 23:45:00	174.039429	174.081812	0.188441	1	0	1	-5835
2019-12-31 23:50:00	174.136572	174.083112	0.181837	0	-1	-1	-5836
2019-12-31 23:55:00	174.218644	174.083208	0.179850	0	-1	-1	-5837

104768 rows × 7 columns

Strategy level risk control using time-based stop loss

- Since the strategy fundamentally adopts the mean-reversion of the spread, there is a statistically robust way to determine an optimal holding period that does not depend on the actual trades. According to E. Chan, the mean reversion of a time series can be modelled by an Ornstein-Uhlenbeck process.
- Given a time series of the daily spread values, we can perform a linear regression of the daily change in the spread against the spread itself. The slope of the regression line is the mean reversion speed of the spread. The optimal holding period is then given by the reciprocal of the mean reversion speed multiplied by a ln(2).
- In our simulation system, if any of our positions has not been closed after the optimal holding period, we will close it at the end of the day. This is to prevent the strategy from holding positions for too long and incurring large losses.
- Reference: E. P. Chan, *Quantitative Trading How to Build Your Own Algorithmic Trading Business*, (pp 140-142), Wiley, 2008.

Compute the optimal holding duration for all spreads

In [41]:
```
ltc_duration = Crypto_Price_2019['BTC Close Price']/Crypto_Price_2
bnb_duration = Crypto_Price_2019['BTC Close Price']/Crypto_Price_2
ada_duration = Crypto_Price_2019['BTC Close Price']/Crypto_Price_2
eth_duration = Crypto_Price_2019['BTC Close Price']/Crypto_Price_2
#Crypto_Price_2019['BTC Close Price']/Crypto_Price_2019['DOGE Clos
```

In [42]:
```
Crypto_Price2020 = Crypto_Price.loc['2019-08-01':'2020-07-01'].cop
doge_duration = Crypto_Price2020['BTC Close Price']/Crypto_Price20
```

In [43]:
```
doge_duration
```

Out[43]:
```
Time
2019-08-01 00:00:00    3.528459e+06
2019-08-01 00:05:00    3.525695e+06
2019-08-01 00:10:00    3.527767e+06
2019-08-01 00:15:00    3.524265e+06
2019-08-01 00:20:00    3.511856e+06
                           ...
2020-07-01 23:35:00    3.982232e+06
2020-07-01 23:40:00    3.980133e+06
2020-07-01 23:45:00    3.969881e+06
2020-07-01 23:50:00    3.976469e+06
2020-07-01 23:55:00    3.978796e+06
Length: 96441, dtype: float64
```

In [44]:
```python
ltc_halflife = utils.optimal_holding_duration(ltc_duration)
bnb_halflife = utils.optimal_holding_duration(bnb_duration)
ada_halflife = utils.optimal_holding_duration(ada_duration)
wth_halflife = utils.optimal_holding_duration(eth_duration)
doge_halflife = utils.optimal_holding_duration(doge_duration)
print('the optimal holding duration for LTC is', ltc_halflife)
print('the optimal holding duration for BNB is', bnb_halflife)
print('the optimal holding duration for ADA is', ada_halflife)
print('the optimal holding duration for ETH is', wth_halflife)
print('the optimal holding duration for DOGE is', doge_halflife)
```

```
the optimal holding duration for LTC is 73127.51128517912
the optimal holding duration for BNB is 14122.92688314639
the optimal holding duration for ADA is 52566.38993888397
the optimal holding duration for ETH is 38651.815446806635
the optimal holding duration for DOGE is 861.109581698344
```

6. Simulation and Analysis

- Depending on the exchange, account privileges, traded volume, and trading style (taker vs maker), the typical transaction cost of a crypto-crypto trade is around 1 to 4 bps. We will assume an optimisitc 1 bps transaction cost for our simulation.

In [45]:
```
Crypto_Price_2019
```

Out[45]:

Time	BTC Close Price	BTC Volume	ADA Close Price	ADA Volume	BNB Close Price	BNB Volume	DOGE Close Price	DOGE Volume	DOT Close Price
2019-01-01 00:00:00	3696.32	85.572181	0.04034	312256.3	6.1002	6778.02	NaN	NaN	NaN
2019-01-01 00:05:00	3692.34	62.296581	0.04030	603576.0	6.0999	3911.35	NaN	NaN	NaN
2019-01-01 00:10:00	3697.31	43.105333	0.04022	467300.9	6.1050	9809.52	NaN	NaN	NaN
2019-01-01 00:15:00	3693.00	48.551084	0.04011	295711.0	6.0790	6634.95	NaN	NaN	NaN
2019-01-01 00:20:00	3692.18	47.706443	0.04009	260797.1	6.0649	5590.58	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
2019-12-31 23:35:00	7191.86	34.360944	0.03287	1979674.1	13.7051	150.33	0.002014	5100.0	NaN
2019-12-31 23:40:00	7195.43	25.521108	0.03288	54155.8	13.7197	2299.51	0.002014	0.0	NaN
2019-12-31 23:45:00	7194.79	30.479906	0.03285	187612.3	13.7300	1542.77	0.002014	0.0	NaN
2019-12-31 23:50:00	7204.03	33.066805	0.03288	473203.6	13.7425	635.75	0.002014	0.0	NaN
2019-12-31 23:55:00	7195.23	76.038334	0.03285	213247.6	13.7161	2178.86	0.002014	0.0	NaN

104768 rows × 20 columns

In [46]:
```
simulated_pnl = utils.simulation(Crypto_Price_2019, 'LTC', 500, 2,
print(f'The simulated Pnl over the period is {simulated_pnl.iloc[-
```

```
The simulated Pnl over the period is 186869.51219698388 USD
```

In [48]:
```
simulated_pnl[simulated_pnl['slippage']>0].head()
```

Out[48]:

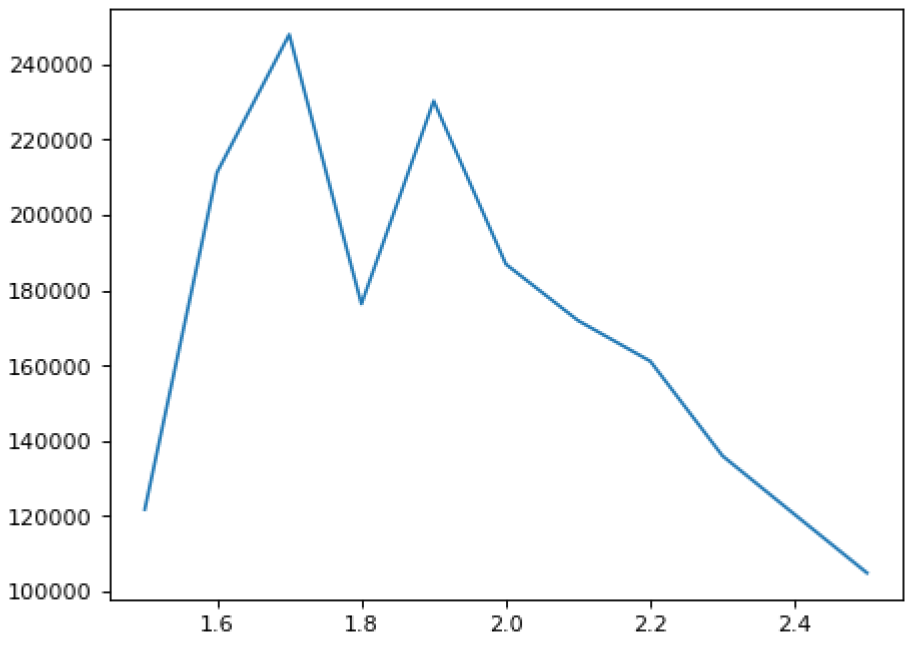
	BTC vs LTC	BTC Close Price	BTC Volume	LTC Volume	LTC Close Price	500h mean	500h std	signal	B
Time									
2019-01-04 06:30:00	116.881737	3795.15	118.601200	4972.50539	32.47	119.707142	1.325185	1	13
2019-01-04 06:35:00	116.869565	3790.08	133.636688	1191.20655	32.43	119.695548	1.334830	1	13
2019-01-04 06:40:00	117.481390	3787.60	88.485985	827.93280	32.24	119.686503	1.339582	0	
2019-01-05 00:40:00	117.008615	3802.78	198.964720	1157.75001	32.50	119.336058	1.084523	1	13
2019-01-05 00:45:00	116.846484	3805.69	102.060805	1878.88379	32.57	119.326024	1.093792	1	13

5 rows × 23 columns

In [132]:
```
tmp_df = pd.DataFrame()
ts = [1.5,1.6,1.7,1.8,1.9,2,2.1,2.2,2.3,2.4,2.5]
for t in ts:
    tmp_df[t] = utils.simulation(Crypto_Price_2019, 'LTC', 500, t,
```

In [133]:
```
tmp_df.iloc[-1].plot()
```

Out[133]:
```
<Axes: >
```



In [51]:
```
diff_df = tmp_df.resample('500min').first().diff()
```

In [52]:
```
max_columns = diff_df.idxmax(axis=1)
```

In [53]:
```
temp = diff_df.shift(-1)
```

In [54]:
```
max_columns = max_columns.dropna()
optimized_pnl = []
for index, value in max_columns.items():
    optimized_pnl.append(temp.loc[index,value])
k = np.array(optimized_pnl)
```

In [55]:
```
pd.DataFrame(k,index = max_columns.index).dropna().cumsum()
```

Out[55]:

	0
Time	
2019-01-01 08:20:00	0.000000
2019-01-01 16:40:00	0.000000
2019-01-02 01:00:00	0.000000
2019-01-02 09:20:00	0.000000
2019-01-02 17:40:00	0.000000
...	...
2019-12-30 04:40:00	-22474.188025
2019-12-30 13:00:00	-22474.188025
2019-12-30 21:20:00	-22474.188025
2019-12-31 05:40:00	-22474.188025
2019-12-31 14:00:00	-21608.579885

1050 rows × 1 columns

In [56]:
```
fig = px.line(simulated_pnl, x=simulated_pnl.index, y=simulated_pr
fig.update_layout(title={'x':0.5,'xanchor': 'center'})
fig.show()
```

In [57]:
```
simulated_pnl
```

Out[57]:

Time	BTC vs LTC	BTC Close Price	BTC Volume	LTC Volume	LTC Close Price	500h mean	500h std	signal	B' targ p
2019-01-01 00:00:00	123.788346	3696.32	85.572181	102.33669	29.86	NaN	NaN	0	(
2019-01-01 00:05:00	123.779417	3692.34	62.296581	65.06410	29.83	NaN	NaN	0	(
2019-01-01 00:10:00	123.862982	3697.31	43.105333	86.42782	29.85	NaN	NaN	0	(
2019-01-01 00:15:00	123.801542	3693.00	48.551084	134.54598	29.83	NaN	NaN	0	(
2019-01-01 00:20:00	123.774053	3692.18	47.706443	140.46924	29.83	NaN	NaN	0	(
...	...	...	...	...	...	...	...	...	
2019-12-31 23:35:00	174.390398	7191.86	34.360944	650.53238	41.24	172.073780	1.620232	0	(
2019-12-31 23:40:00	174.476964	7195.43	25.521108	284.41931	41.24	172.083373	1.624093	0	(
2019-12-31 23:45:00	174.039429	7194.79	30.479906	1290.01631	41.34	172.091182	1.625544	0	(
2019-12-31 23:50:00	174.136572	7204.03	33.066805	1663.72562	41.37	172.099347	1.627425	0	(
2019-12-31 23:55:00	174.218644	7195.23	76.038334	415.87013	41.30	172.107807	1.629673	0	(

104768 rows × 23 columns

We will be using the 5 pairs of currencies with minimal SSD, as previously illustrated.

In [58]:
```
ssd_pd.iloc[:5,]
```

Out[58]:

Pair	SSD
LTC/BTC	3847.395609
BNB/BTC	6011.913598
ADA/BTC	6988.220252
ETH/BTC	12479.299030
DOGE/BTC	18634.689689

In [102]:
```
pnl_ltc = utils.simulation(Crypto_Price['2020-07-01':], 'LTC', 50
pnl_bnb = utils.simulation(Crypto_Price['2020-07-01':], 'BNB', 50
pnl_ada = utils.simulation(Crypto_Price['2020-07-01':], 'ADA', 50
pnl_eth = utils.simulation(Crypto_Price['2020-07-01':], 'ETH', 50
pnl_dog = utils.simulation(Crypto_Price['2020-07-01':], 'DOGE', 5
```

In [103]:
```
pnl_dog.loc[pnl_dog['slippage_DOGE'].idxmax():]
```

Out[103]:

	BTC vs DOGE	BTC Close Price	BTC Volume	DOGE Volume	DOGE Close Price	500h mean	500h s
Time							
2020-07-05 00:05:00	3.958863e+06	9138.64	72.821737	124932.0	0.002308	3.932926e+06	15319.4339
2020-07-05 00:10:00	3.957915e+06	9136.45	46.752117	65315.0	0.002308	3.933027e+06	15370.4671
2020-07-05 00:15:00	3.962474e+06	9139.05	76.172583	16915.0	0.002306	3.933146e+06	15452.7546
2020-07-05 00:20:00	3.956420e+06	9133.00	147.595958	617433.0	0.002308	3.933240e+06	15492.0736
2020-07-05 00:25:00	3.955196e+06	9134.13	77.081400	109122.0	0.002309	3.933328e+06	15523.4121
...	...	...	...	...	...	...	
2023-02-02 00:55:00	2.528203e+05	24189.85	1394.042710	11166439.0	0.095680	2.517276e+05	5950.5372
2023-02-02 01:00:00	2.536309e+05	24132.98	1542.742520	11096546.0	0.095150	2.517352e+05	5939.8630
2023-02-02 01:05:00	2.533255e+05	24093.79	1979.277790	12611147.0	0.095110	2.517415e+05	5928.8450
2023-02-02 01:10:00	2.526923e+05	24063.89	1318.924240	4961271.0	0.095230	2.517453e+05	5917.3034
2023-02-02 01:15:00	2.531369e+05	24096.10	1499.251360	8939882.0	0.095190	2.517509e+05	5906.1338

271040 rows × 23 columns

In [104]:
```
# Merge returns of traded cryptocurrency pairs
pnls = pd.concat([pnl_ltc[['PNL']], pnl_bnb[['PNL']], pnl_ada[['PN
pnls.tail()
```

Out[104]:

	PNL	PNL	PNL	PNL	PNL
Time					
2023-02-02 00:55:00	5.123936e+06	-184022.347164	4.638352e+06	-847662.763707	6.202128e+06
2023-02-02 01:00:00	5.123936e+06	-184022.347164	4.638352e+06	-843822.201836	6.202128e+06
2023-02-02 01:05:00	5.123936e+06	-184022.347164	4.638352e+06	-845364.396319	6.202128e+06
2023-02-02 01:10:00	5.123936e+06	-184022.347164	4.638352e+06	-846852.306095	6.202128e+06
2023-02-02 01:15:00	5.123936e+06	-184022.347164	4.638352e+06	-843457.584186	6.202128e+06

In [105]:
```
downsampled_pnl_ltc = pnl_ltc.resample('2H').first()
downsampled_pnl_bnb = pnl_bnb.resample('2H').first()
downsampled_pnl_ada = pnl_ada.resample('2H').first()
downsampled_pnl_eth = pnl_eth.resample('2H').first()
downsampled_pnl_dog = pnl_dog.resample('2H').first()
```

In [143]:
```
lst = ['LTC','BNB','ADA','ETH','DOG']
```

In [146]:
```
fig = make_subplots(rows=3, cols=2, shared_xaxes=True, vertical_sp
                    subplot_titles=[f"{coin} Cumulative PnL" for c
                    )
fig.update_layout(height=900, width=900, title={ "text": "Cumulati
fig.update_xaxes(title_text='Date')
fig.update_yaxes(title_text='PnL in USD')

fig.add_trace(go.Scatter(x=downsampled_pnl_ltc.index, y=downsample
fig.add_trace(go.Scatter(x=downsampled_pnl_bnb.index, y=downsample
fig.add_trace(go.Scatter(x=downsampled_pnl_ada.index, y=downsample
fig.add_trace(go.Scatter(x=downsampled_pnl_eth.index, y=downsample
fig.add_trace(go.Scatter(x=downsampled_pnl_dog.index, y=downsample
fig.show()
```

7. Portfolio Construction: Capital Allocation

- Out of the numerous ways to optimize allocation weights, we chose to determine the weights using the Markowitz portfolio optimization technique. We do not only care about returns, but also risk-adjusted returns, which is why we optimize the portfolio

Sharpe ratio as a way to maximize return per unit of risk. In this case, the Sharpe Ratio is defined as the excess return divided by volatility of returns. These weights that optimize SR are known as the tangency weights.

$$\frac{R_{\text{excess}}}{\sigma_p}$$

- We consider each pair of traded currency to be an asset i, and hold a corresponding position w_i in the asset. These positions construct a portfolio of currency pairs, on which we will optimize by maximizing the portfoio Sharpe Ratio. We will find out a vector that varies by time, $\mathbf{w}_t$, of optimal weights that gives us this solution. (Note that this is only considered optimal based on the benchmark of Sharpe Ratio, which is not necessariliy optimal in the global context but certainly a way to improve our capital allocation).
- Each week is a session of 1440 timestamps. In each week, we will compute the optimal tangency weights based on last week's data. We will then implement the tagency weights in the next week to avoid lookahead bias. Therefore, we are always using the previous optimal weights on a new unseen week of data. In the first week, we begin with the default weights [0.2, 0.2, 0.2, 0.2, 0.2].

Baseline portfolio

- In the baseline scenario, we assume equal cash alloation for the 5-strategies that are being traded. This is equivalent to a 20% allocation for each strategy. We will compare the performance of this baseline portfolio to the optimized portfolio.

In [107]:
```python
pnls['baseline'] = pnls.sum(axis=1) * 0.2
```

In [139]:
```
fig = px.line(pnls, x=pnls.index, y=pnls['baseline'], labels={"y":
fig.update_layout(title={'x':0.5,'xanchor': 'center'})
fig.show()
```

In [109]:
```
#compute returns
pnl_hourly = pnls.resample('H').first()
return_hourly = pnl_hourly.diff() / 10_000_1000
return_hourly.dropna()
```

Out[109]:

	PNL	PNL	PNL	PNL	PNL	baseline
Time						
2020-07-01 05:00:00	0.000000	0.0	0.0	0.000000	0.0	0.000000
2020-07-01 06:00:00	0.000000	0.0	0.0	0.000000	0.0	0.000000
2020-07-01 07:00:00	0.000000	0.0	0.0	0.000000	0.0	0.000000
2020-07-01 08:00:00	0.000000	0.0	0.0	0.000000	0.0	0.000000
2020-07-01 09:00:00	0.000000	0.0	0.0	0.000000	0.0	0.000000
...	...	...	...	...	...	...
2023-02-01 21:00:00	0.000000	0.0	0.0	-0.000134	0.0	-0.000027
2023-02-01 22:00:00	0.000000	0.0	0.0	0.000301	0.0	0.000060
2023-02-01 23:00:00	0.000069	0.0	0.0	0.000073	0.0	0.000028
2023-02-02 00:00:00	0.000000	0.0	0.0	0.000054	0.0	0.000011
2023-02-02 01:00:00	0.000000	0.0	0.0	-0.000373	0.0	-0.000075

22673 rows × 6 columns

Run optimizations on the returns, we project the weights to the subsequent trading in a rolling window fashion.

In [110]:
```
weights, opt_pnl = utils.optimize_weekly_rebalance(pnl_hourly.iloc
```

In [111]:
```
opt_pnl = pd.DataFrame(opt_pnl, index=pnl_hourly.index, columns=['
opt_pnl['cumsum'] = opt_pnl['opt_pnl'].cumsum()
```

We have significantly improved our returns by tangency weights for the first half of the year.

In [138]:
```
fig = px.line(opt_pnl, x=opt_pnl.index, y=opt_pnl['cumsum'], label
fig.update_layout(title={'x':0.5,'xanchor': 'center'})
fig.show()
```

In [113]:
```
### 8. Analysis and Future Discussion
```

In [114]:
```
pnl_hourly = pnls.resample('H').first()
return_hourly = pnl_hourly.diff()
return_hourly.columns = ['LTC Pnl','BNB Pnl','ADA Pnl','ETH Pnl','
return_hourly= return_hourly.dropna()
return_hourly['Optimal Allocation'] = opt_pnl.dropna()['opt_pnl']
sum_stats = utils.performance_summary(return_hourly)
```

In [116]:
```
#compute max drawdown
def MaxDrawdown(nv_list):
    i = np.argmax((np.maximum.accumulate(nv_list) - nv_list) / np.
    if i == 0:
        return 0
    j = np.argmax(nv_list[:i])
    return (nv_list[j] - nv_list[i]) / (nv_list[j])

#merge all stats
def perform_statistics(return_all,rf=0,freq='D'):
    multiply = {'D':250,'W':52,'M':12,'Q':4,'Y':1,'H':6000}
    index_metric = ['Annual Return %', 'Sharpe Ratio', 'Sortino',
                    'semi-variance %', 'VaR(30days) %','Max drawdw
                    '1 year return (rolling) %','3 year return (ro
    result = pd.DataFrame(columns=return_all.columns,index=index_m
    for i in return_all.columns:

        return_s =  return_all[i].dropna()
        return_s.index = pd.DatetimeIndex(return_s.index)
        ret = gmean(return_s.fillna(0)+1)**multiply[freq] -1
        vol = return_s.std() * np.sqrt(multiply[freq])
        sharpe = (ret-rf) / vol
        return_s_monthly=return_s.resample('M').sum()
        odds=return_s_monthly[return_s_monthly>0].count()/return_s

        ret_ts_1y=(return_s+1).map(lambda x:math.log(x)).rolling(m
        ret_ts_3y=(return_s+1).map(lambda x:math.log(x)).rolling(m
        ret_ts_5y=(return_s+1).map(lambda x:math.log(x)).rolling(m
        ret_1y=ret_ts_1y.dropna().mean()
        ret_3y=ret_ts_3y.dropna().mean()
        ret_5y=ret_ts_5y.dropna().mean()
        VaR_1y=ret_ts_1y.dropna().quantile(0.05)


        return_prf = pd.DataFrame()
        rf = 0
        return_prf["ex_ret"] = return_s - rf
        return_prf["neg_ex_ret"] = return_prf["ex_ret"][return_prf
        return_prf["pos_ex_ret"] = return_prf["ex_ret"][return_prf
        return_prf = return_prf.fillna(0)


        semi_var = np.sqrt(np.sum(return_prf["neg_ex_ret"] ** 2) /
        if return_prf["neg_ex_ret"].sum() == 0:
            sortino = np.nan
            omega = np.nan
        else:
            sortino = return_prf["ex_ret"].mean() / semi_var
            omega = return_prf["pos_ex_ret"].sum() / - return_prf[


        rank_ratio = return_s[1:].sort_values(ascending=True)
        var = np.percentile(rank_ratio, 5, interpolation='lower')

        max_drawback = MaxDrawdown((1 + return_s).cumprod())

        result[i] = np.array([ret * 100, sharpe, sortino, omega, v

    return result.round(2)
```

In [136]:
```
return_hourly
```

Out[136]:

	LTC Pnl	BNB Pnl	ADA Pnl	ETH Pnl	DOG Pnl	baseline	Optimal Allocation
Time							
2020-07-01 05:00:00	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.000000
2020-07-01 06:00:00	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.000000
2020-07-01 07:00:00	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.000000
2020-07-01 08:00:00	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.000000
2020-07-01 09:00:00	0.000000	0.0	0.0	0.000000	0.0	0.000000	0.000000
...	...	...	...	...	...	...	...
2023-02-01 21:00:00	0.000000	0.0	0.0	-13442.982288	0.0	-2688.596458	-268.832762
2023-02-01 22:00:00	0.000000	0.0	0.0	30076.210583	0.0	6015.242117	601.464065
2023-02-01 23:00:00	6870.685012	0.0	0.0	7294.591369	0.0	2833.055276	283.277200
2023-02-02 00:00:00	0.000000	0.0	0.0	5400.343878	0.0	1080.068776	107.996078
2023-02-02 01:00:00	0.000000	0.0	0.0	-37314.755118	0.0	-7462.951024	-746.220480

22673 rows × 7 columns

In [137]:
```
perform_statistics(return_hourly[['LTC Pnl']]/10000000,freq = 'H')
```
Out[137]:

	LTC Pnl
Annual Return %	13.77
Sharpe Ratio	1.23
Sortino	0.02
Omega	1.24
Volatility %	11.16
semi-variance %	0.11
VaR(30days) %	0.00
Max drawdwon %	11.17
return/Max drawdwon	1.23
positive rate %	65.62
1 year return (rolling) %	20.35
3 year return (rolling) %	74.68
5 year return (rolling) %	NaN
1 year rolling VaR(5%) %	1.68

In [117]:
```
perform_statistics(return_hourly[['Optimal Allocation']]/10000000,
```
Out[117]:

	Optimal Allocation
Annual Return %	11.35
Sharpe Ratio	3.24
Sortino	0.09
Omega	2.00
Volatility %	3.51
semi-variance %	0.02
VaR(30days) %	-0.09
Max drawdwon %	2.77
return/Max drawdwon	4.09
positive rate %	75.00
1 year return (rolling) %	11.50
3 year return (rolling) %	40.12
5 year return (rolling) %	NaN
1 year rolling VaR(5%) %	3.64

In [81]:
```
sum_stats
```

Out[81]:

	Mean Return	Volatility	Sharpe Ratio	Skewness	Excess Kurtosis	VaR (0.05)	CVa
LTC Pnl	233.366927	14405.933580	0.016199	-11.063121	988.204607	0.000000	-984.0
BNB Pnl	-4.833363	11846.941937	-0.000408	-4.762225	267.268041	0.000000	-947.6
ADA Pnl	204.925296	17255.939121	0.011876	11.485470	998.366710	0.000000	-1228.0
ETH Pnl	-35.461688	8413.052675	-0.004215	0.303664	424.216366	0.000000	-670.4
DOG Pnl	276.224410	34416.075468	0.008026	-7.556259	867.883023	0.000000	-2057.4
baseline	134.844316	10291.615014	0.013102	2.413647	912.994363	-4130.406557	-18582.1
Optimal Allocation	169.795923	4199.817001	0.040429	16.248031	608.542763	-411.326252	-3423.4

Analysis

Returns

- According to the summary table above, most of the pairs have a positive mean return. Since the baseline portfolio uses the equal weights, the return should be a quite reasonable positive number. In general the volatility is quite high though it varies a lot by different cryptocurrencies. From the baseline portfolio to optimal allocation portfolio, an obvious improvement is shown as the volatility decreased dramatically.
- In conclusion, trading the cryptocurrencies in a portfolio might not be a good idea and a better stop loss/hedging method should be discussed in order to lower the volatility. The method we have implied in this project does not work very well, however, the process of getting the simulation done was quite interesting. We have explored a varies of ways to implement the stop loss to the simulation.

Transparency

- The stretgy related to crypto pair trade might face the issue of lack of transparency, which might cause some potential risk on trading performance and PnL:
- Market manipulation: Without transparency,we might face the risk that some market participants may manipulate the market by using insider information or engaging in fraudulent activities.
- Liquidity risks: Lack of transparency can make it difficult to assess the liquidity of a crypto pair, which can increase the risk of price volatility and create difficulties in executing trades. This can be particularly challenging for illiquid crypto pairs, as we choose the biggest crypto in our cases, this issues migth be consider less affected.
- Security risks: In a non-transparent market, it can be difficult to assess the security and reliability of the trading platform or exchange.
- Counterparty risk: Trading crypto pairs involves counterparty risk, which is the risk that the other party in the trade will default or fail to fulfill their obligations. It can be difficult to assess the creditworthiness or reliability of the counterparty, which can increase the risk of default.
- Regulatory risks: It can also create regulatory risks, as regulators may be more likely to view non-transparent markets as risky and may take actions to restrict or regulate them.

Regulation

- The platform that we choose to perform our trading strategy should also be carfully selected. Considering on-chain and off-chain transaction both have their own benefit and drawback, we assume the transaction will be executed through exchange since strategy has involved short position which is only available in limited amount of platforms. Therefore, we also face risk that the crypto exchange brings:
- Cybersecurity: Cryptocurrency exchanges are vulnerable to hacking, cyberattacks, and other security breaches that can result in the theft or loss of funds. This risk can be particularly high for smaller or less established exchanges that may have weaker security systems or controls.
- Regulatory risk: Cryptocurrency exchanges are subject to varying degrees of regulation in different jurisdictions, and there is a risk that new regulations or changes in existing regulations could impact the operations or viability of exchanges. This risk can be particularly high for exchanges that operate in countries with uncertain or evolving regulatory frameworks.
- Operational risk: Trading on a crypto exchange involves operational risks, such as errors or system failures that can result in the loss of funds or other negative impacts. This risk can be particularly high for exchanges that have limited resources or inadequate risk management practices.
- Illegal and unproper decision made by exchange: RIP FTX.

```
In [ ]:
```