Sector Performance Analysis

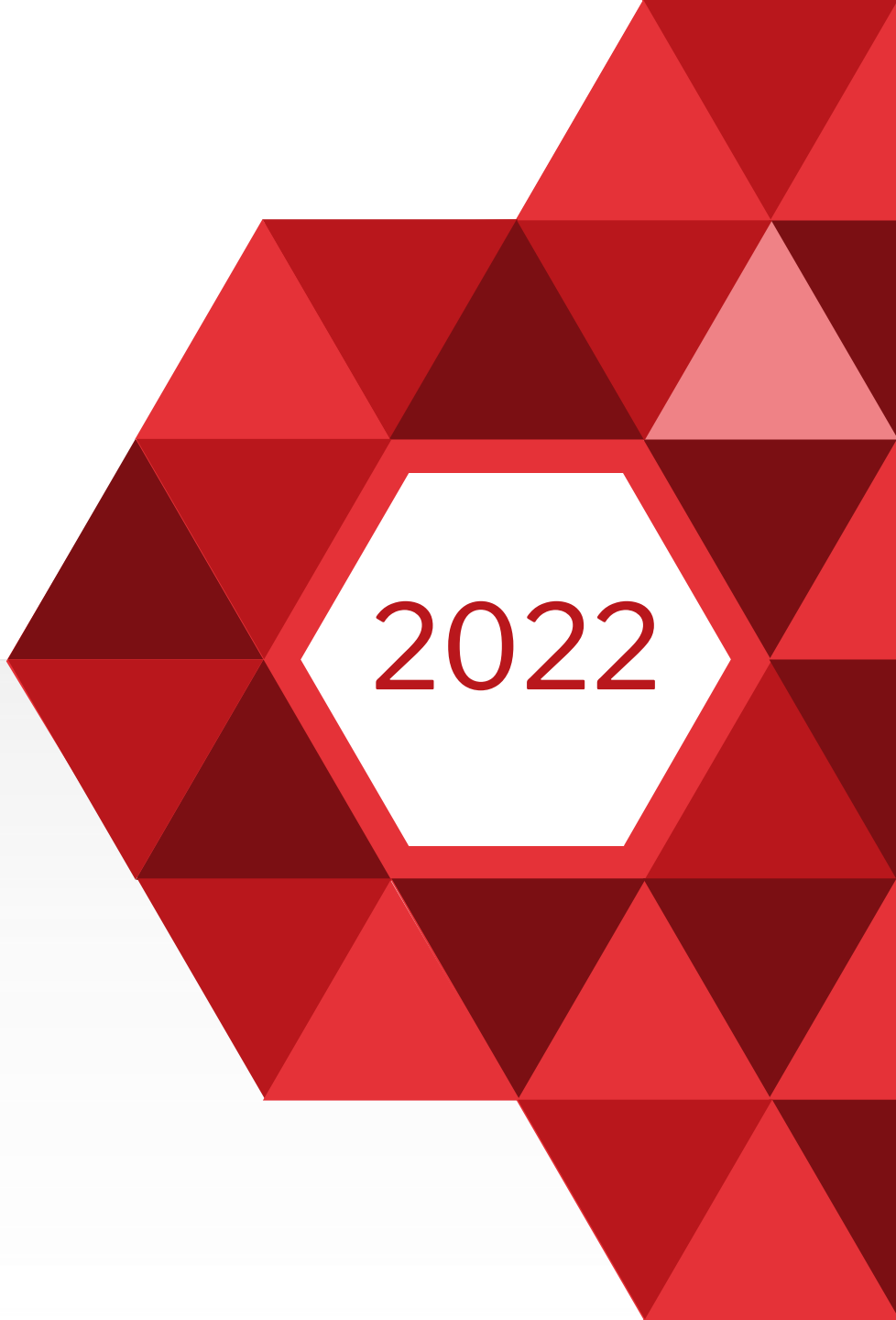
Stocks, Bonds, & Cryptocurrencies

2022

Joseph Hagemann, Preston Kirschner, Emilio Cubero, & Jorge Betancourt

Summary

The ultimate goal of this project is to utilize APIs, SQL, and Data Analytics to compare performance of stock sector ETFs, bond types, and top cryptocurrencies over the last thousand trading days



Comparative analysis within asset classes, then across asset classes

- Which sector stock ETF, bond ETF, and cryptocurrency are most attractive within each asset class?
- Which asset class is most attractive based on aggregation of all ETFs/crypto within each class?

00

IMPORTED LIBRARIES

LIBRARIES AND DEPENDENCIES USED FOR PROJECT

```
# Import libraries and dependencies
import os
import json
import requests
import numpy as np
import pandas as pd
import datetime as dt
import alpaca_trade_api as tradeapi
import plotly.express as px
import hvplot.pandas
import matplotlib.pyplot as plt
from sqlalchemy import create_engine
engine = create_engine('postgresql://postgres:password@localhost:5432/sector_analysis_performance')
from panel.interact import interact
from panel import widgets
import seaborn as steve
```

CONTENT

01. STOCKS — HISTORICAL FINANCIAL ANALYSIS ON SECTOR STOCK ETFS

02. BONDS — HISTORICAL FINANCIAL ANALYSIS ON GOVERNMENT BONDS

03. CRYPTO — HISTORICAL FINANCIAL ANALYSIS ON TOP CRYPTOCURRENCIES

04. ASSET CLASSES — HISTORICAL COMPARATIVE ANALYSIS ON ALL ASSET CLASSES

01

STOCKS

- ENERGY - XLE
- FINANCIAL SERVICES - XLF
- REAL ESTATE - XLRE
- TECHNOLOGY - XLK
- CONSUMER STAPLES - XLP
- BASIC MATERIALS - XLV
- INDUSTRIAL - XLI
- HEALTHCARE - XLB
- UTILITIES - XLU
- CONSUMER DISCRETIONARY - XLY
- COMMUNICATION SERVICES - XLC

01

DATA CLEANING & EXPLORATION

IMPORTING FINANCIAL DATA FOR ALL STOCK ETFs

API CALL

```
# Set timeframe to '1D'
timeframe = "1D"

# Set start and end datetimes
start_date = pd.Timestamp("2018-02-9", tz="America/New_York").isoformat()
end_date = pd.Timestamp("2022-01-28", tz="America/New_York").isoformat()

# Set the ticker information

stock_tickers = ["XLE","XLF","XLRE", "XLK", "XLP", "XLB", "XLI", "XLV", "XLU", "XLY", "XLC"]

# Get historical price data for last 1000 trading days
stock_ticker_df = api.get_barset(
    stock_tickers,
    timeframe,
    start=start_date,
    end=end_date,
    limit=1000,
).df

# Display sample data
stock_ticker_df.head()
```

01

DATA CLEANING & EXPLORATION

IMPORTING FINANCIAL DATA FOR ALL STOCK ETFs

DATA CLEANING

	XLB				
	open	high	low	close	volume
time					
2018-02-09 00:00:00-05:00	58.05	58.81	56.62	58.410	24625943

```
# Remove timestamp from date index
stock_ticker_df['Date'] = stock_ticker_df.index.date
stock_ticker_df.set_index('Date', inplace = True)
stock_ticker_df.index = pd.to_datetime(stock_ticker_df.index)
stock_ticker_df.head()
```

```
# multi-level column deconstruction
stock_ticker_close_df = stock_ticker_df.iloc[:, stock_ticker_df.columns.get_level_values(1)=='close']
stock_ticker_close_df.columns = stock_ticker_close_df.columns.droplevel(1)
# Remove NAs
stock_ticker_close_df.dropna(inplace = True)
```

01

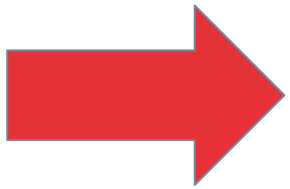
STOCKS CLOSE DATAFRAME

IMPORTING FINANCIAL DATA FOR ALL STOCK ETFs

API CALL

DATA CLEANING

CREATION OF DATAFRAME

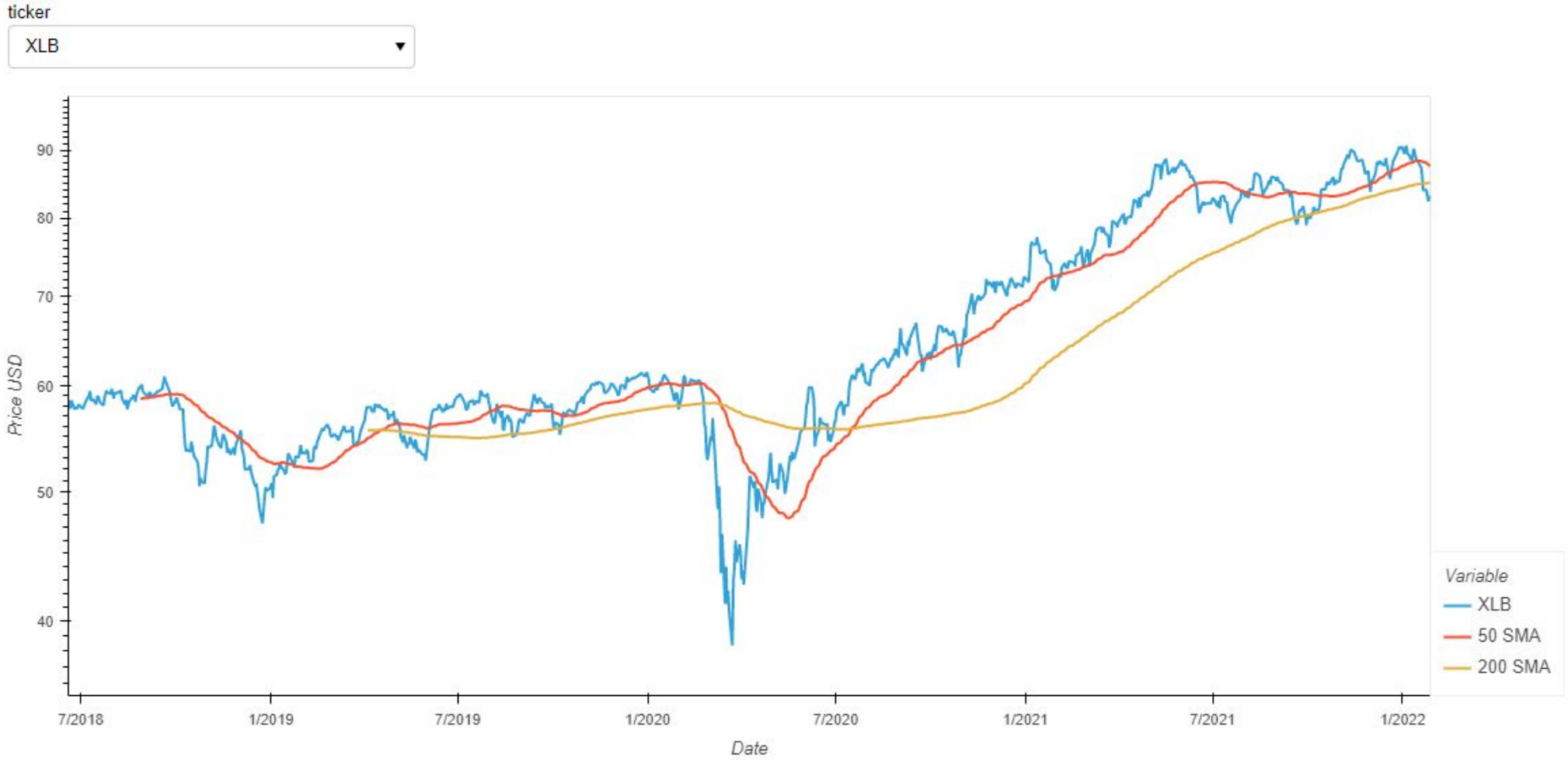


Date	XLB	XLC	XLE	XLF	XLI	XLK	XLP	XLRE	XLU	XLV	XLY
2018-06-19	58.52	49.96	74.73	27.360	73.24	71.40	51.05	31.500	50.22	84.85	111.77
2018-06-20	58.30	50.58	75.08	27.265	73.26	71.58	51.11	31.850	50.23	85.03	112.30
2018-06-21	57.70	50.27	73.62	27.215	72.38	71.02	51.20	32.040	50.40	84.56	111.49
2018-06-22	58.49	50.49	75.21	27.070	72.60	70.80	51.62	32.300	50.75	84.91	111.34
2018-06-25	57.63	49.45	73.56	26.790	71.70	69.34	51.87	32.235	51.60	84.15	108.91
...	...	...	...	...	...	...	...	...	...	...	...
2022-01-24	84.09	72.73	62.86	38.310	101.76	155.11	75.72	46.990	68.18	129.02	182.59
2022-01-25	83.33	71.27	65.30	38.470	100.78	151.61	74.87	46.860	67.11	128.21	179.47
2022-01-26	82.43	70.16	65.18	38.570	99.87	152.42	74.28	46.040	66.56	127.54	178.70
2022-01-27	82.74	70.32	65.94	38.220	99.00	151.43	74.80	45.260	67.14	127.85	174.23
2022-01-28	83.22	72.31	65.63	38.730	99.71	157.98	75.67	46.810	67.92	130.48	178.04

01

200 & 50 DAY MOVING AVERAGE FOR STOCKS

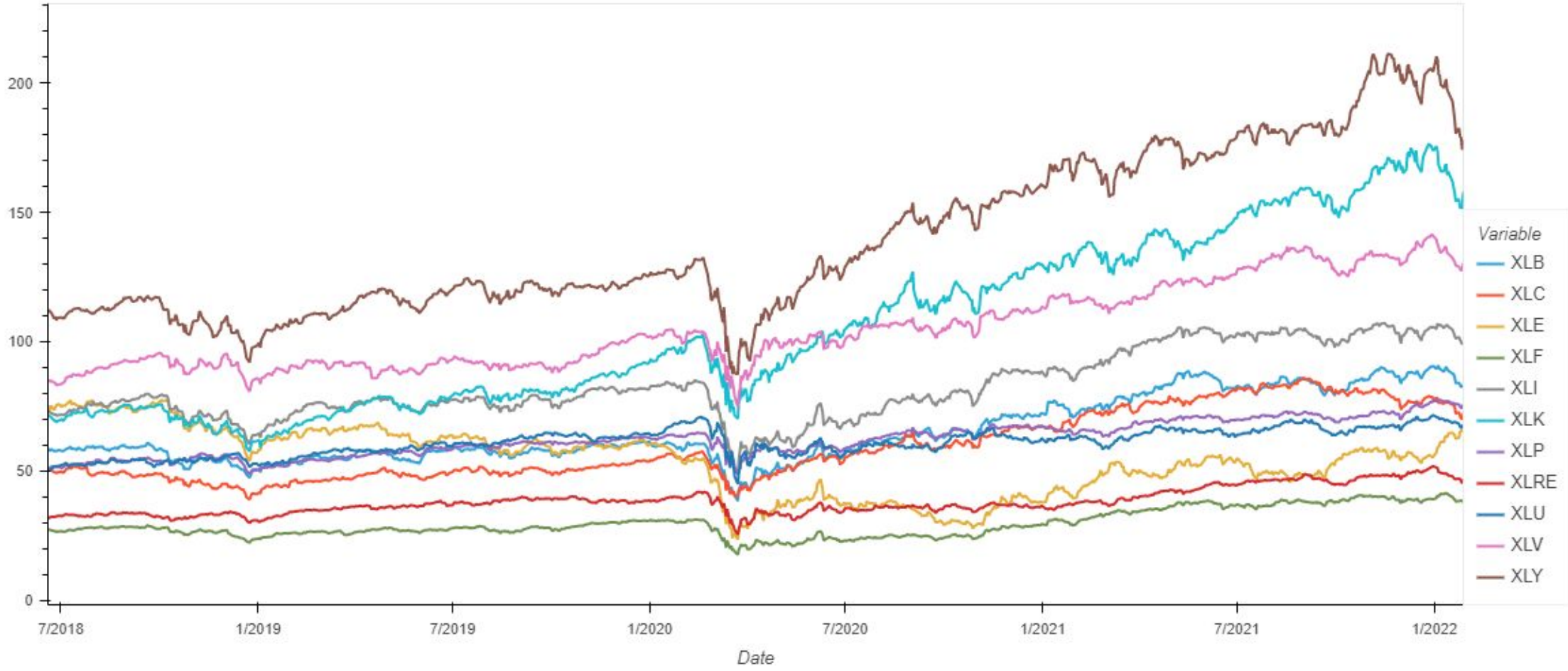
STOCKS CHART CONTAINING CLOSING PRICES AND MOVING AVERAGES WITH DROP DOWN SELECTOR



01

STOCKS CLOSE PRICE PLOT

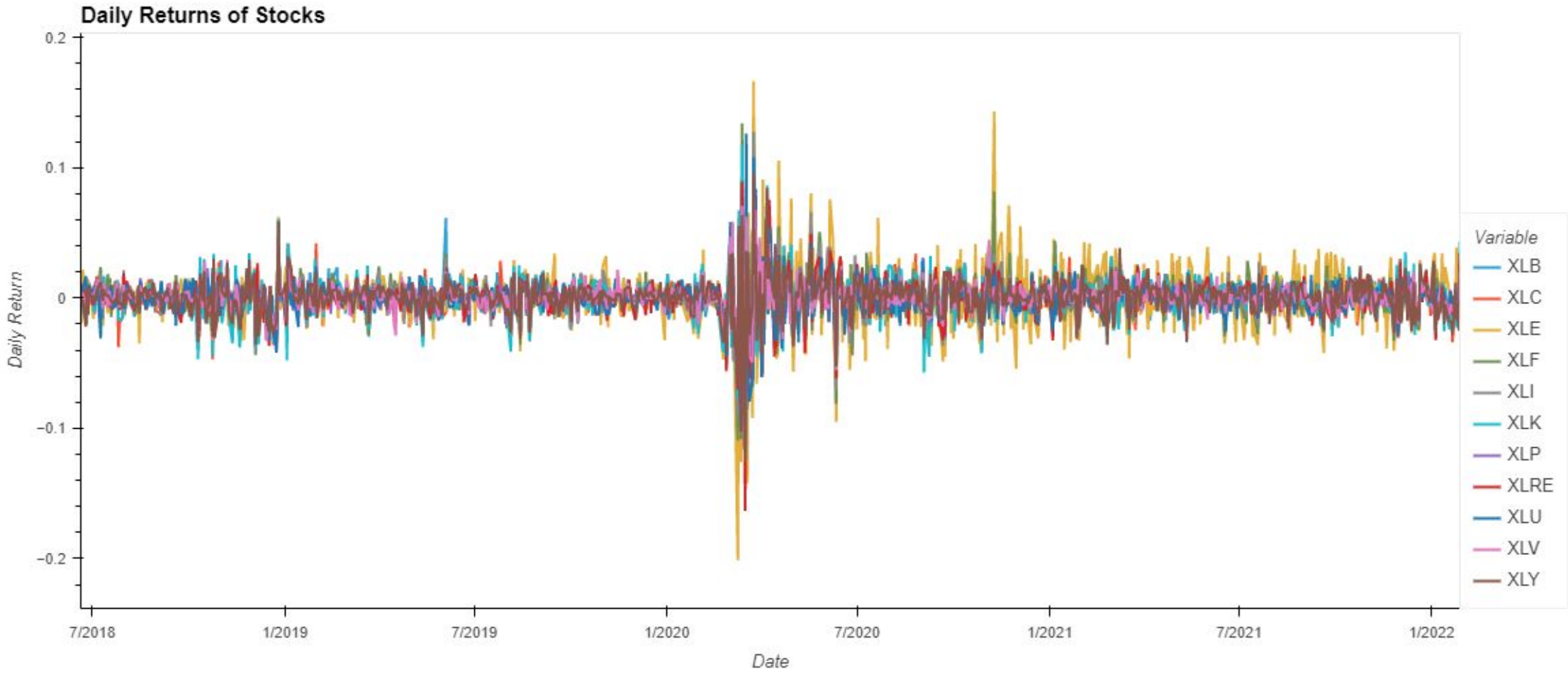
STOCKS CHART CONTAINING CLOSING PRICES FOR ALL STOCK ETFs



01

STOCK DAILY RETURNS

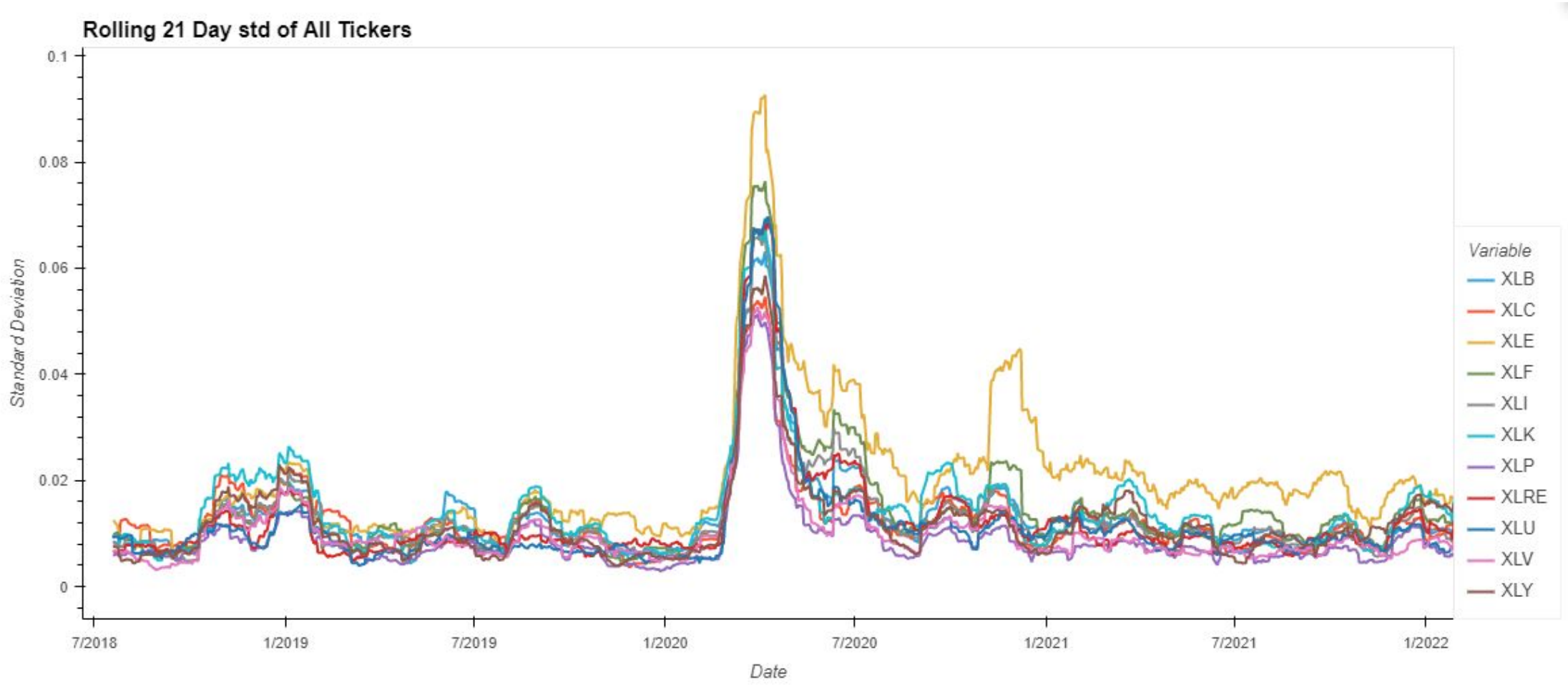
STOCKS CHART CONTAINING DAILY RETURNS FOR ALL STOCK ETF'S



01

STOCK ROLLING 21 STANDARD DEVIATION

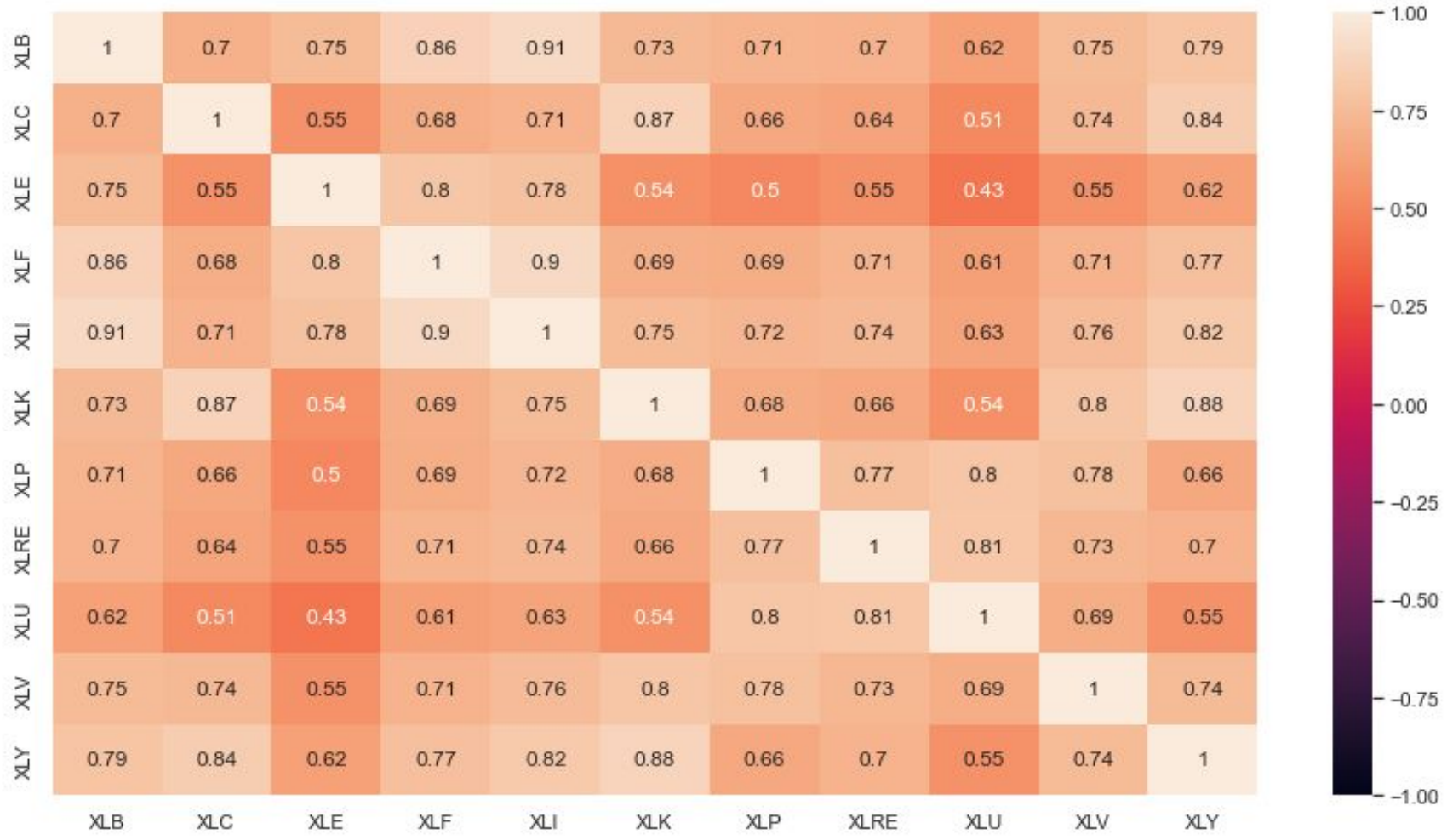
STOCKS CHART CONTAINING ROLLING 21 STANDARD DEVIATION FOR ALL STOCK ETF'S



01

CORRELATION OF STOCKS

STOCKS HEAT MAP CONTAINING GENERAL CORRELATION FOR ALL STOCK ETF'S

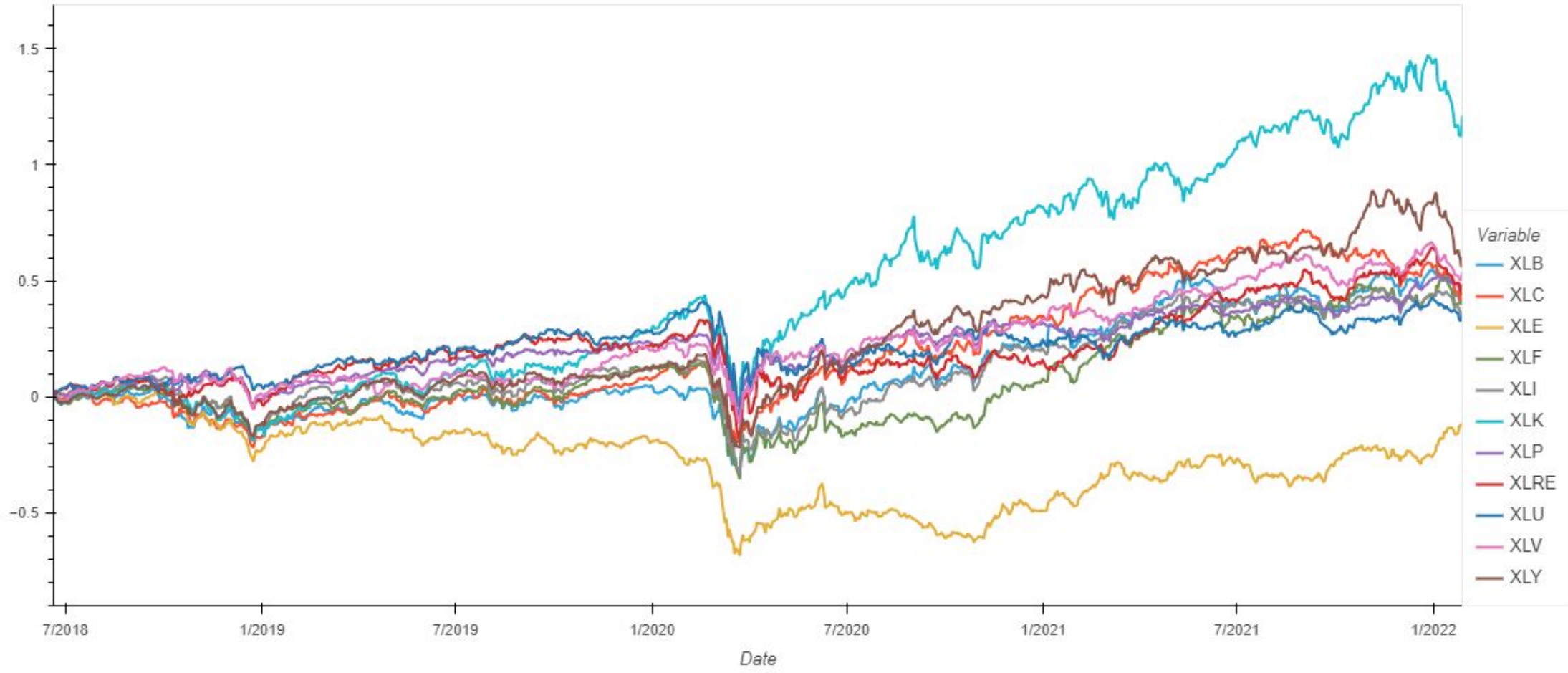


01

STOCK CUMULATIVE RETURNS

STOCKS CHART CONTAINING CUMULATIVE RETURNS FOR ALL STOCK ETFs

- Takeaway - (XLK) SPDR Tech sector ETF had significantly higher cumulative returns than other sectors



01

FINDINGS OF CUMULATIVE RETURNS

INVESTMENT SCENARIO IF INVESTING $10,000 IN EACH CRYPTOCURRENCY ON JUNE 19, 2018

ETF TICKER	PRICE JUNE 19 2018	PRICE AT PEAK	GROSS RETURN
XLK	$71.40	$177.04	$24,795.52
XLY	$111.17	$215.06	$19,241.30
XLV	$84.85	$141.98	$16,733.06
XLRE	$31.50	$52.17	$16,561.91
XLC	$49.96	$86.36	$17,285.83

$50,000 To $94,617.62 Max Return.
1.8X +

* Gross return does not include fees

02

BONDS

- iShares 3-7 Year Treasury Bond ETF - IEI
- iShares 7-10 Year Treasury Bond ETF - IEF
- iShares 10-20 Year Treasury Bond ETF - TLH
- iShares 20+ Year Treasury Bond ETF - TLT

02

BONDS CLOSE DATAFRAME

IMPORTING FINANCIAL DATA FOR ALL GOVERNMENT BONDS

API CALL

```
# Set timeframe to '1D'
timeframe = "1D"

# Set start and end datetimes
start_date = pd.Timestamp("2018-02-09", tz="America/New_York").isoformat()
end_date = pd.Timestamp("2022-01-28", tz="America/New_York").isoformat()

# Set the ticker information
bond_tickers = ["IEI", "IEF", "TLT", "TLH"]

# Get 1000 trading days worth of historical price data for tickers
bond_ticker_df = api.get_barset(
    bond_tickers,
    timeframe,
    start=start_date,
    end=end_date,
    limit=1000,
).df

# Display sample data
bond_ticker_df.head()
```

02

BONDS CLOSE DATAFRAME

IMPORTING FINANCIAL DATA FOR ALL GOVERNMENT BONDS

DATA CLEANING

	IEF				
	open	high	low	close	volume
time					
2018-02-09 00:00:00-05:00	102.11	102.81	102.110	102.22	3951443.0

```
# Remove timestamp from date index
bond_ticker_df['Date'] = bond_ticker_df.index.date
bond_ticker_df.set_index('Date', inplace = True)
bond_ticker_df.index = pd.to_datetime(bond_ticker_df.index)
bond_ticker_df.head()
```

```
# Drop all columns except close
bond_ticker_close_df = bond_ticker_df.iloc[:, bond_ticker_df.columns.get_level_values(1)=='close']
```

```
# Drop multilevel column name 'close' to leave ticker names for each column
bond_ticker_close_df.columns = bond_ticker_close_df.columns.droplevel(1)
bond_ticker_close_df.head()
```

02

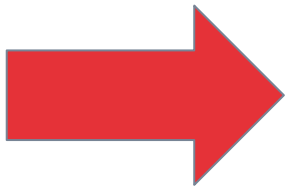
BONDS CLOSE DATAFRAME

IMPORTING FINANCIAL DATA FOR ALL GOVERNMENT BONDS

API CALL

DATA CLEANING

CREATION OF DATAFRAME

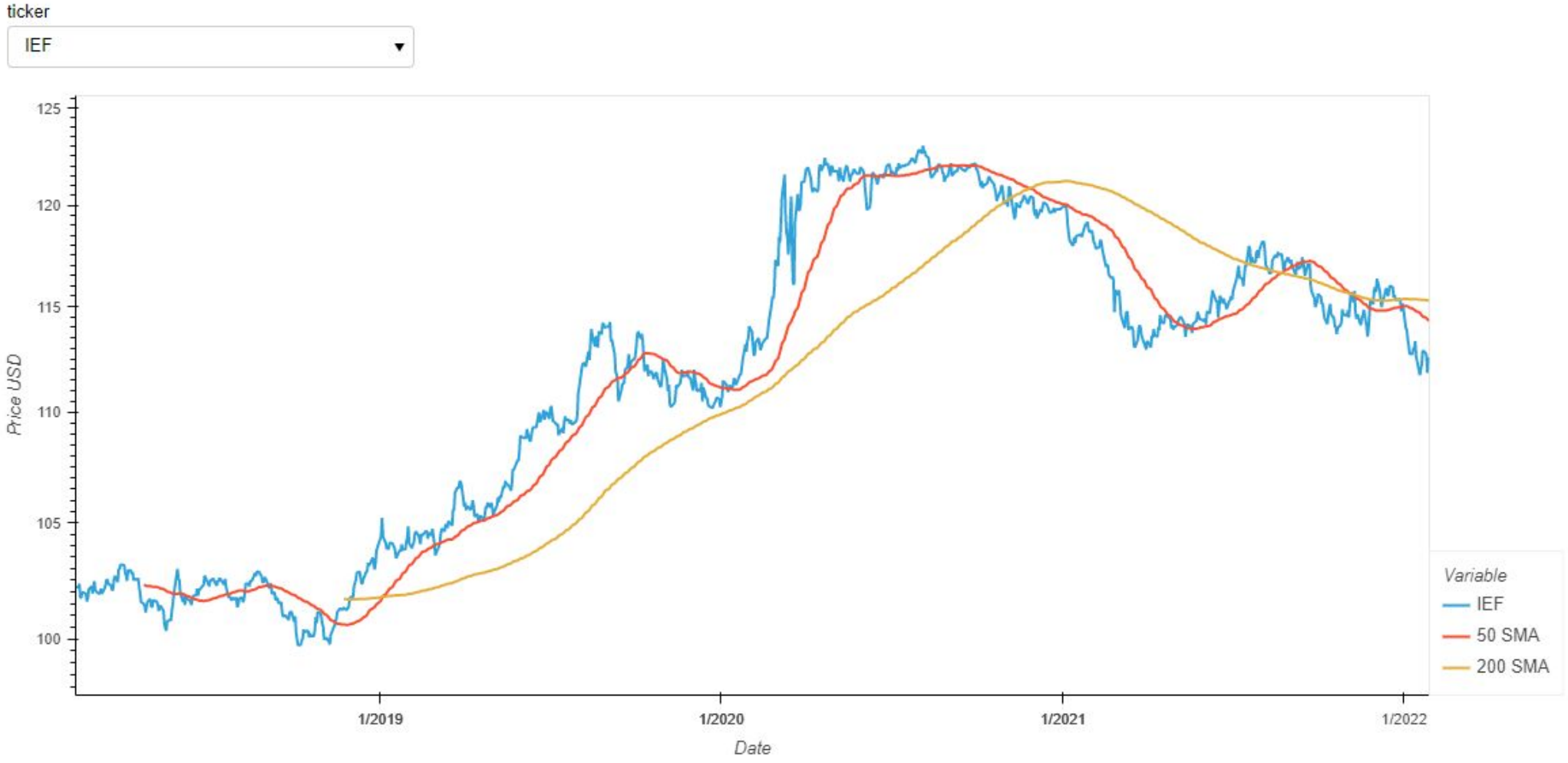


Date	IEF	IEI	TLH	TLT
2018-02-09	102.22	120.420	129.8700	117.90
2018-02-12	102.19	120.300	130.1600	118.47
2018-02-13	102.35	120.385	130.3600	118.99
2018-02-14	101.73	119.890	129.3999	117.71
2018-02-15	101.78	119.880	129.6404	118.07
...	...	...	...	...
2022-01-24	112.78	127.190	143.5800	142.50
2022-01-25	112.65	127.080	143.3500	142.18
2022-01-26	111.84	126.490	141.7900	140.51
2022-01-27	112.33	126.560	143.7150	143.12
2022-01-28	112.56	126.800	143.9500	143.12

02

200 & 50 DAY MOVING AVERAGE FOR BONDS

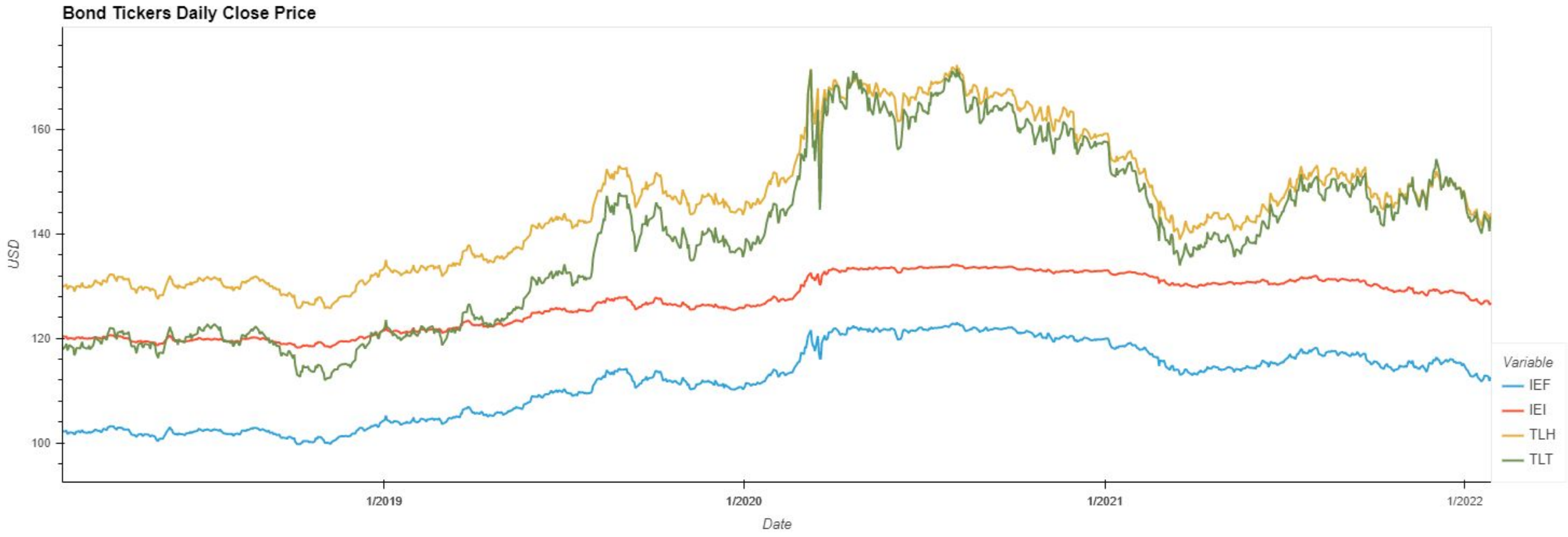
BONDS CHART CONTAINING CLOSING PRICES AND MOVING AVERAGES WITH DROP DOWN SELECTOR



02

BONDS CLOSE PRICE PLOT

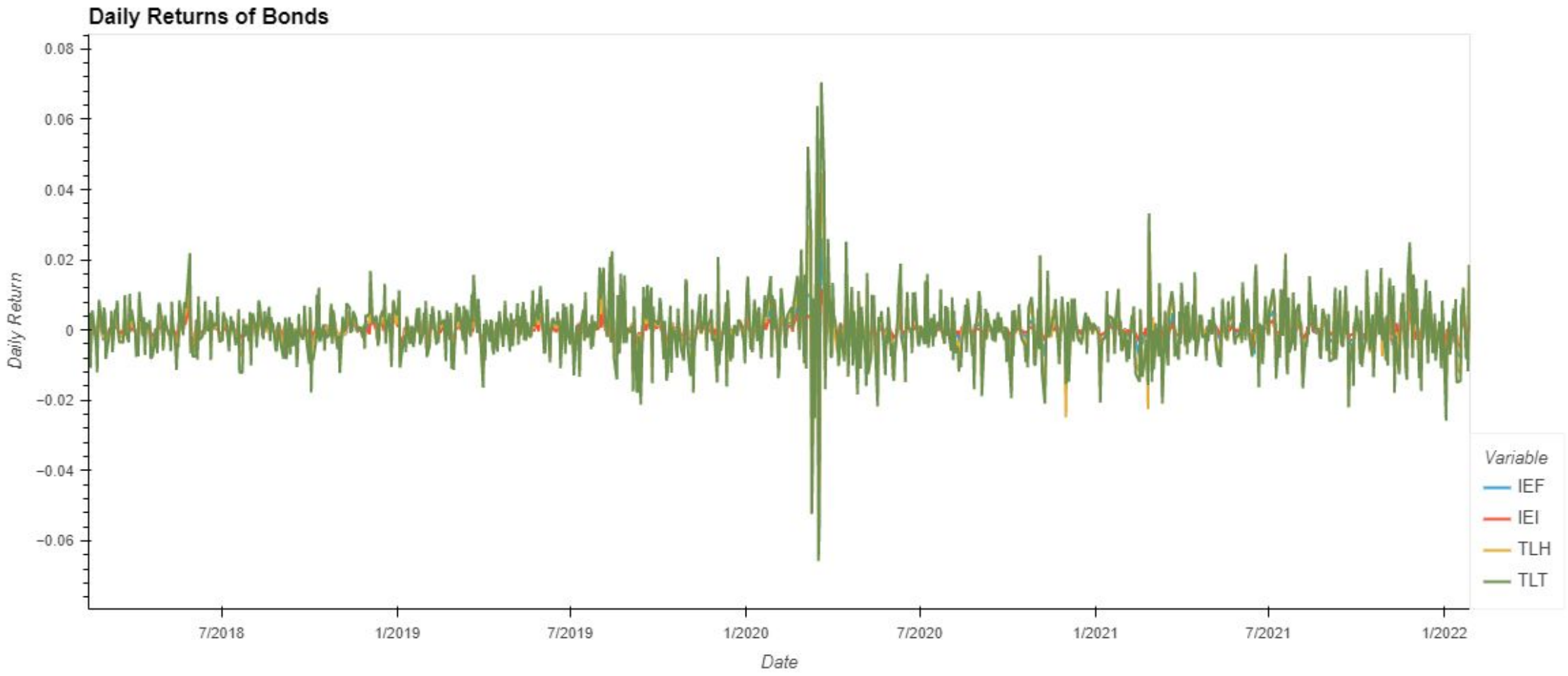
BONDS CHART CONTAINING CLOSING PRICES FOR ALL GOVERNMENT BONDS



02

BONDS DAILY RETURNS

BONDS CHART CONTAINING DAILY RETURNS FOR ALL GOVERNMENT BONDS

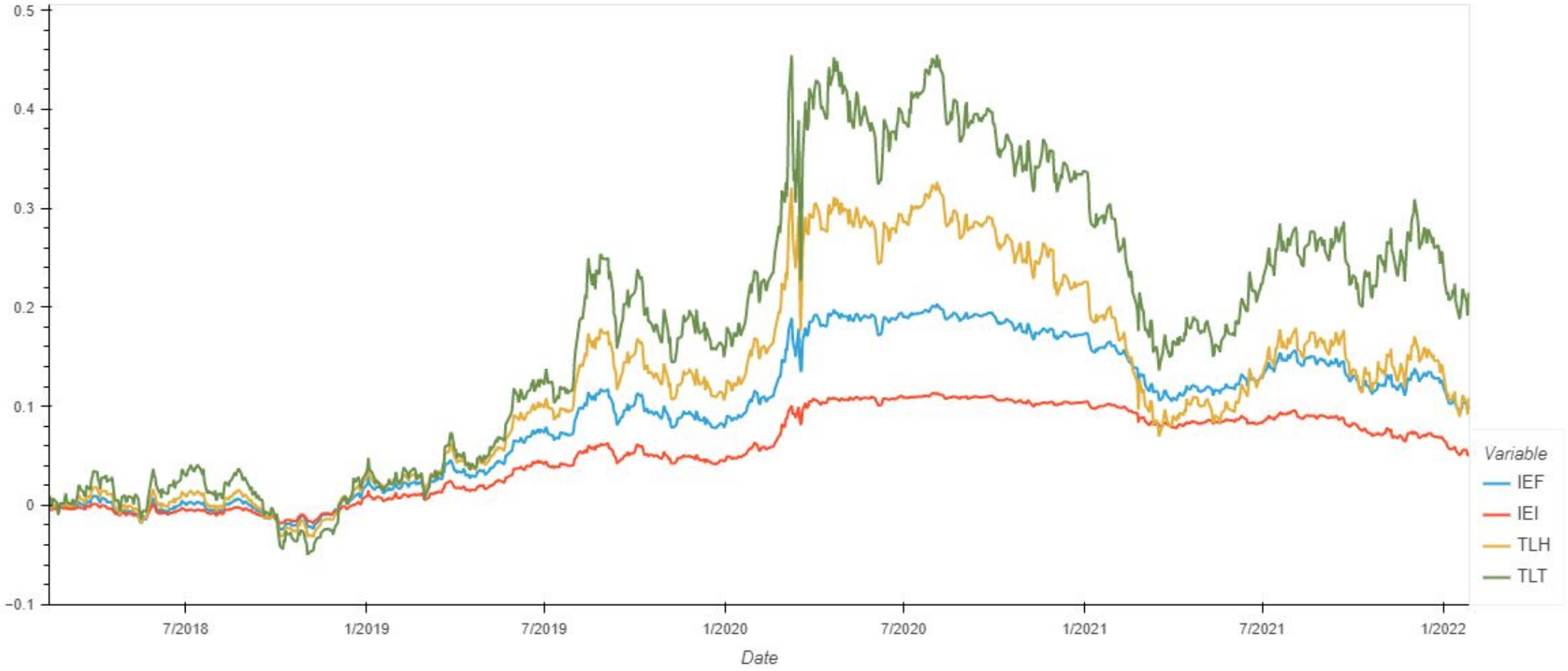


02

BONDS CUMULATIVE RETURNS

CHART CONTAINING CUMULATIVE RETURNS FOR ALL GOVERNMENT BONDS

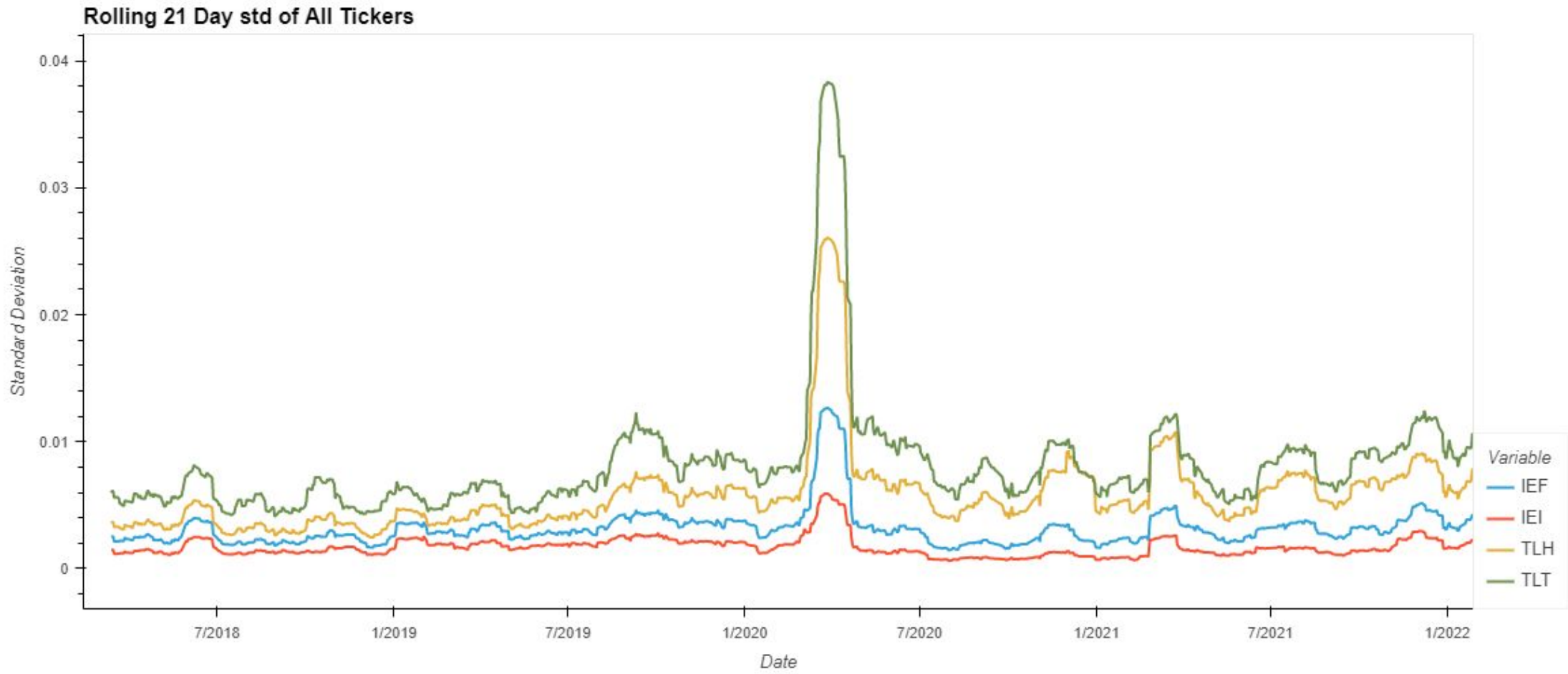
- Takeaway - TLT (20+) has highest cumulative returns over our analysis time frame



02

BOND ROLLING 21 STANDARD DEVIATION

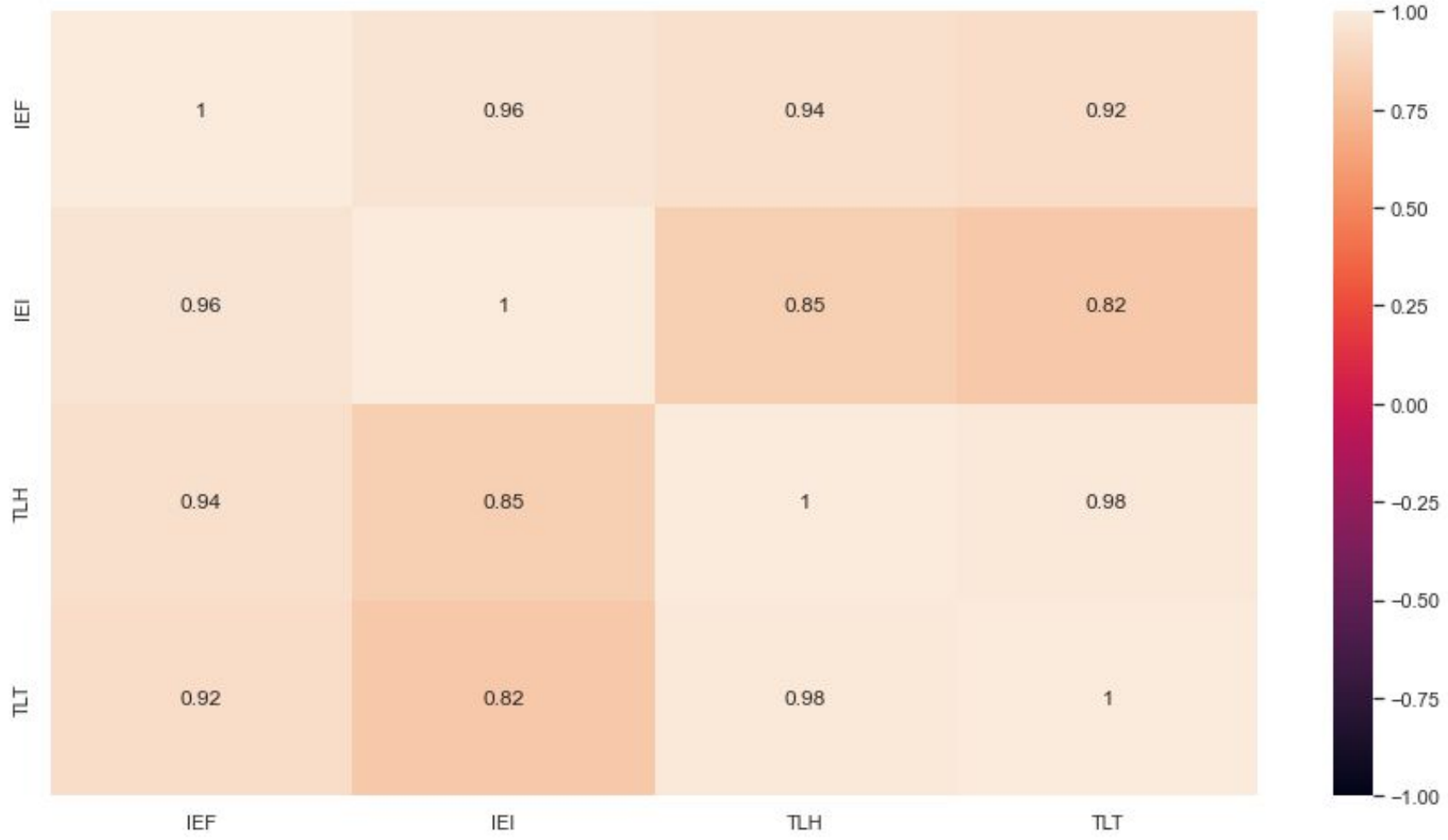
CHART CONTAINING ROLLING 21 STANDARD DEVIATION FOR ALL GOVERNMENT BONDS



02

CORRELATION OF BONDS

HEAT MAP CONTAINING GENERAL CORRELATION FOR ALL GOVERNMENT BONDS



02

FINDINGS OF CUMULATIVE RETURNS

INVESTMENT SCENARIO IF INVESTING $10,000 IN EACH CRYPTOCURRENCY ON FEBRUARY 2018

BOND	PRICE FEB 09 2020	PRICE AT PEAK	GROSS RETURN
IEF	$102.22	$123.41	$12,072.98
IEI	$120.42	$134.11	$11,136.85
TLH	$129.87	$175.92	$13,545.85
TLT	$117.90	$179.70	$15,241.73

$40,000 To $51,997.35 Max Return.
1.29X

* Gross return does not include fees

03

CRYPTOCURRENCY

- Bitcoin - BTC
- Ethereum - ETH
- Binance Coin - BNB
- Cardano - ADA
- Ripple - XRP

03

DATA CLEANING & EXPLORATION

IMPORTING FINANCIAL DATA FOR FIVE CRYPTOCURRENCIES

API CALL

Binance API Call and Creation of Dataframe for Each Crypto

```
# Get crypto data for each ticker
def get_crypto(symbol, interval, startTime, endTime):

    url = "https://api1.binance.com/api/v3/klines"

# Set start time, end time, and limit variables for parameters
    startTime = str(int(startTime.timestamp() * 1000))
    endTime = str(int(endTime.timestamp() * 1000))
    limit = "1000"

# Create a dataframe to store paramater values
    req_params = {"symbol" : symbol, 'interval' : interval, 'startTime' : startTime, 'endTime' : endTime, 'limit' : limit}

# Pull first 6 parameter columns from api for each cryptocurrency
    df_crypto = pd.DataFrame(json.loads(requests.get(url, params = req_params).text))
    df_crypto  = df_crypto.iloc[:, 0:5]
    df_crypto.columns = ['datetime', 'open', 'high', 'low', 'close']



# Set datetime as index and drop unused columns
    df_crypto.index = [dt.datetime.fromtimestamp(x / 1000.0) for x in df_crypto.datetime]
    df_crypto.drop(['datetime', 'high', 'low', 'open'], axis=1, inplace=True)
    return df_crypto
```

03

DATA CLEANING & EXPLORATION

IMPORTING FINANCIAL DATA FOR FIVE CRYPTOCURRENCIES

DATA CLEANING

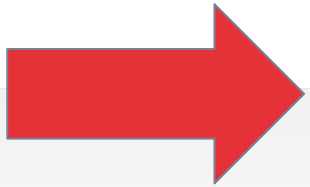
	BTC
2019-05-04 18:00:00	5775.62
2019-05-05 18:00:00	5747.79
2019-05-06 18:00:00	5846.34
2019-05-07 18:00:00	5987.29
2019-05-08 18:00:00	6209.18
...	...
2022-01-23 17:00:00	36660.35
2022-01-24 17:00:00	36958.32
2022-01-25 17:00:00	36809.34
2022-01-26 17:00:00	37160.10
2022-01-27 17:00:00	37716.56

	ETH
2019-05-04 18:00:00	163.05
2019-05-05 18:00:00	173.00
2019-05-06 18:00:00	169.18
2019-05-07 18:00:00	170.85
2019-05-08 18:00:00	170.90
...	...
2022-01-23 17:00:00	2439.29
2022-01-24 17:00:00	2458.83
2022-01-25 17:00:00	2462.99
2022-01-26 17:00:00	2424.25
2022-01-27 17:00:00	2544.99

	BNB
2019-05-04 18:00:00	22.9777
2019-05-05 18:00:00	22.1895
2019-05-06 18:00:00	20.6602
2019-05-07 18:00:00	20.6673
2019-05-08 18:00:00	18.8388
...	...
2022-01-23 17:00:00	371.3000
2022-01-24 17:00:00	385.1000
2022-01-25 17:00:00	375.1000
2022-01-26 17:00:00	389.9000
2022-01-27 17:00:00	385.9000

	ADA
2019-05-04 18:00:00	0.06603
2019-05-05 18:00:00	0.06696
2019-05-06 18:00:00	0.06412
2019-05-07 18:00:00	0.06470
2019-05-08 18:00:00	0.06169
...	...
2022-01-23 17:00:00	1.06500
2022-01-24 17:00:00	1.04300
2022-01-25 17:00:00	1.07700
2022-01-26 17:00:00	1.04200
2022-01-27 17:00:00	1.04800

	XRP
2019-05-04 18:00:00	0.30044
2019-05-05 18:00:00	0.30283
2019-05-06 18:00:00	0.29819
2019-05-07 18:00:00	0.29958
2019-05-08 18:00:00	0.29488
...	...
2022-01-23 17:00:00	0.61180
2022-01-24 17:00:00	0.61800
2022-01-25 17:00:00	0.61930
2022-01-26 17:00:00	0.60880
2022-01-27 17:00:00	0.61110



	BTC	ETH	BNB	ADA	XRP
2019-05-04 18:00:00	5775.62	163.05	22.9777	0.06603	0.30044
2019-05-05 18:00:00	5747.79	173.00	22.1895	0.06696	0.30283
2019-05-06 18:00:00	5846.34	169.18	20.6602	0.06412	0.29819
2019-05-07 18:00:00	5987.29	170.85	20.6673	0.06470	0.29958
2019-05-08 18:00:00	6209.18	170.90	18.8388	0.06169	0.29488
...	...	...	...	...	...
2022-01-23 17:00:00	36660.35	2439.29	371.3000	1.06500	0.61180
2022-01-24 17:00:00	36958.32	2458.83	385.1000	1.04300	0.61800
2022-01-25 17:00:00	36809.34	2462.99	375.1000	1.07700	0.61930
2022-01-26 17:00:00	37160.10	2424.25	389.9000	1.04200	0.60880
2022-01-27 17:00:00	37716.56	2544.99	385.9000	1.04800	0.61110

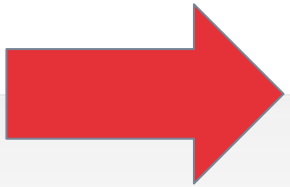
03

DATA CLEANING & EXPLORATION

IMPORTING FINANCIAL DATA FOR FIVE CRYPTOCURRENCIES

DATA CLEANING

	BTC	ETH	BNB	ADA	XRP
2019-05-04 18:00:00	5775.62	163.05	22.9777	0.06603	0.30044
2019-05-05 18:00:00	5747.79	173.00	22.1895	0.06696	0.30283
2019-05-06 18:00:00	5846.34	169.18	20.6602	0.06412	0.29819
2019-05-07 18:00:00	5987.29	170.85	20.6673	0.06470	0.29958
2019-05-08 18:00:00	6209.18	170.90	18.8388	0.06169	0.29488
...	...	...	...	...	...
2022-01-23 17:00:00	36660.35	2439.29	371.3000	1.06500	0.61180
2022-01-24 17:00:00	36958.32	2458.83	385.1000	1.04300	0.61800
2022-01-25 17:00:00	36809.34	2462.99	375.1000	1.07700	0.61930
2022-01-26 17:00:00	37160.10	2424.25	389.9000	1.04200	0.60880
2022-01-27 17:00:00	37716.56	2544.99	385.9000	1.04800	0.61110

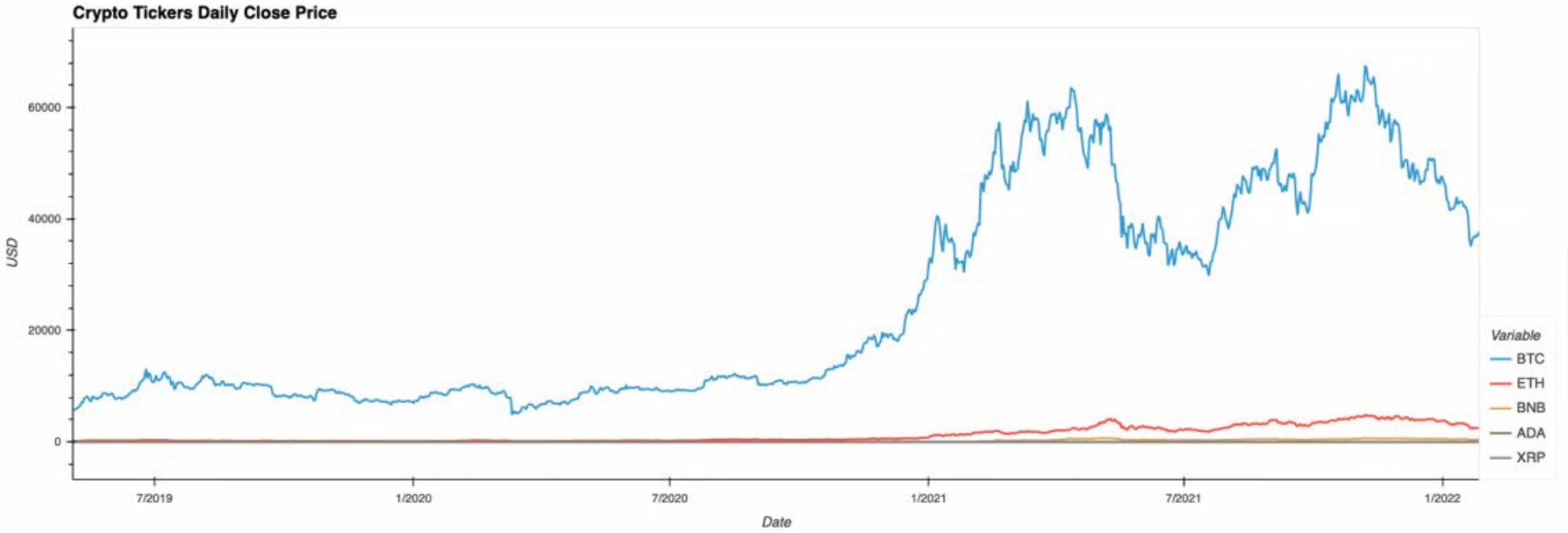


Date	BTC	ETH	BNB	ADA	XRP
2019-05-04	5775.62	163.05	22.9777	0.06603	0.30044
2019-05-05	5747.79	173.00	22.1895	0.06696	0.30283
2019-05-06	5846.34	169.18	20.6602	0.06412	0.29819
2019-05-07	5987.29	170.85	20.6673	0.06470	0.29958
2019-05-08	6209.18	170.90	18.8388	0.06169	0.29488
...	...	...	...	...	...
2022-01-23	36660.35	2439.29	371.3000	1.06500	0.61180
2022-01-24	36958.32	2458.83	385.1000	1.04300	0.61800
2022-01-25	36809.34	2462.99	375.1000	1.07700	0.61930
2022-01-26	37160.10	2424.25	389.9000	1.04200	0.60880
2022-01-27	37716.56	2544.99	385.9000	1.04800	0.61110

03

CRYPTOCURRENCY CLOSE PRICE PLOT

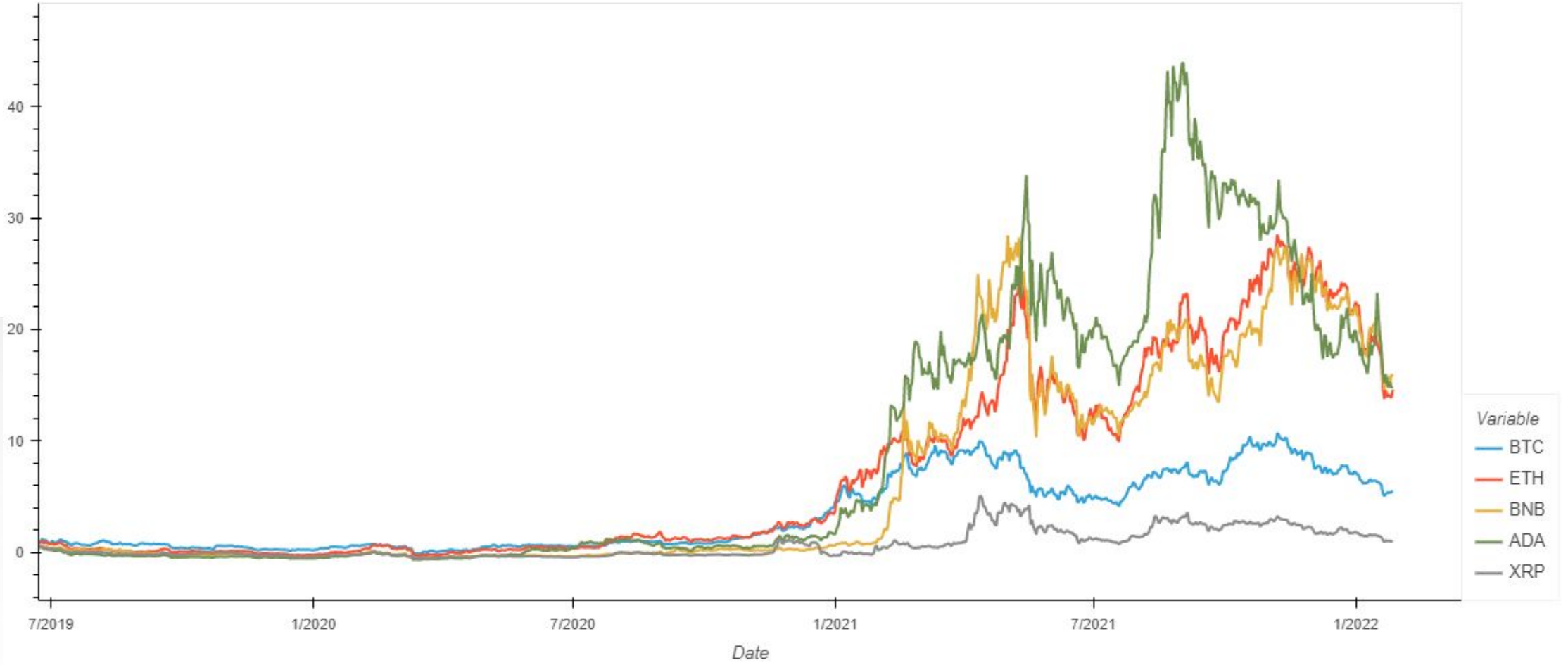
CHART CONTAINING CLOSING PRICES FOR FIVE CRYPTOCURRENCIES



03

CRYPTOCURRENCY CUMULATIVE RETURNS

CHART CONTAINING CUMULATIVE RETURNS FOR FIVE CRYPTOCURRENCIES



03

FINDINGS OF CUMULATIVE RETURNS

INVESTMENT SCENARIO IF INVESTING $10,000 IN EACH CRYPTOCURRENCY ON MAY 4TH 2019

CRYPTO	PRICE MAY 4TH 2019	PRICE AT PEAK	GROSS RETURN
BITCOIN	$5775.62	$69,000.00	$119,467.69
ETHEREUM	$163.05	$4868	$298,558.72
BINANCE COIN	$22.97	$691.80	$301,175.45
CARDANO	$0.06603	$3.10	$469,483.57
XRP	$0.30044	$1.97	$65,570.50

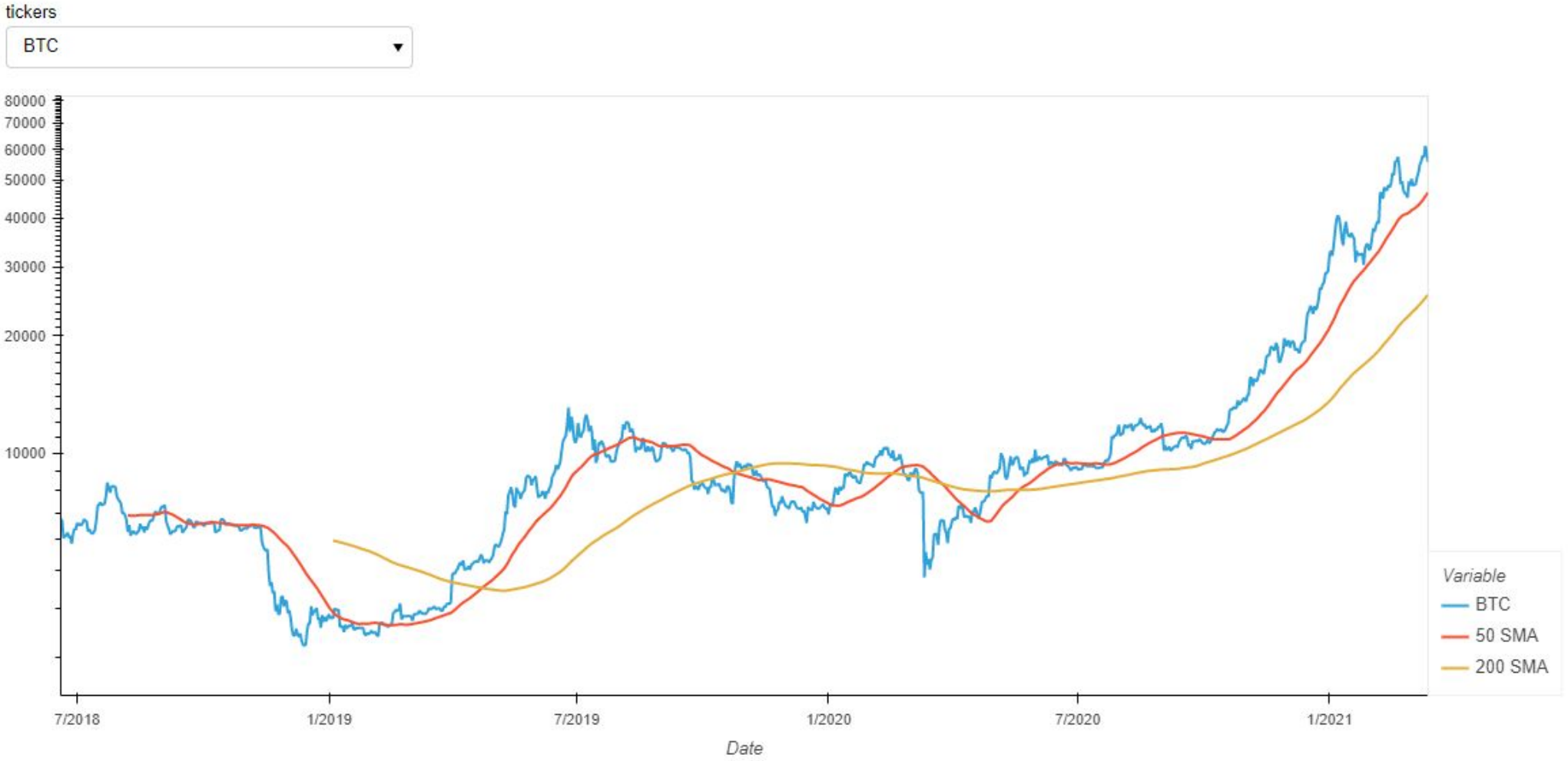
$50,000 To $1,254,255.93 Max Return.
25X +

* Gross return does not include fees

03

200 & 50 DAY MOVING AVERAGE FOR CRYPTO

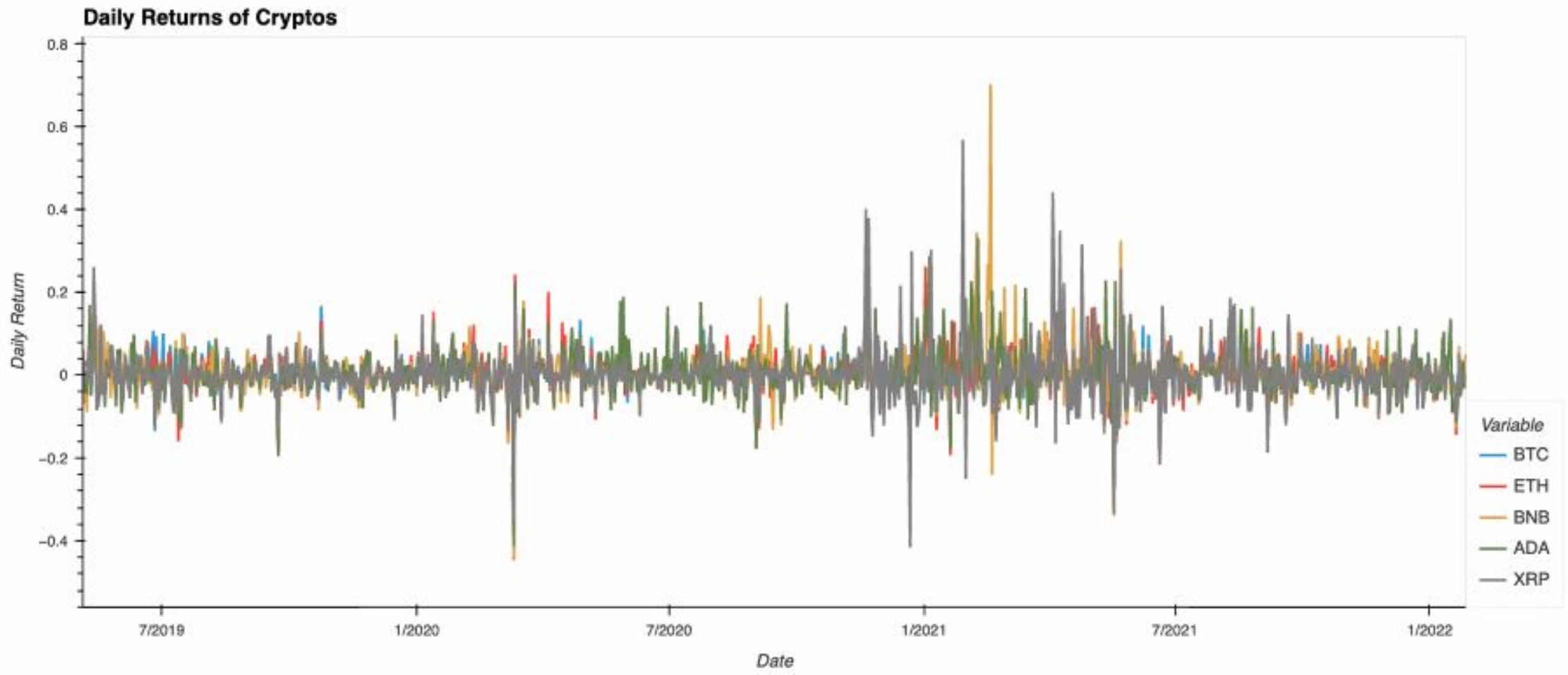
CHART CONTAINING CLOSING PRICES AND MOVING AVERAGES WITH DROP DOWN SELECTOR



03

CRYPTOCURRENCY DAILY RETURNS

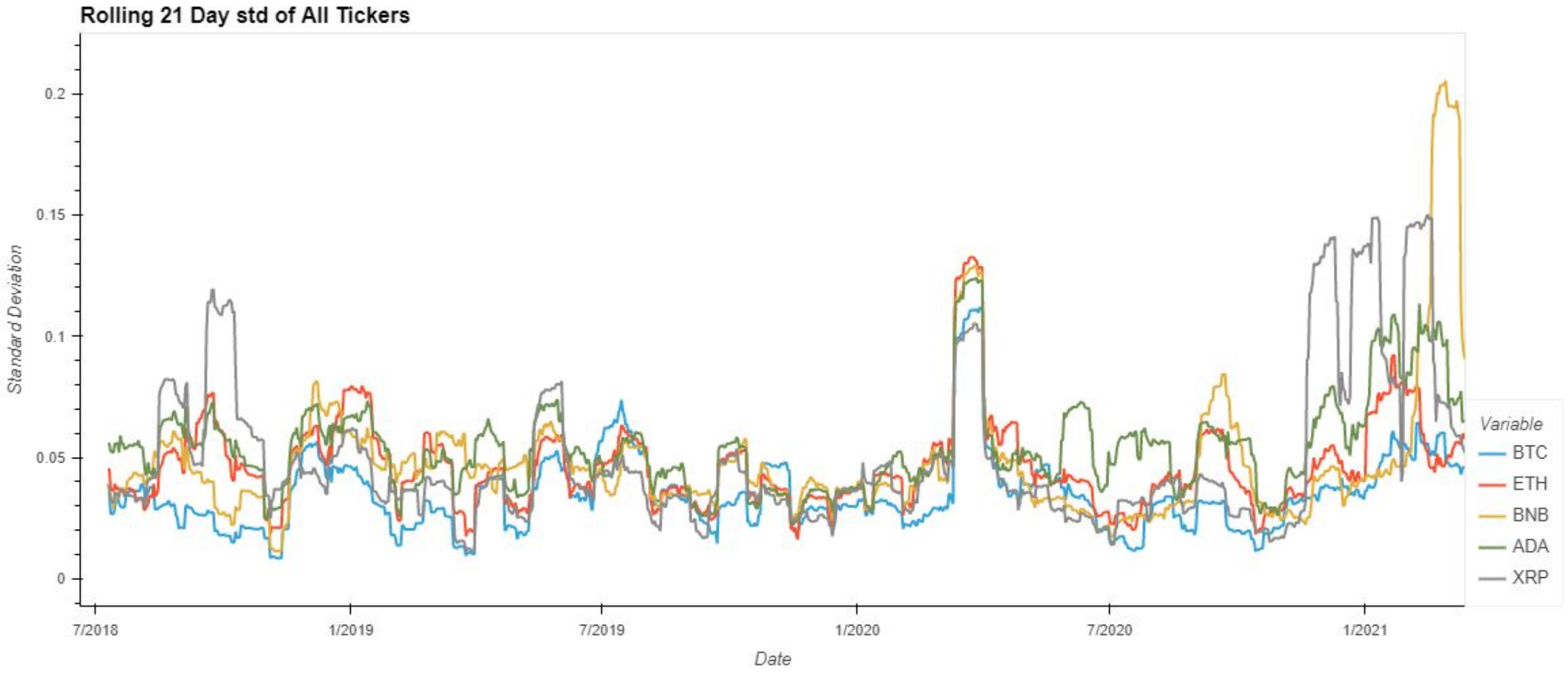
CHART CONTAINING DAILY RETURNS FOR FIVE CRYPTOCURRENCIES



03

CRYPTO ROLLING 21 STANDARD DEVIATION

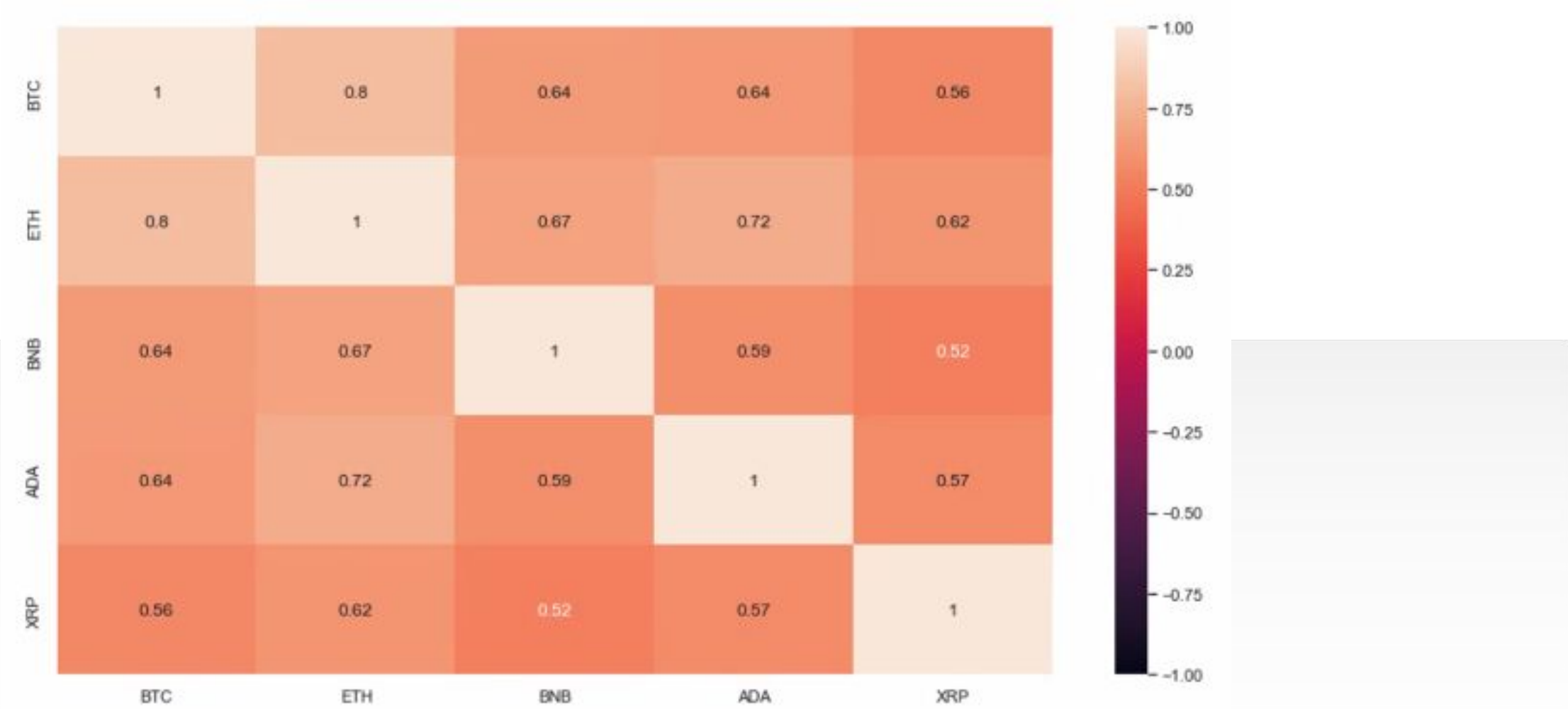
CHART CONTAINING ROLLING 21 STANDARD DEVIATION FOR FIVE CRYPTOCURRENCIES



03

CORRELATION OF CRYPTOCURRENCIES

HEAT MAP CONTAINING GENERAL CORRELATION FOR FIVE CRYPTOCURRENCIES



04

ASSET CLASS ANALYSIS

STOCKS, BONDS, AND CRYPTOCURRENCIES

04 DATA AGGREGATION

Aggregating data for asset class comparison

Mean Daily Return for All Stocks

```
stocks_daily_returns_mean = stock_daily_returns.mean(axis=1)
```

```
stocks_daily_returns_mean
```

```
Date
2018-06-20    0.002910
2018-06-21   -0.005379
2018-06-22    0.005485
2018-06-25   -0.010163
2018-06-26    0.002526
              ...
2022-01-24    0.002355
2022-01-25   -0.006488
2022-01-26   -0.006581
2022-01-27   -0.002763
2022-01-28    0.017555
Length: 907, dtype: float64
```

- First we found the daily returns mean for each asset class using the mean function on our previously created daily returns dataframes
  - Comparing on return % is a more “level playing field” than comparing on prices which widely vary
- Second, we found the mean cumulative returns for each asset class
  - In order to identify which class yielded largest cumulative returns
- Third, we found the mean standard deviation for each asset class
  - In order to identify the most and least volatile asset classes

Mean Cumulative Return for All Stocks

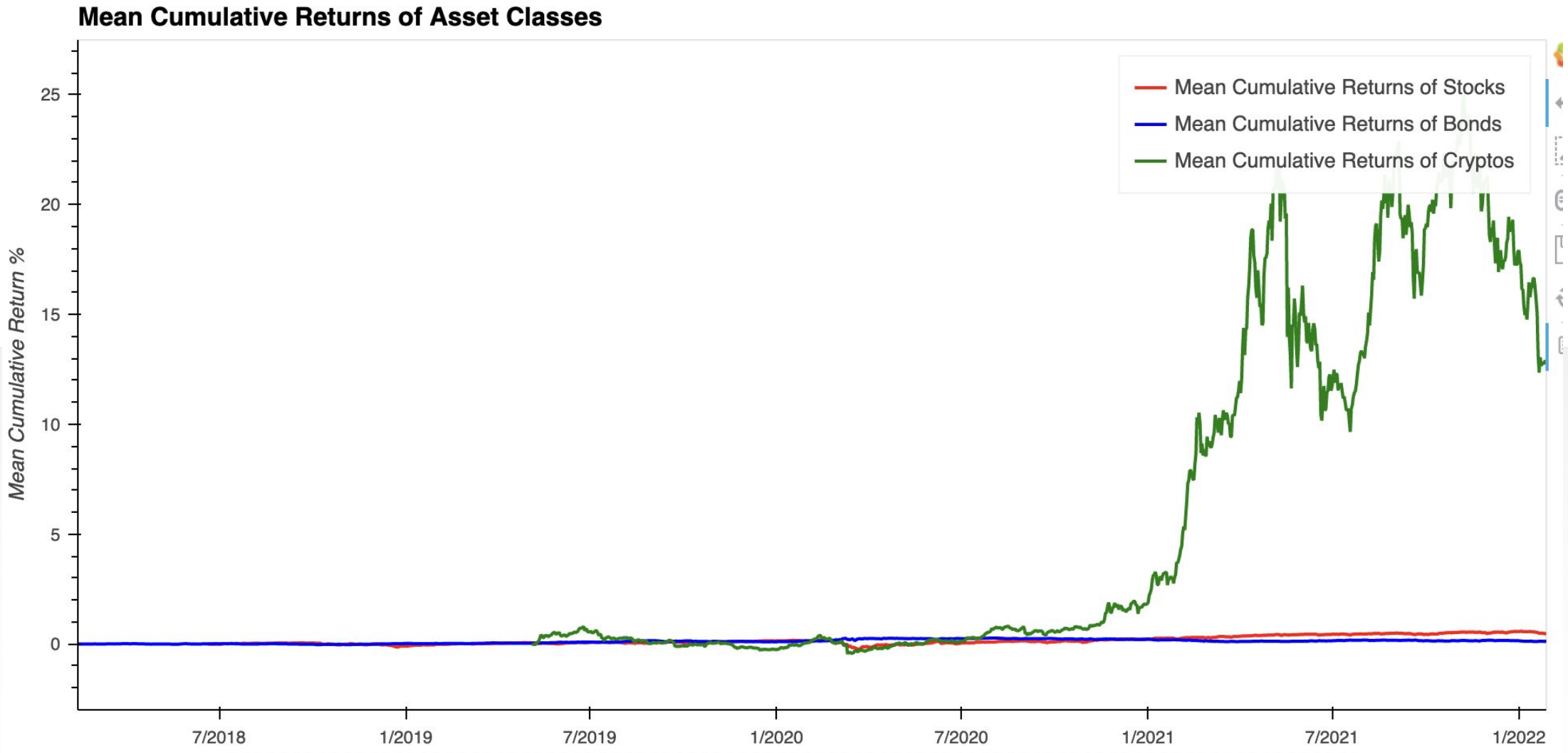
```
stocks_cumulative_returns_mean = (1 + stocks_daily_returns_mean).cumprod() - 1
```

```
# Calculate the daily standard deviations of all portfolios
stocks_daily_returns_mean_std = stocks_daily_returns_mean.std()
```

04

DATA EXPLORATION

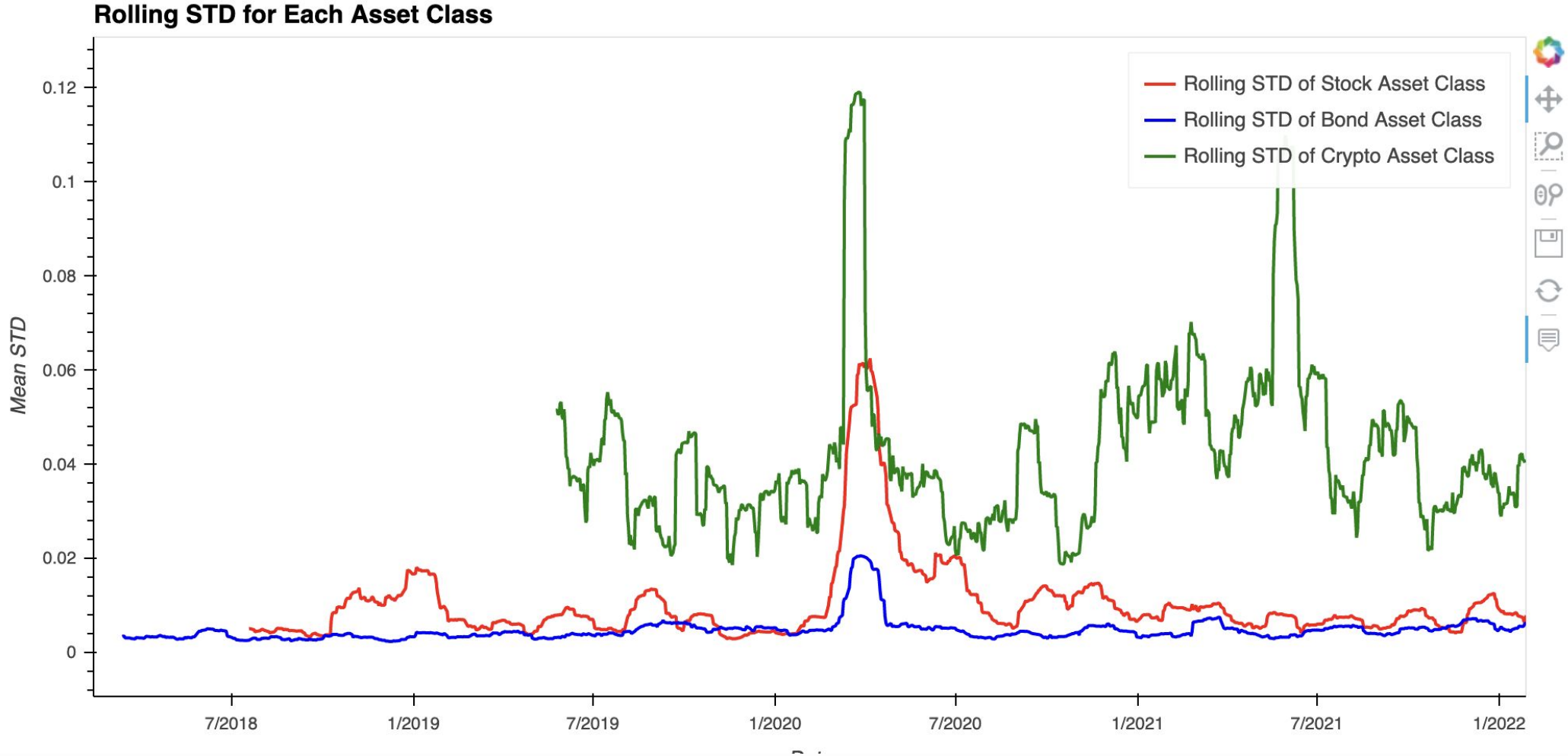
Plotting Mean Cumulative Returns of Each Asset Class



04

DATA EXPLORATION

Plotting Mean Cumulative Returns of Each Asset Class

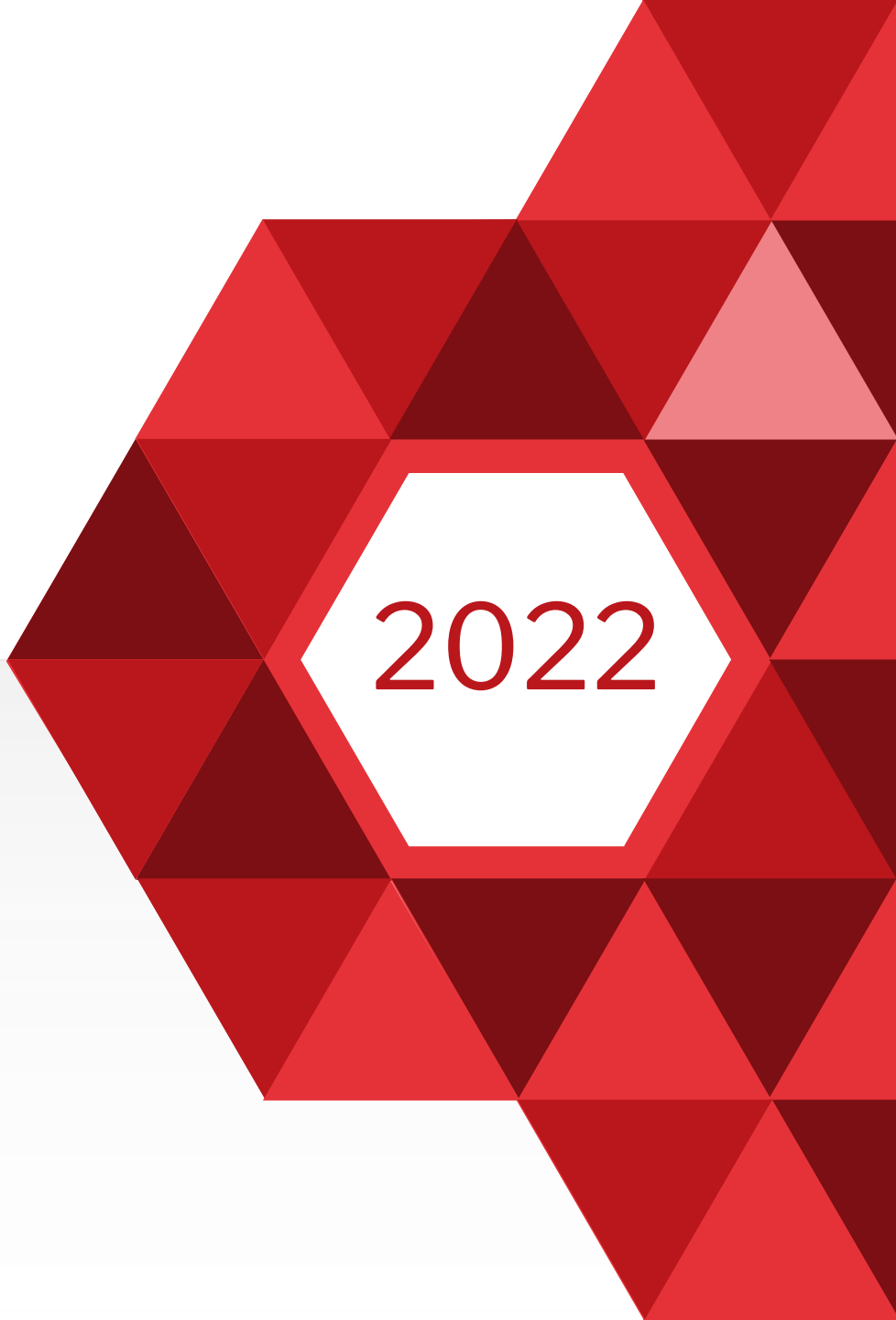


Key Findings

- Within each asset class, the assets tend to have a medium to strong positive correlation with each other over our analyzed time frame

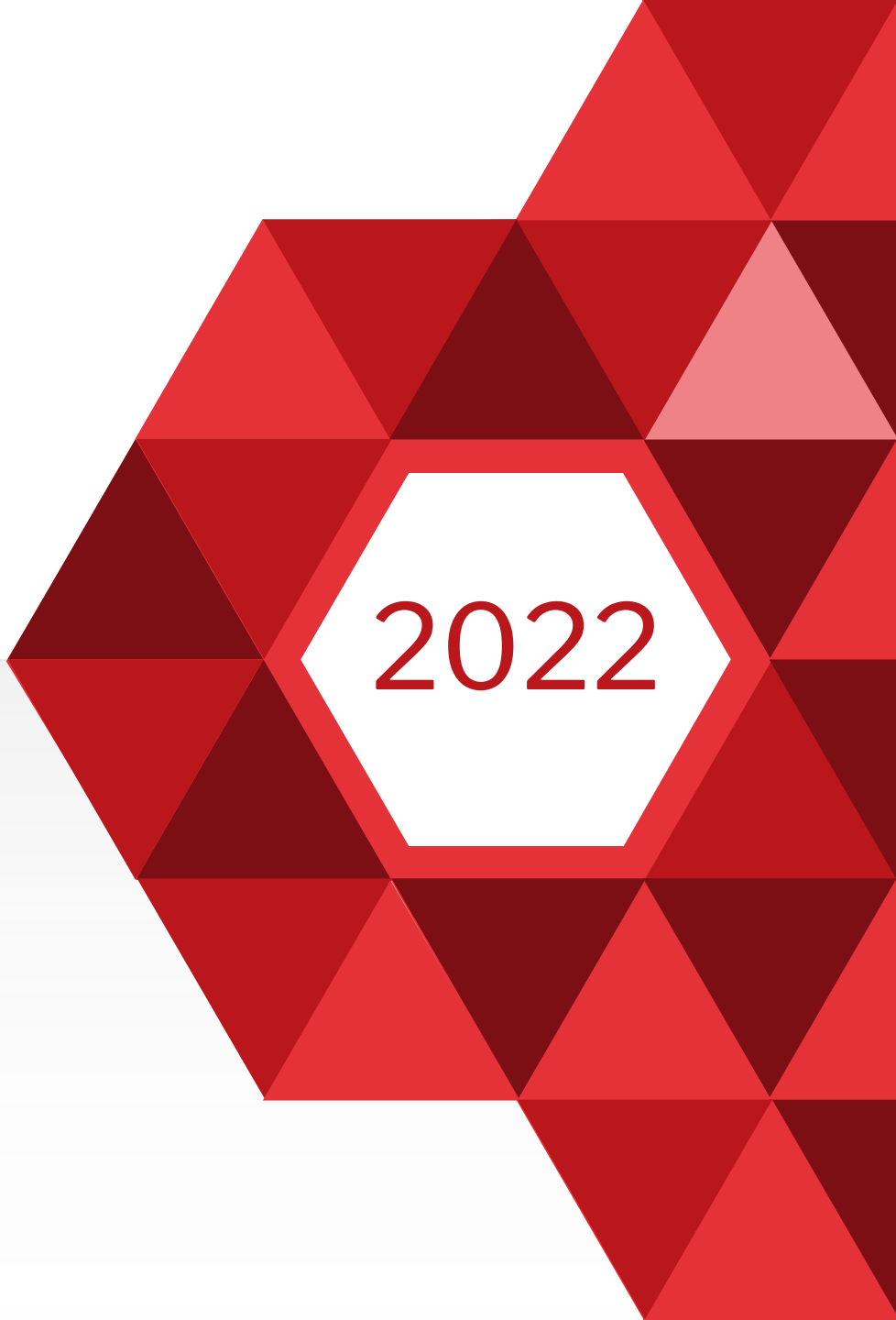
- When comparing each asset class, because crypto has the higher rolling standard deviation, our analysis indicates that this higher risk yields higher return - nearly 1500% more than the other asset classes over our analyzed time frame

- Although crypto had the best performance during the analyzed time frame, all three asset classes suffered during the economic downturn of 2020, with crypto being impacted the most



Key Takeaways

- We were able to successfully used ALPACA API and the BINANCE API to call historical data of Stocks, Bonds, and Cryptocurrencies and compare them
- Despite our API call to ALPACA sending us multi-indexed columns, we were able to perform operations that allowed us to unstack the data and analyze only columns of interest
- For each asset class we created the same standard set of plots to compare how they performed. We included drop down widgets and interactive hvplots
- We used PostgresQL to store the key dataframes from our analyses notebook so that they can be called in our dashboard notebook
- We programmed the dashboard logic in a python script and called it into the dashboard notebook for ease of access
- We created a dashboard that can be used to easily compare assets within the asset classes, and then asset classes themselves



THANK YOU!