Table of Contents

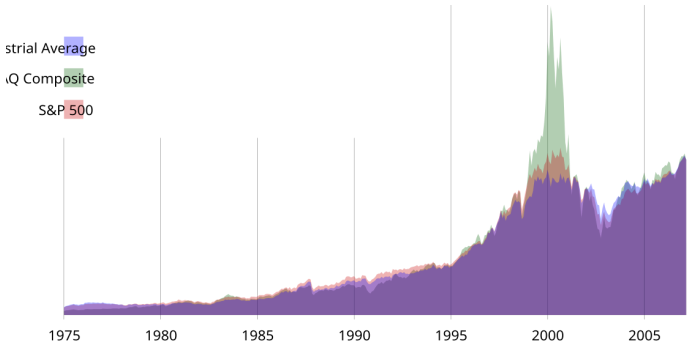
- 1 Data
  - 1.1 Yahoo! Finance
  - 1.2 S&P 500 and the VIX
    - 1.2.1 Prices and Returns from Yahoo! Finance
  - 1.3 Tradeable VIX-like Products
  - 1.4 CBOE Historical Data
- 2 Portfolio Management Relative To a Benchmark
  - 2.1 What's the Investable Universe?
  - 2.2 Active Management Formulation
  - 2.3 Least Squares Formulation
- 3 Constructing a Volatility Index Tracking Fund
  - 3.1 Tracking the VIX with an Investable Universe
    - 3.1.1 Tracking the VIX with Futures
    - 3.1.2 Tracking the VIX with Negative Beta Stocks
    - 3.1.3 Tracking the VIX with High Volatility Stocks
    - 3.1.4 Tracking the VIX with High Correlation with VIX Stocks
    - 3.1.5 Constructing the Investable Universe Returns Data Set
  - 3.2 Decision Variables
  - 3.3 Objective
  - 3.4 Constraints
  - 3.5 Solution and Analysis
-

Imports

```
import os
import sys

import numpy as np
import pandas as pd
import pandas_datareader as pdr
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

import tensorflow as tf
import cvxpy as cp

from google.colab import drive
drive.mount('/content/drive')

print("python \t {}".format(".".join(map(str, sys.version_info[:3]))))
print("numpy \t {}".format(np.__version__))
print("pandas \t {}".format(pd.__version__))
print("cvxpy \t {}".format(cp.__version__))
print("seaborn \t {}".format(sns.__version__))
print("matplotlib \t {}".format(matplotlib.__version__))
print("pandas-datareader {}".format(pdr.__version__))
```

```
Mounted at /content/drive
python   3.10.12
numpy    1.23.5
pandas   1.5.3
cvxpy    1.3.2
```

```
seaborn          0.12.2
matplotlib       3.7.1
pandas-datareader 0.10.0
```



Data

Yahoo! Finance

- We will use a Python package called Pandas Data Reader to quickly scrape data from Yahoo! Finance.
- Pandas Data Reader has additional interesting functionality that is worth checking out, if interested.

```
! pip install pandas-datareader
```

```
Requirement already satisfied: pandas-datareader in /usr/local/lib/python3.10/dist-packages (0.10.0)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (from pandas-datareader) (4.9.3)
Requirement already satisfied: pandas>=0.23 in /usr/local/lib/python3.10/dist-packages (from pandas-datareader) (1.5.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from pandas-datareader) (2.31.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.23->pandas-
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.23->pandas-datareader
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.23->pandas-datareade
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pandas-datare
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pandas-
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pandas-
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas>=0.2
```

```
##### Imports

import pandas_datareader as pdr
import datetime
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import cvxpy as cp
```

S&P 500 and the VIX

- We will download daily OHLC (Open-High-Low-Close) variables for the VIX index as well as the S&P 500 ETF (SPY).

- It would be a good idea to write a function that allows us to grab any ticker we want from Yahoo! Finance and return a merged data frame consisting of adjusted closing prices and single period close-to-close returns, for each ticker.

- We will see, empirically, one particular benefit of being able to have a tradable asset that replicates the VIX, which motivates the need for developing such a product.

```
import pandas_datareader as pdr
import yfinance as yf
import datetime
```

```
spy = yf.download('SPY', start='1980-09-10', end=datetime.date.today())
```

```
vix = yf.download('^VIX', start='1980-09-10', end=datetime.date.today())
```

```
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
```

```
spy.head()
```

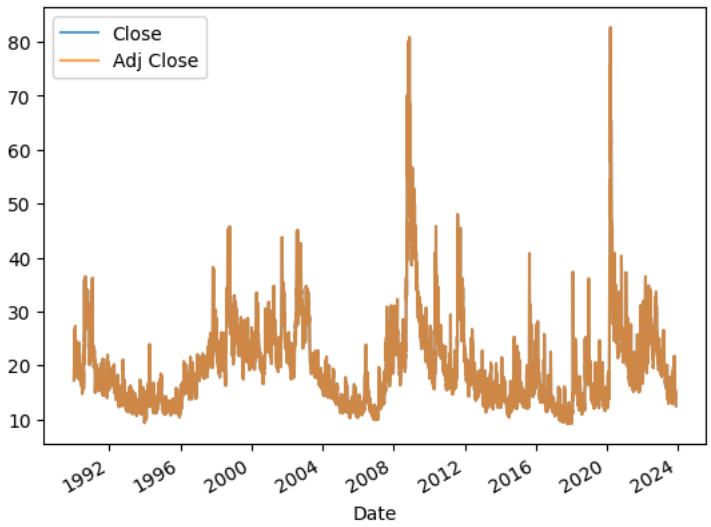
Date	Open	High	Low	Close	Adj Close	Volume
1993-01-29	43.96875	43.96875	43.75000	43.93750	24.941399	1003200
1993-02-01	43.96875	44.25000	43.96875	44.25000	25.118786	480500
1993-02-02	44.21875	44.37500	44.12500	44.34375	25.172005	201300
1993-02-03	44.40625	44.84375	44.37500	44.81250	25.438105	529400
1993-02-04	44.96875	45.09375	44.46875	45.00000	25.544527	531500

```
# Plot the difference between SPY's close and adjusted closing prices
spy[['Close', 'Adj Close']].plot(alpha=0.7)
```

```
<Axes: xlabel='Date'>
```


```
vix[['Close', 'Adj Close']].plot(alpha=0.7)
```

```
<Axes: xlabel='Date'>
```



- There is a difference between the **closing price** and the **adjusted closing price** (described by Yahoo! Finance).
  - The closing price is simply the cash value of that specific piece of stock at day's end while the adjusted closing price reflects the closing price of the stock in relation to other stock attributes.
  - In general, the adjusted closing price is considered to be a more technically accurate reflection of the true value of the stock.
  - The closing price of a stock is only its cash value at day's end, whereas the adjusted closing price factors in things like dividends, stock splits and new stock offerings.
  - For more, see here.

Prices and Returns from Yahoo! Finance

- The data, reported after closing (4pm ET), contains prices stamped at various times throughout the day (t):
  $$P_t^{open}, P_t^{high}, P_t^{low}, P_t^{close},$$
  as well as a measure of volume (V_t) and a closing price that is adjusted for various corporate actions
  $$P_t^{adj-close}.$$
- Note, the **return** of a security can be defined as the percentage change of the security's price
  $$r_{i,t} = \frac{P_{i,t} - P_{i,t-1}}{P_{i,t-1}},$$
  where $P_{i,t}$ is the price of security i at time t.
- Additionally, we can make a very crude estimate of the expected return of a security by taking the sample average of a time series of returns
  $$\mathbb{E}[r_{i,t}] \approx \frac{1}{T}\sum_{j=0}^{T-1} r_{i,t-j}\ .$$

□

```
# Create a data frame for daily returns

spy_vix_returns = pd.DataFrame()

spy_vix_returns['SPY_ret'] = (spy.loc[:, 'Adj Close'] / spy.loc[:, 'Adj Close'].shift()) - 1
```

```
spy_vix_returns['VIX_ret'] = (vix.loc[:, 'Adj Close'] / vix.loc[:, 'Adj Close'].shift()) - 1


# Create a portfolio that does a daily rebalancing to ensure there is 90% SPY and 10% VIX portfolio weights
spy_vix_returns['(0.9SPY+0.1VIX)_ret'] = 0.9*spy_vix_returns['SPY_ret'] + 0.1*spy_vix_returns['VIX_ret']


# (Arithmetic) Average of returns of the two portfolios
spy_vix_returns[['SPY_ret', '(0.9SPY+0.1VIX)_ret']].mean()
```

```
SPY_ret               0.000444
(0.9SPY+0.1VIX)_ret   0.000634
dtype: float64
```

```
# Standard deviation of daily returns of the two portfolios
spy_vix_returns[['SPY_ret', '(0.9SPY+0.1VIX)_ret']].std()
```

```
SPY_ret               0.011838
(0.9SPY+0.1VIX)_ret   0.007546
dtype: float64
```

```
# Sharpe ratio of the daily returns of the two portfolios
spy_vix_returns[['SPY_ret', '(0.9SPY+0.1VIX)_ret']].mean() / \
 spy_vix_returns[['SPY_ret', '(0.9SPY+0.1VIX)_ret']].std()
```

```
SPY_ret               0.037542
(0.9SPY+0.1VIX)_ret   0.084010
dtype: float64
```

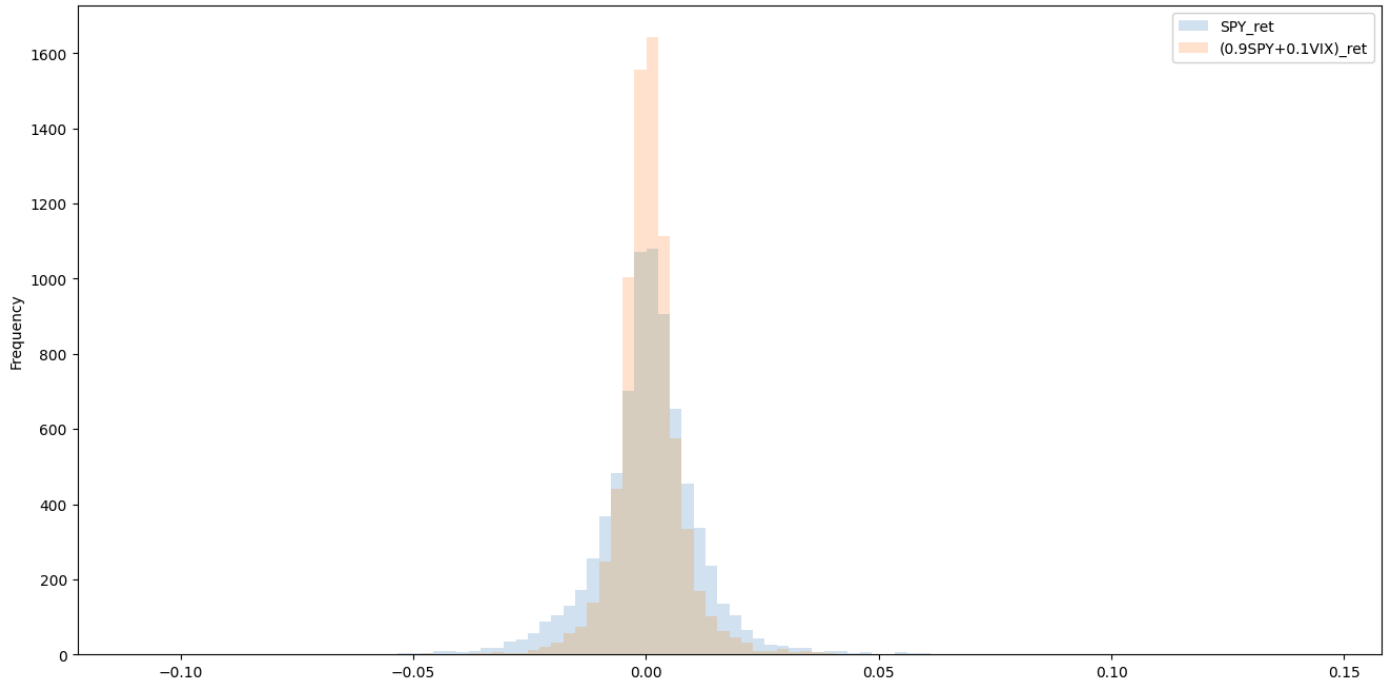
- It is always a good idea to plot your data and see what you observe.
- This is one area where the skills you have developed in DSO 545 and GSBA 545 can be deployed.
  - Summary statistics and hypothesis testing,
  - time series and histogram visualizations,
  - algorithmic exploratory data analysis - clustering, principal components or factor analysis, etc.,
  - etc.

```
# Plot a time series of the daily returns
spy_vix_returns[['SPY_ret', '(0.9SPY+0.1VIX)_ret']].plot(alpha=0.8)
```

```
<Axes: xlabel='Date'>
```

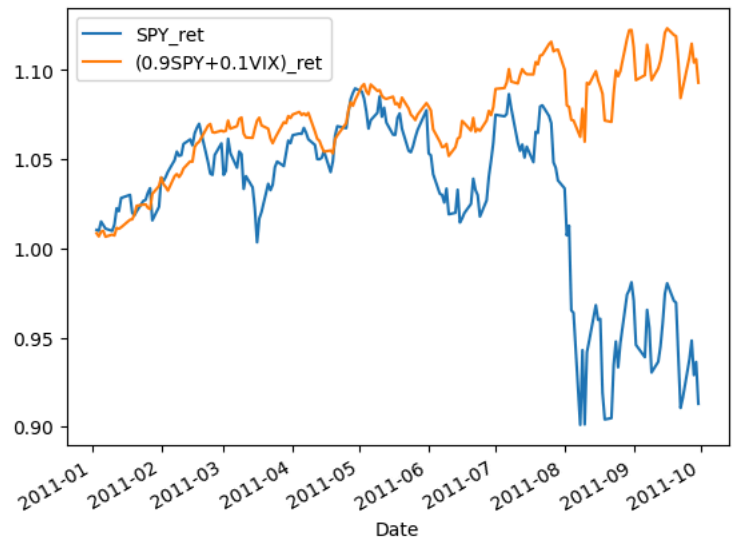
```
# Plot a histogram of the daily returns
spy_vix_returns[['SPY_ret', '(0.9SPY+0.1VIX)_ret']].plot(kind='hist', bins=100, figsize=(16,8), alpha=0.2)
```

```
<Axes: ylabel='Frequency'>
```



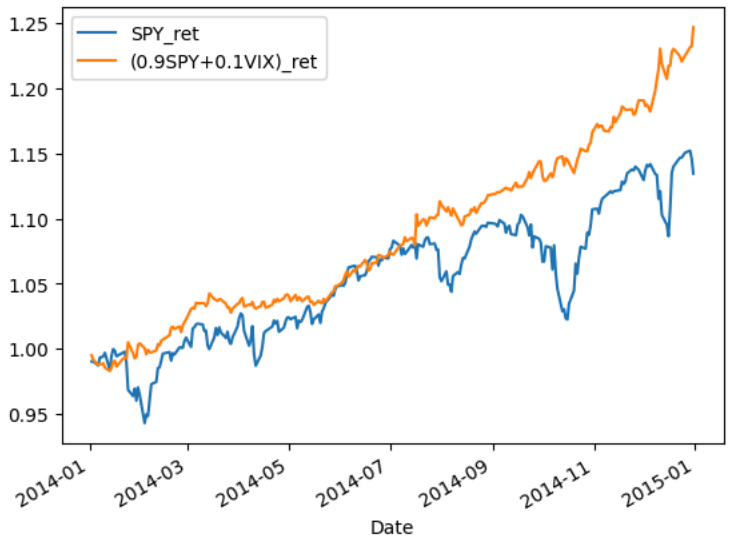

```
portfolio = spy_vix_returns[['SPY_ret', '(0.9SPY+0.1VIX)_ret']].copy()

portfolio.dropna(inplace=True)

# Period where US suffered credit rating downgrade by Standard and Poor's

(portfolio+1).loc['2011-01':'2011-09'].cumprod().plot()
```

```
<Axes: xlabel='Date'>
```



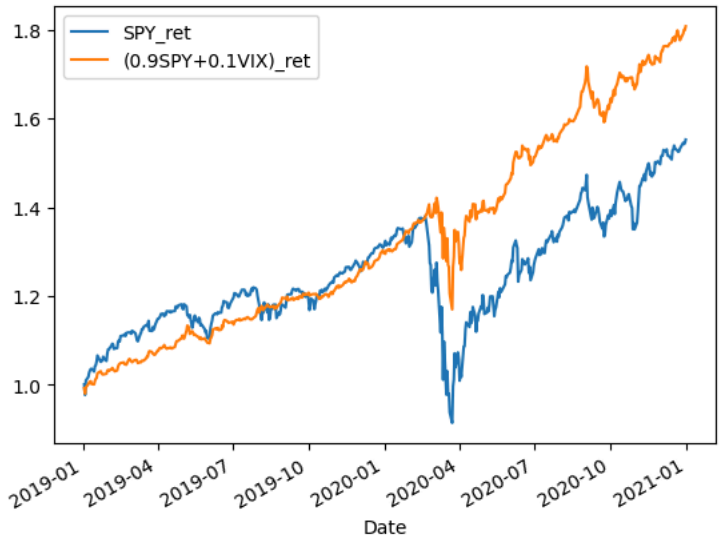

```
(portfolio+1).loc['2014-01':'2014-12'].cumprod().plot()
```

```
<Axes: xlabel='Date'>
```



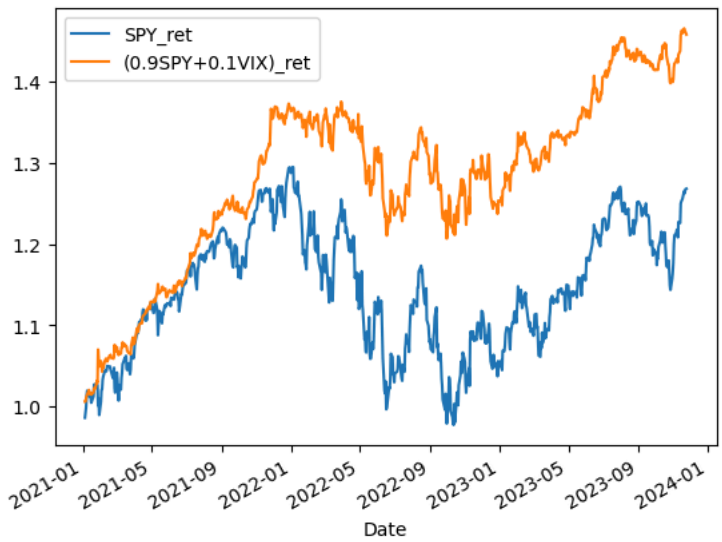

```
(portfolio+1).loc['2019-01':'2020-12'].cumprod().plot()
```

```
<Axes: xlabel='Date'>
```



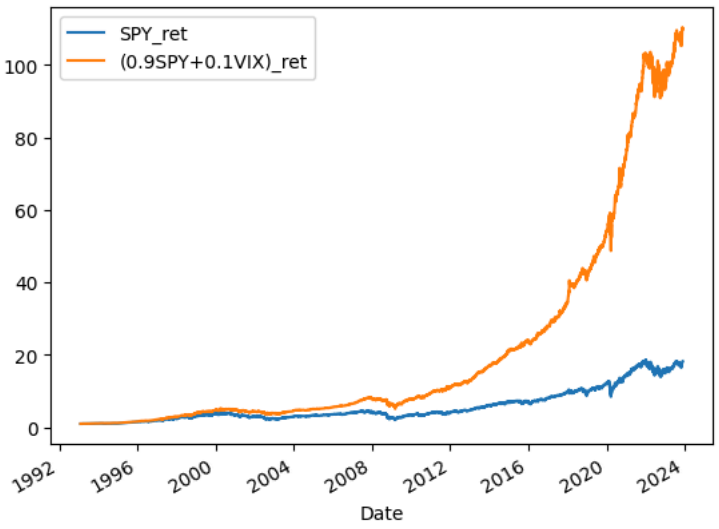

```
(portfolio+1).loc['2021-01':].cumprod().plot()
```

```
<Axes: xlabel='Date'>
```



```
# Plot the Gross Return of SPY and the SPY-VIX daily rebalanced portfolio for the full sample

(portfolio+1).cumprod().plot()
```

```
<Axes: xlabel='Date'>
```



Tradeable VIX-like Products

- We will download daily OHLC (Open-High-Low-Close) variables for the VIX, and its associated ETFs and ETNs that are supposed to benchmark to the VIX.
- This will show us how the current existing products are failing to adequately replicate the VIX index.

```
# VXXB = yf.download('VXXB', start='1980-09-10', end=datetime.date.today())|

vix = yf.download('^VIX', start='1980-09-10', end=datetime.date.today())

# UVXY is one of the worst ETFs in terms of decay.
# If you buy and hold, you are likely to lose a lot.
# Your $28.00 stock was once worth $1.461 million adjusted for reverse stock splits.

uvxy = yf.download('UVXY', start='1980-09-10', end=datetime.date.today())

vxx = yf.download('VXX', start='1980-09-10', end=datetime.date.today())

vixy = yf.download('VIXY', start='1980-09-10', end=datetime.date.today())

# The ETNs are linked to the daily return of the index and do not represent an investment in the VIX.
viix = yf.download('VIIX', start='1980-09-10', end=datetime.date.today())
```

```
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
```

```
# View a sample of the data
vix.head()
```

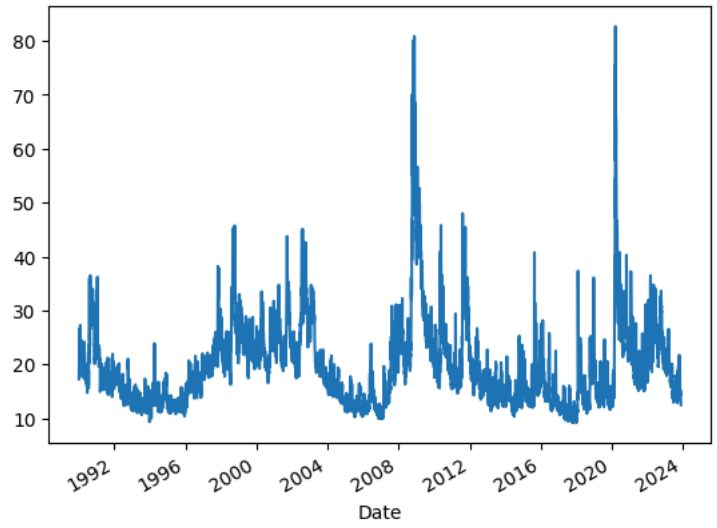
Date	Open	High	Low	Close	Adj Close	Volume
1990-01-02	17.240000	17.240000	17.240000	17.240000	17.240000	0
1990-01-03	18.190001	18.190001	18.190001	18.190001	18.190001	0
1990-01-04	19.219999	19.219999	19.219999	19.219999	19.219999	0
1990-01-05	20.110001	20.110001	20.110001	20.110001	20.110001	0
1990-01-08	20.260000	20.260000	20.260000	20.260000	20.260000	0

```
vxx.head()
```

Date	Open	High	Low	Close	Adj Close	Volume
2018-01-25	442.559998	442.559998	442.559998	442.559998	442.559998	0
2018-01-26	442.559998	442.559998	442.559998	442.559998	442.559998	0
2018-01-29	467.200012	473.279999	467.200012	473.279999	473.279999	88
2018-01-30	492.640015	508.959991	487.839996	488.799988	488.799988	3956
2018-01-31	479.359985	490.880005	479.359985	490.399994	490.399994	575

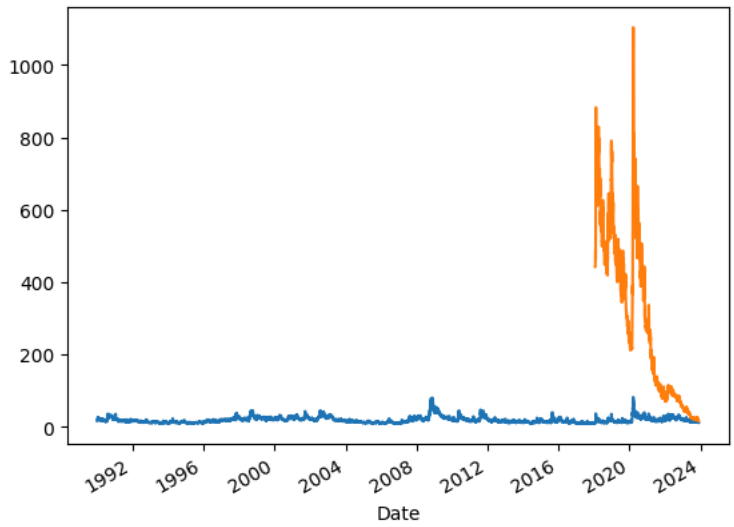
```
# Plot the VIX
vix['Adj Close'].plot()
```

```
<Axes: xlabel='Date'>
```



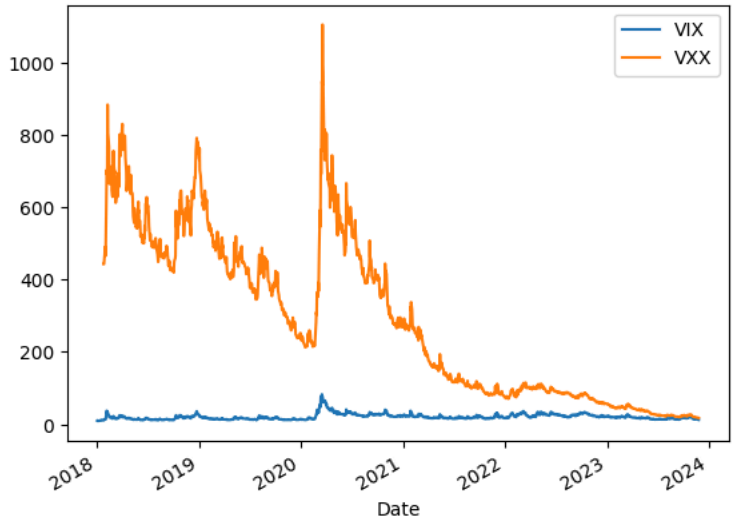

```
# Plot the full sample of the LEVEL
vix['Adj Close'].plot()

vxx['Adj Close'].plot()
```

```
<Axes: xlabel='Date'>
```



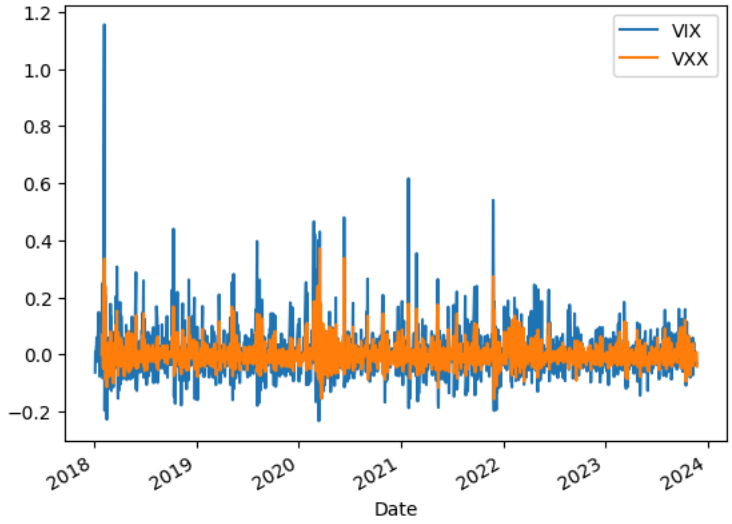

```
# Plot a sub-sample of the LEVEL
vix['Adj Close'].loc['2018':].plot(legend=True, label='VIX')

vxx['Adj Close'].loc['2018':].plot(legend=True, label='VXX')
```

```
<Axes: xlabel='Date'>
```




```
# # Plot the VIX and the normalized VXX
# vix['Adj Close'].loc['2018':].plot(legend=True, label='VIX')

# # VXX is normalized by the inital value of VIX from the subsample
# (vxx['Adj Close'].loc['2018':]/vix['Adj Close'].loc['2018':].iloc[0]).plot(legend=True, label='VXX')


# Plot a sub-sample of the RETURN
vix['Adj Close'].loc['2018':].pct_change().plot(legend=True, label='VIX')

vxx['Adj Close'].loc['2018':].pct_change().plot(legend=True, label='VXX')
```

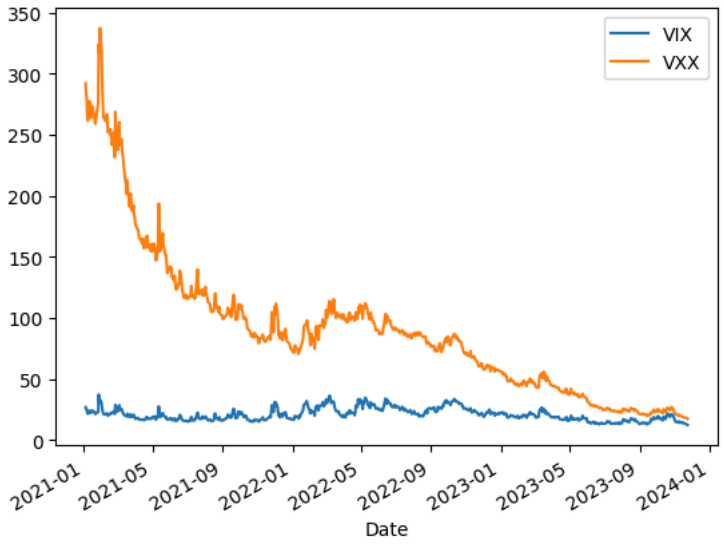
```
<Axes: xlabel='Date'>
```



- We will do further sub-sample analysis to investigate the tracking quality of the VXX (in terms of both the level and percentage change of the VIX) during different time periods.

```
# Plot a different sub-sample of the LEVEL
vix['Adj Close'].loc['2021':].plot(legend=True, label='VIX')

vxx['Adj Close'].loc['2021':].plot(legend=True, label='VXX')
```

```
<Axes: xlabel='Date'>
```



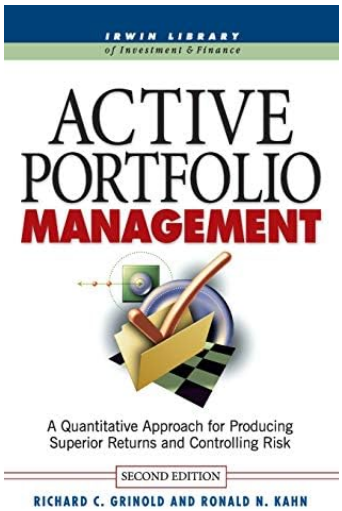

```
# Plot a different sub-sample of the RETURN
vix['Adj Close'].pct_change().loc['2021':].plot(legend=True, label='VIX')

vxx['Adj Close'].pct_change().loc['2021':].plot(legend=True, label='VXX')
```

```
<Axes: xlabel='Date'>
```

- How can we attempt to find a better alternative?

CBOE Historical Data

- CBOE Futures Historical Data
- CBOE VIX Futures contract specifications
- CBOE VIX products historical archived data



Portfolio Management Relative To a Benchmark

- In an investment portfolio, the *security selection* problem is concerned with determining the holdings of specific securities within a given *investable universe*.
- It is customary to manage and evaluate the portfolio of securities **relative** to some predefined **benchmark portfolio** that represents a particular investable universe.
- The **benchmark portfolio** provides a reference point for the construction of a portfolio.
- The management of a portfolio of securities relative to a benchmark could be *passive* or *active*.
- The goal of *passive* security portfolio management is to **replicate** the benchmark and the goal of the *active* security portfolio management is to **beat** the benchmark.

- Our goal is to create a *passive* portfolio that adequately tracks the VIX index (as opposed to an *active* portfolio that trys to beat the VIX index) using only *primitive* securities, such as stocks, (as opposed to futures or other derivatives).
  - This is a very vague objective!
  - Our task is to **quantify** this goal.

□

What's the Investable Universe?

- The **investable universe** is the collection of all tradable assets (stocks, bonds, futures, options, etc.) that are going to make up our portfolio that is designed to track the benchmark.
- Our formulation will crucially depend on how the tracking portfolio's investable universe compares to the benchmark's investable universe.
- In general, the benchmark does not need to be constructed from an universe of investable securities, it is simply a number that represents some quantity.
- In particular, assume we want to construct a portfolio from an investable universe of n securities with percentage holdings

$$\bar{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}.$$

□

Active Management Formulation

- Recall:
  - We can define a return as the relative price change, or growth rate, of the share price
    $$r_{i,t} = \frac{P_{i,t} - P_{i,t-1}}{P_{i,t-1}}$$
  - Fix a point in time t, and drop the time subscript from the returns.
  - We can define a portfolio as a linear combination of individual securities' returns
    $$r_p = w_1 r_1 + w_2 r_2 + \cdots w_n r_n = \bar{w}^T \bar{r},$$
    where n is the total number of securities in your investable universe and w_i is the percentage of your capital that you want to invest in the i^{th} security.
  - The expected rate of return of the portfolio is given by
    $$\mathbb{E}[r_p] = \sum_{i=1}^{n} w_i \mathbb{E}[r_i] = \bar{w}^T \bar{\mu},$$
    where $\bar{w}$ is the vector of weights and $\bar{\mu}$ is vector of expected returns.

- **Assume the benchmark is constructed from the same investable universe as your managed portfolio.**

- Consider a benchmark with returns
  $$r_b = w_1^b r_1 + w_2^b r_2 + \cdots w_n^b r_n = \bar{w_b}^T \bar{r}.$$

- Then the difference between the portfolio return and the benchmark return
  $$r_p - r_b = \bar{w_b}^T \bar{r} - \bar{w_b}^T \bar{r} = (\bar{w} - \bar{w_b})^T \bar{r}$$
  is known as the **active return**, which is the return relative to the benchmark portfolio.

- The vector
  $$\bar{w} - \bar{w_b}$$
  is known as the **active holdings** of the portfolio.

- If we assume the variance to be our measure of *risk*, then we can consider the variance of the active return, known as **active risk** or **tracking error**
  $$Var(r_p - r_b) = Var((\bar{w} - \bar{w_b})^T \bar{r}) = \sum_{i=1}^{n} \sum_{j=1}^{n} (w_i - w_i^b)(w_j - w_j^b) Cov(r_i, r_j)$$
  $$\Updownarrow$$
  $$Var(r_b - r_p) = Var((\bar{w} - \bar{w_b})^T \bar{r}) = (\bar{w} - \bar{w_b})^T \Sigma (\bar{w} - \bar{w_b}),$$
  where $(\bar{w} - \bar{w_b})$ is the vector of active weights and Σ is the matrix of covariances of returns.

- $$Var(r_b - r_p) = Var((\bar{w} - \bar{w_b})^T \bar{r}) = (\bar{w} - \bar{w_b})^T \Sigma (\bar{w} - \bar{w_b})$$
  $$= \underbrace{\begin{bmatrix} w_1 - w_1^b & w_2 - w_2^b & \cdots & w_n - w_n^b \end{bmatrix}}_{\substack{(\bar{w} - \bar{w_b})^T \\ (1 \times n)}} \underbrace{\begin{bmatrix} Cov(r_1, r_1) & Cov(r_1, r_2) & \cdots & Cov(r_1, r_n) \\ \vdots & \vdots & \cdots & \vdots \\ Cov(r_n, r_1) & Cov(r_n, r_2) & \vdots & Cov(r_n, r_n) \end{bmatrix}}_{\substack{\Sigma \\ (n \times n)}} \underbrace{\begin{bmatrix} w_1 - w_1^b \\ w_2 - w_2^b \\ \vdots \\ w_n - w_n^b \end{bmatrix}}_{\substack{(\bar{w} - \bar{w_b}) \\ (n \times 1)}}$$

- We can construct a wide range of goals and constraints to achieve a portfolio replication that satisfies active return and tracking error goals and constraints.

- Be aware that different formulations have different qualities of solutions, so additional experimentation will be warranted.

- Also, what if your benchmark isn't constructed from the same set of investable securities that you want to use to form your managed portfolio?
- What if your benchmark isn't constructed from any set investable securities?
- **We need a reformulation!**

□

Least Squares Formulation

- **Assume the benchmark is NOT constructed from the same investable universe as your managed portfolio.**
- **In particular, assume the benchmark is just a number that we want to track.**
- One (non-unique) way of interpreting this vague objective is by *determining* a portfolio whose value is *close* to the benchmark that we want to track.
  - How do you *determine* a portfolio?
  - How do you stay *close* to a benchmark?
  - How do you measure *closeness*?
- The answers to these questions are entirely in our creative control!

- Assume we are trying to track the returns to the VIX (our benchmark), which we can denote as r_b, with some portfolio with returns r_p, for all time t.
- We decide to measure closeness between the tracker and benchmark portfolios as the daily sum of squared errors

$$SSE(r_p, r_b) = \sum_{t=1}^{T} (r_{p,t} - r_{b,t})^2 = \cdots = ||r_p - r_b||_2^2 = \cdots = ||R\bar{w} - r_b||_2^2,$$

where

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ r_{T,1} & r_{T,2} & \cdots & r_{T,n} \end{bmatrix}_{(\text{time} \times \text{assets})}.$$

is the matrix (table) of historical returns of all assets in the tracking portfolio's investable universe.

- Every column in R represents the history of a particular asset i, and every row in R represents the cross-section of asset returns at a particular time t.
- This formulation allows us to track a number using any investable universe, or set of securities, we want!

□

Constructing a Volatility Index Tracking Fund

- As mentioned before, this is a real problem that has been studied from various angles.
- Two such angles are: Tracking VIX with VIX Futures: Portfolio Construction and Performance, and Constructing Zero-Beta VIX Portfolios with Dynamic CAPM.
- We will be taking a less technical and more experimental approach, which should cause us to have low expectations of success; however, our approach can be iterated upon and extended in many ways.
- This problem gives a playground to apply all fields of analytics from the descriptive, to the predictive, and finally to the prescriptive!

Tracking the VIX with an Investable Universe

- We have to answer the following question:
    - **What products do we want to manage to track the VIX index?**
- Answering this question requires a lot of data analysis, as well as some domain knowledge.
- In the interest of time, we will look at a small subset of securities that we hope will have some signal.

Tracking the VIX with Futures

- The universe of futures would be a very good starting point, however, they are a more complicated product that requires more advanced data wrangling techniques.
- If you are up for the challenge, feel free to use these products as part of your investable universe!
    - I have downloaded data on these products from the CBOE, and have placed the files on the Google drive.

Tracking the VIX with Negative Beta Stocks

```
# https://www.suredividend.com/negative-beta-stocks/
# https://www.marketbeat.com/market-data/negative-beta-stocks/
# https://seekingalpha.com/news/3730901-inverse-qqq-etfs
```

- We will download daily OHLC (Open-High-Low-Close) variables for a few securities that we believe tend move in opposite directions as the S&P 500.
- We hope these securities are related to the VIX.

```
import datetime as dt


# PSQ, SH, CLX, TORM

# ProShares Short QQQ ETF
psq = yf.download('PSQ', start='1980-09-10', end=datetime.date.today())

# ProShares Short S&P500 ETF
sh = yf.download('SH', start='1980-09-10', end=datetime.date.today())

# Clorox Co
clx = yf.download('CLX', start='1980-09-10', end=datetime.date.today())

# TORM is a world-leading specialist carrier of energy and clean petroleum products
torm = yf.download('TORM', start='1980-09-10', end=datetime.date.today())
```

```
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
```

```
psq.head()
```

Date	Open	High	Low	Close	Adj Close	Volume
2006-06-21	274.480011	274.480011	274.480011	274.480011	226.441238	50
2006-06-22	276.519989	279.480011	276.519989	279.200012	230.335144	3650
2006-06-23	280.799988	280.799988	277.839996	279.160004	230.302124	4175
2006-06-26	279.239990	280.480011	278.760010	279.200012	230.335144	4050
2006-06-27	279.200012	284.799988	279.200012	284.600006	234.790024	15675

Tracking the VIX with High Volatility Stocks

```
# https://www.tradingview.com/markets/stocks-usa/market-movers-most-volatile/
```

- We will download daily OHLC (Open-High-Low-Close) variables for a few securities that we believe tend to move around a lot and have high volatility.
- We hope these securities are related to the VIX.

```
# AGBA, COMP, PIXY, VTYX

# AGBA Acquisition Ltd
agba = yf.download('AGBA', start='1980-09-10', end=datetime.date.today())
```

```
# Compass Inc
comp = yf.download('COMP', start='1980-09-10', end=datetime.date.today())

# ShiftPixy Inc
pixy = yf.download('PIXY', start='1980-09-10', end=datetime.date.today())

# Ventyx Biosciences, Inc.
vtyx = yf.download('VTYX', start='1980-09-10', end=datetime.date.today()) #PLXP
```

```
[**********************100%%**********************]  1 of 1 completed
[**********************100%%**********************]  1 of 1 completed
[**********************100%%**********************]  1 of 1 completed
[**********************100%%**********************]  1 of 1 completed
```

```
agba.head()
```

Date	Open	High	Low	Close	Adj Close	Volume
2019-07-31	9.9	9.9	9.9	9.9	9.9	400
2019-08-01	9.9	9.9	9.9	9.9	9.9	0
2019-08-02	10.0	10.0	10.0	10.0	10.0	100
2019-08-05	10.0	10.0	10.0	10.0	10.0	200
2019-08-06	10.0	10.0	10.0	10.0	10.0	0

Tracking the VIX with High Correlation with VIX Stocks

```
# https://www.businessinsider.com/investing-101-five-profitable-stocks-with-a-strong-vix-correlation-2011-9
# https://seekingalpha.com/article/823781-10-high-yield-dividend-stocks-with-positive-vix-correlation
```

- We will download daily OHLC (Open-High-Low-Close) variables for a few securities that we believe tend move in the same directions as the VIX index.
- We hope these securities are related to the VIX.

```
# DG, GME, MCY, KR

# Dollar General
dg = yf.download('DG', start='1980-09-10', end=datetime.date.today())

# GameStop Corp.
gme = yf.download('GME', start='1980-09-10', end=datetime.date.today())

# Mercury General Corp
mcy = yf.download('MCY', start='1980-09-10', end=datetime.date.today())

# # Credit Suisse Group AG
# cs = yf.download('CS', start='1980-09-10', end=datetime.date.today())

# Kroger
kr = yf.download('KR', start='1980-09-10', end=datetime.date.today())
```

```
[**********************100%%**********************]  1 of 1 completed
[**********************100%%**********************]  1 of 1 completed
[**********************100%%**********************]  1 of 1 completed
[**********************100%%**********************]  1 of 1 completed
```

```
kr.head()
```

Date	Open	High	Low	Close	Adj Close	Volume
1980-09-10	1.304688	1.351563	1.304688	1.335938	0.118091	265600
1980-09-11	1.335938	1.343750	1.312500	1.312500	0.116020	340800
1980-09-12	1.312500	1.328125	1.296875	1.328125	0.117401	108800
1980-09-15	1.328125	1.328125	1.296875	1.312500	0.116020	611200
1980-09-16	1.312500	1.367188	1.312500	1.367188	0.120854	334400

Constructing the Investable Universe Returns Data Set

- R is the matrix (table) of historical returns of all assets in the tracking portfolio's investable universe, that is:

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ r_{T,1} & r_{T,2} & \cdots & r_{T,n} \end{bmatrix}_{(\text{time} \times \text{assets})}.$$

- Every column in R represents the history of a particular asset i, and every row in R represents the cross-section of asset returns at a particular time t.

□

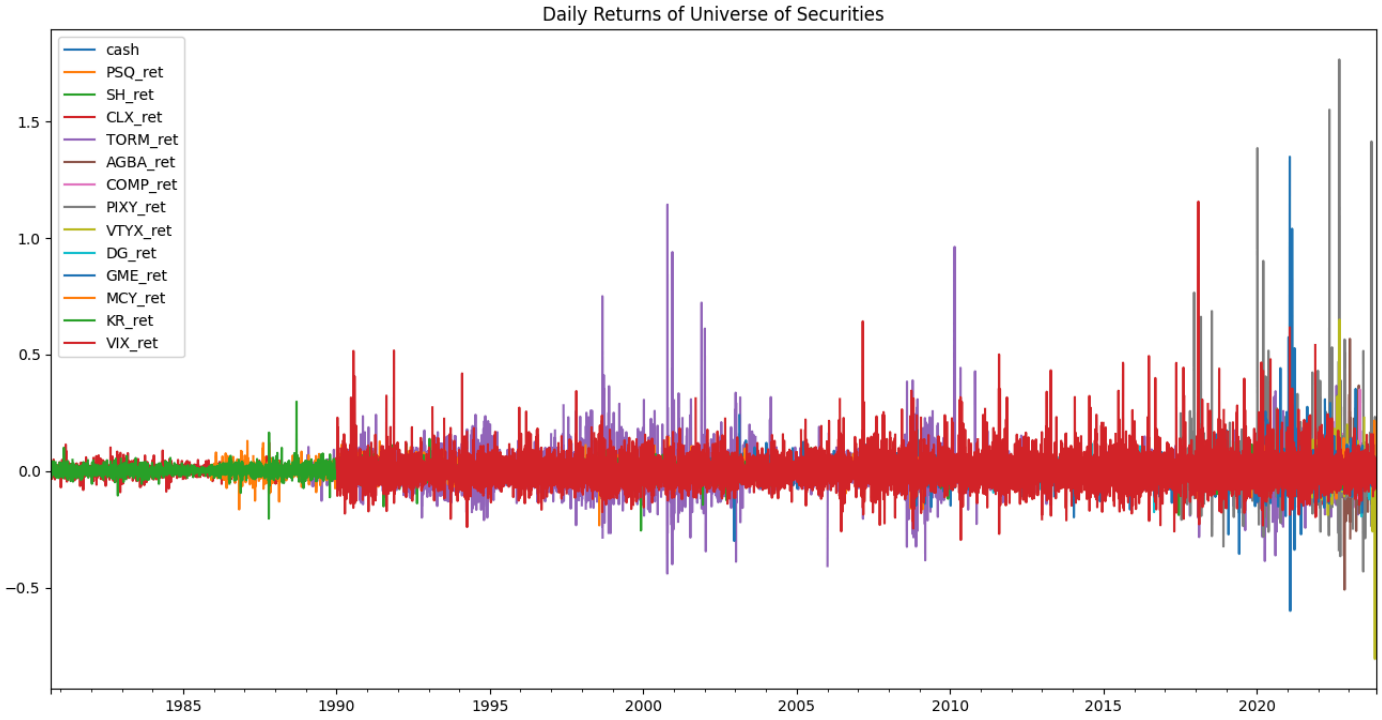
```
today = datetime.date.today()

two_years_ago = today - datetime.timedelta(2*365)

returns = pd.DataFrame({'cash':0}, index=pd.date_range('1980-09-10', today, freq='b'))

ticker_symbols = ['PSQ', 'SH', 'CLX', 'TORM',
                  'AGBA', 'COMP', 'PIXY', 'VTYX',
                  'DG', 'GME', 'MCY', 'KR',
                  'VIX']

ticker_data = [psq, sh, clx, torm,
               agba, comp, pixy, vtyx,
               dg, gme, mcy, kr,
               vix]

for ticker, data in zip(ticker_symbols, ticker_data):
    returns[ticker+'_ret'] = data['Adj Close'].pct_change()
```

- It is always a good idea to plot your data and see what you observe.
- This is one area where the skills you have developed in DSO 545 and GSBA 545 can be deployed.
  - Summary statistics and hypothesis testing,
  - time series and histogram visualizations,
  - algorithmic exploratory data analysis - clustering, principal components or factor analysis, etc.,

- etc.

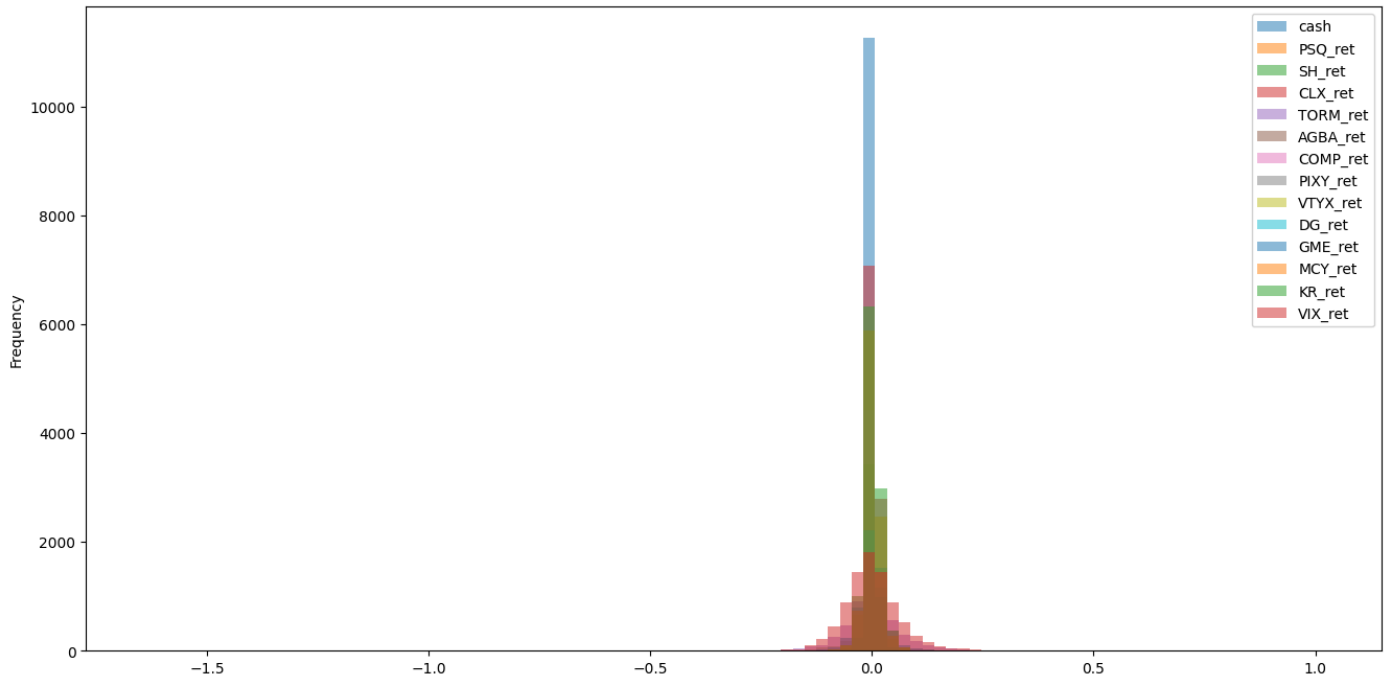
```
returns.plot(figsize=(16,8), legend=True)

plt.title('Daily Returns of Universe of Securities')
```

```
Text(0.5, 1.0, 'Daily Returns of Universe of Securities')
```




```
np.log(returns+1).plot(kind='hist', bins=100, alpha=0.5, figsize=(16,8))
```

```
<Axes: ylabel='Frequency'>
```




```
mean_variance = pd.DataFrame({'Mean': returns.mean(),
                              'Volatility': returns.std()})

mean_variance
```

	Mean	Volatility
cash	0.000000	0.000000
PSQ_ret	-0.000610	0.014024
SH_ret	-0.000384	0.012548
CLX_ret	0.000657	0.016089
TORM_ret	0.001397	0.060549
AGBA_ret	-0.001352	0.053505
COMP_ret	-0.001879	0.055977
PIXY_ret	-0.001938	0.118973
VTYX_ret	-0.000891	0.068759
DG_ret	0.000654	0.016848
GME_ret	0.001489	0.050431
MCY_ret	0.000538	0.019502
KR_ret	0.000732	0.019395
VIX_ret	0.002261	0.069968

- The see the expected return and variance of the three securities for a single day, as well as the cross-correlations between the securities.

- Recall, the correlation between two variables (X_i, X_j) is defined as

$$\rho_{X_i,X_j} = \frac{Cov(X_i, X_j)}{\sqrt{Var(X_i)}\sqrt{Var(X_j)}},$$

and measures the extent to which the two variables **linearly** move together.

- This is far from what you would use in practice, and there are so many ways to innovate on obtaining these estimates using analytics techniques, both old and new.
- How might this change if we used different horizons?
- How can you think of these estimates (pros and cons) and how they're going to be used from the different perspectives of
  - descriptive analytics,
  - predictive analytics,
  - prescriptive analytics,
  - and business development and strategy, as well as product management?

□

```
corr_matrix = returns.corr()

corr_matrix
```

	cash	PSQ_ret	SH_ret	CLX_ret	TORM_ret	AGBA_ret	COMP_ret	PIXY_ret	VTYX_ret	DG_ret	GME_ret	MCY_ret	KR_
cash	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I
PSQ_ret	NaN	1.000000	0.917930	-0.344679	-0.063027	-0.001278	-0.506306	-0.147021	-0.213147	-0.348524	-0.239343	-0.456192	-0.310
SH_ret	NaN	0.917930	1.000000	-0.394355	-0.070928	-0.004029	-0.498612	-0.119816	-0.218541	-0.376331	-0.247796	-0.566701	-0.367
CLX_ret	NaN	-0.344679	-0.394355	1.000000	0.010928	-0.020691	0.093877	0.000374	-0.005575	0.270895	0.140064	0.179342	0.207
TORM_ret	NaN	-0.063027	-0.070928	0.010928	1.000000	0.001211	-0.026862	-0.027856	-0.035295	0.010048	0.026928	0.030680	0.034
AGBA_ret	NaN	-0.001278	-0.004029	-0.020691	0.001211	1.000000	0.041204	0.016784	0.042616	-0.022449	0.006805	-0.039148	0.025
COMP_ret	NaN	-0.506306	-0.498612	0.093877	-0.026862	0.041204	1.000000	0.182395	0.182976	0.132552	0.336313	0.159799	0.072
PIXY_ret	NaN	-0.147021	-0.119816	0.000374	-0.027856	0.016784	0.182395	1.000000	0.011464	0.021935	0.058157	0.031471	0.002
VTYX_ret	NaN	-0.213147	-0.218541	-0.005575	-0.035295	0.042616	0.182976	0.011464	1.000000	0.026701	0.170133	0.010465	0.015
DG_ret	NaN	-0.348524	-0.376331	0.270895	0.010048	-0.022449	0.132552	0.021935	0.026701	1.000000	0.075212	0.192346	0.302
GME_ret	NaN	-0.239343	-0.247796	0.140064	0.026928	0.006805	0.336313	0.058157	0.170133	0.075212	1.000000	0.151021	0.156
MCY_ret	NaN	-0.456192	-0.566701	0.179342	0.030680	-0.039148	0.159799	0.031471	0.010465	0.192346	0.151021	1.000000	0.183
KR_ret	NaN	-0.310698	-0.367656	0.207415	0.034574	0.025500	0.072128	0.002193	0.015132	0.302584	0.156886	0.183365	1.000
VIX_ret	NaN	0.692248	0.719872	-0.239571	-0.064362	-0.058974	-0.342072	-0.120684	-0.141173	-0.284476	-0.147354	-0.321839	-0.241

```
covariance_matrix = returns.cov()

covariance_matrix
```

	cash	PSQ_ret	SH_ret	CLX_ret	TORM_ret	AGBA_ret	COMP_ret	PIXY_ret	VTYX_ret	DG_ret	GME_ret	MCY_ret
cash	0.0	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000
PSQ_ret	0.0	0.000197	0.000162	-6.471634e-05	-0.000047	-0.000001	-0.000432	-2.678582e-04	-0.000242	-0.000076	-0.000182	-0.000119
SH_ret	0.0	0.000162	0.000157	-6.625001e-05	-0.000048	-0.000003	-0.000317	-1.796988e-04	-0.000184	-0.000070	-0.000169	-0.000132
CLX_ret	0.0	-0.000065	-0.000066	2.588547e-04	0.000010	-0.000019	0.000090	7.272530e-07	-0.000007	0.000060	0.000094	0.000053
TORM_ret	0.0	-0.000047	-0.000048	9.993038e-06	0.003666	0.000004	-0.000088	-1.994924e-04	-0.000142	0.000008	0.000071	0.000035
AGBA_ret	0.0	-0.000001	-0.000003	-1.940666e-05	0.000004	0.002863	0.000157	1.192580e-04	0.000225	-0.000022	0.000036	-0.000046
COMP_ret	0.0	-0.000432	-0.000317	9.020880e-05	-0.000088	0.000157	0.003133	1.500881e-03	0.000765	0.000142	0.001058	0.000188
PIXY_ret	0.0	-0.000268	-0.000180	7.272530e-07	-0.000199	0.000119	0.001501	1.415459e-02	0.000128	0.000046	0.000571	0.000077
VTYX_ret	0.0	-0.000242	-0.000184	-6.908611e-06	-0.000142	0.000225	0.000765	1.282081e-04	0.004728	0.000038	0.000642	0.000016
DG_ret	0.0	-0.000076	-0.000070	6.040167e-05	0.000008	-0.000022	0.000142	4.578367e-05	0.000038	0.000284	0.000074	0.000054
GME_ret	0.0	-0.000182	-0.000169	9.407047e-05	0.000071	0.000036	0.001058	5.710946e-04	0.000642	0.000074	0.002543	0.000134
MCY_ret	0.0	-0.000119	-0.000132	5.349686e-05	0.000035	-0.000046	0.000188	7.655605e-05	0.000016	0.000054	0.000134	0.000380

```
mean_variance_correlations = pd.concat([mean_variance, corr_matrix], axis=1)

mean_variance_correlations
```

	Mean	Volatility	cash	PSQ_ret	SH_ret	CLX_ret	TORM_ret	AGBA_ret	COMP_ret	PIXY_ret	VTYX_ret	DG_ret	GM
cash	0.000000	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
PSQ_ret	-0.000610	0.014024	NaN	1.000000	0.917930	-0.344679	-0.063027	-0.001278	-0.506306	-0.147021	-0.213147	-0.348524	-0.
SH_ret	-0.000384	0.012548	NaN	0.917930	1.000000	-0.394355	-0.070928	-0.004029	-0.498612	-0.119816	-0.218541	-0.376331	-0.
CLX_ret	0.000657	0.016089	NaN	-0.344679	-0.394355	1.000000	0.010928	-0.020691	0.093877	0.000374	-0.005575	0.270895	0.
TORM_ret	0.001397	0.060549	NaN	-0.063027	-0.070928	0.010928	1.000000	0.001211	-0.026862	-0.027856	-0.035295	0.010048	0.
AGBA_ret	-0.001352	0.053505	NaN	-0.001278	-0.004029	-0.020691	0.001211	1.000000	0.041204	0.016784	0.042616	-0.022449	0.
COMP_ret	-0.001879	0.055977	NaN	-0.506306	-0.498612	0.093877	-0.026862	0.041204	1.000000	0.182395	0.182976	0.132552	0.
PIXY_ret	-0.001938	0.118973	NaN	-0.147021	-0.119816	0.000374	-0.027856	0.016784	0.182395	1.000000	0.011464	0.021935	0.
VTYX_ret	-0.000891	0.068759	NaN	-0.213147	-0.218541	-0.005575	-0.035295	0.042616	0.182976	0.011464	1.000000	0.026701	0.
DG_ret	0.000654	0.016848	NaN	-0.348524	-0.376331	0.270895	0.010048	-0.022449	0.132552	0.021935	0.026701	1.000000	0.
GME_ret	0.001489	0.050431	NaN	-0.239343	-0.247796	0.140064	0.026928	0.006805	0.336313	0.058157	0.170133	0.075212	1.
MCY_ret	0.000538	0.019502	NaN	-0.456192	-0.566701	0.179342	0.030680	-0.039148	0.159799	0.031471	0.010465	0.192346	0.
KR_ret	0.000732	0.019395	NaN	-0.310698	-0.367656	0.207415	0.034574	0.025500	0.072128	0.002193	0.015132	0.302584	0.
VIX_ret	0.002261	0.069968	NaN	0.692248	0.719872	-0.239571	-0.064362	-0.058974	-0.342072	-0.120684	-0.141173	-0.284476	-0.

Decision Variables

- We want to construct a portfolio from an investable universe of n securities with percentage holdings

$$\bar{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}.$$

□

```
# We choose to work with a subset of the data that is more relevant to future performance
returns = returns.loc[two_years_ago:today, :].copy()
```

```
# Number of nulls per column
returns.isna().sum()
```

```
cash         0
PSQ_ret     20
SH_ret      20
CLX_ret     20
TORM_ret    20
AGBA_ret    20
COMP_ret    20
PIXY_ret    20
VTYX_ret    20
DG_ret      20
GME_ret     20
MCY_ret     20
KR_ret      20
VIX_ret     20
dtype: int64
```

```
# Drop the rows that have a null value
returns.dropna(axis=0, inplace=True)
```

```
returns.shape
```

```
(501, 14)
```

```
# Returns contains the target variable, VIX, which we don't want a weight for
weights = cp.Variable(returns.shape[1] - 1)
```

Objective

- Assume we are trying to track the returns to the VIX (our benchmark), which we can denote as r_b, with some portfolio with returns r_p, for all time t.
- We decide to measure closeness between the tracker and benchmark portfolios as the daily sum of squared errors.
- We want to choose the portfolio weights that minimizes the distance, or error, between the tracking portfolio and the VIX portfolio.
- $$\text{minimize: } SSE(r_p, r_b) = \sum_{t=1}^{T}(r_{p,t} - r_{b,t})^2 = ||R\bar{w} - r_b||_2^2$$

□

```
# Matrix multiply the returns data frame (without the VIX ) and the weights vector
returns.iloc[:, :-1].values @ weights
```

```
Expression(AFFINE, UNKNOWN, (501,))
```

```
# Define the error between the tracking portfolio and the VIX portfolio
error = returns.iloc[:,:-1].values @ weights - returns['VIX_ret'].values
```

```
# Compute the sum of squared errors between the tracking portfolio and the VIX portfolio
sum_of_squared_error = cp.sum_squares(error)
```

Constraints

- For simplicity, we will assume the sum of the weights must equal one.
- $$\sum_{i=1}^{n} w_i = \bar{w}^T \bar{\mathbb{1}} = 1.$$
- This allows for shorting and applying leverage, but typically keeps them from being too overweight in any given asset.
- There are plenty more constraints that we should consider on a second passing.

□

```
prob = cp.Problem(cp.Minimize(sum_of_squared_error), [sum(weights)==1])
```

Solution and Analysis

- We're seeking a prescription regarding how to build a minimum error tracking portfolio for the VIX index.

```
# Minimal error - needs to be compared with an existing product for reference, say VXX
prob.solve()
```

```
0.9011780388452199
```

```
# Prescribed portfolio weights for constructing the tracking portfolio
np.round(weights.value, 2)
```

```
array([-3.72, -0.84,  5.2 ,  0.2 ,  0.03, -0.08,  0.02, -0.  ,  0.01,
        0.2 , -0.07, -0.15,  0.2 ])
```

```
pd.DataFrame(np.round(weights.value, 2),
             index=returns.columns[:-1],
             columns=['weights'])
```

	weights
cash	-3.72

```
vix_returns = returns['VIX_ret']
vxx_returns = vxx.loc[returns.index, 'Adj Close'].pct_change()

error_between_vix_vxx = ((vix_returns - vxx_returns)**2).sum()

error_between_vix_vxx
```

```
0.7418504925368637
```

COMP_ret	0.02

```
dates = returns.index

# Performance of IN-SAMPLE tracking portfolio
tracking_portfolio = pd.Series(returns.iloc[:, :-1].values @ weights.value,
                               index=dates)
```

GME_ret	-0.07

```
plt.figure(figsize=(16,8))

vix_returns.plot(label='VIX', legend=True, alpha=0.5)
vxx_returns.plot(label='VXX', legend=True, alpha=0.5)
tracking_portfolio.plot(label='Tracking Port', legend=True, alpha=0.5)
```