Diebold-Li/Black-Litterman

Jack

Rationale for Diebold-Li

- The Nelson-Siegel (1987) yield curve is modelled by:

$$y_t(\tau) = \beta_{1t} + \beta_{2t}\left(\frac{1 - e^{-\lambda_t \tau}}{\lambda_t \tau}\right) + \beta_{3t}\left(\frac{1 - e^{-\lambda_t \tau}}{\lambda_t \tau} - e^{-\lambda_t \tau}\right).$$

- This is a static curve that does not well capture the dynamic structure of yields

- Diebold-Li suggests modelling the growth of the beta parameters, capturing the dynamic structure

$$y_t(\tau) = \beta_{1t} + \beta_{2t}\left(\frac{1 - e^{-\lambda_t \tau}}{\lambda_t \tau}\right) + \beta_{3t}\left(\frac{1 - e^{-\lambda_t \tau}}{\lambda_t \tau} - e^{-\lambda_t \tau}\right).$$

- Lambda is the decay rate, i.e. small lambda means slow decay, and governs beta3 maximum

- The loading on beta1 is constant 1, can be viewed as a long-term factor

- The loading on beta2 is $\left(\frac{1 - e^{-\lambda_t \tau}}{\lambda_t \tau}\right)$, a function that starts at 1 but decays monotonically and quickly to zero

- Beta2 can be viewed as short-term factor

- Beta3 starts at zero, increases, then decreases, and hence is viewed as medium-term

Claim: The Diebold-Li methodology performs better than traditional yield curve forecasting methods both in and out of sample

Goal: Replicate the study and validate the claim

Data

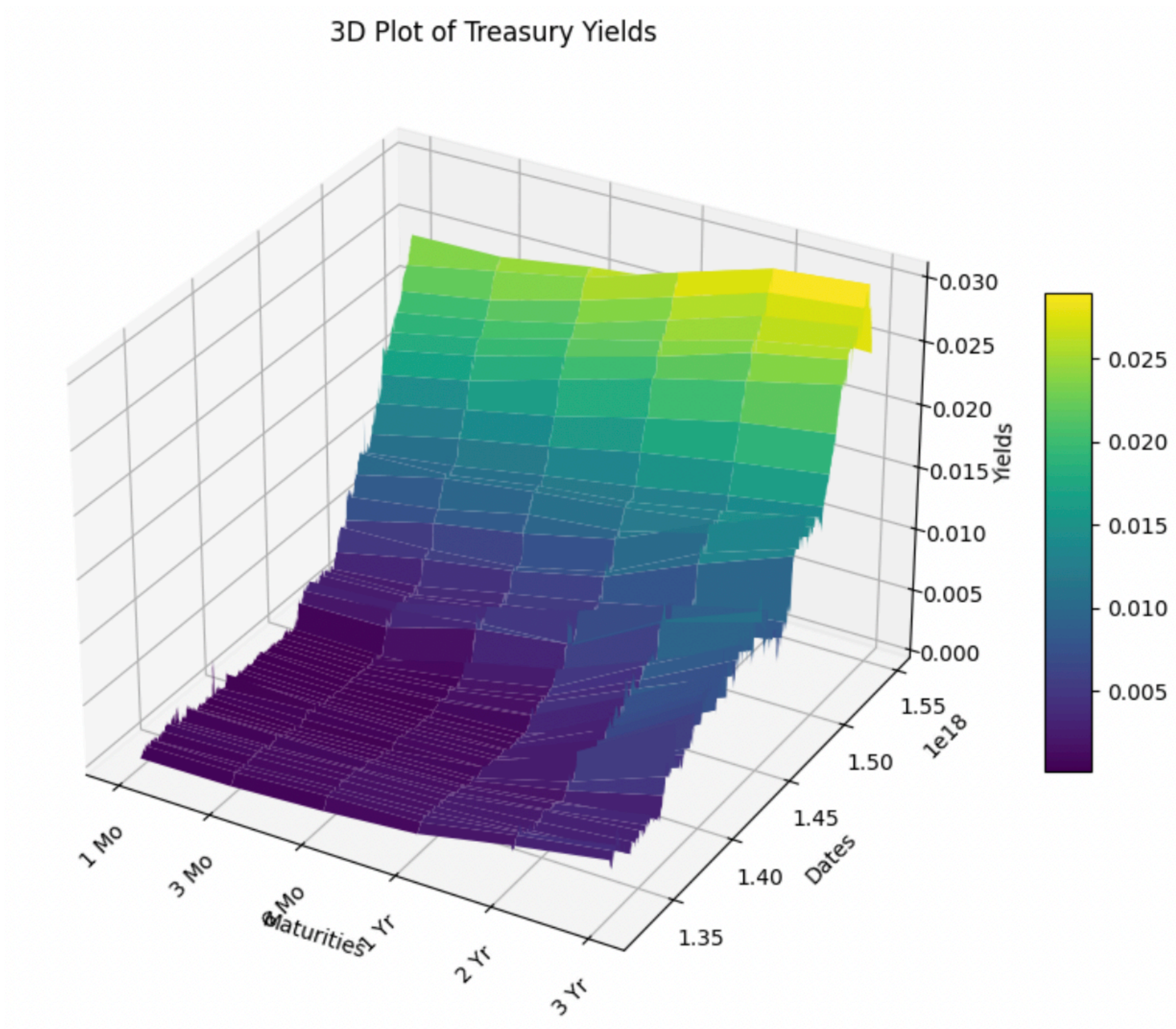
- Source: Daily Treasury Par Yield Curve Rates, U.S. Department of The Treasury
- Daily data from 2012-2018
- Out-of-sample testing data is from same source, 2019

Methodology

- 1: For each point in time, find the 3 beta parameters that fits the curve
- 2: Fit individual AR(1) models to each beta1, beta2, beta3
- 3: Simultaneously fit a VAR(1) model directly on the yields for comparison
- 4: For each method, forecast, graph, and find RMSE for out-of-sample predictions
- Note: lambda is set to 0.0609/365

```
data2012 = pd.read_csv("daily-treasury-rates (18).csv").iloc[::-1]
data2013 = pd.read_csv("daily-treasury-rates (11).csv").iloc[::-1]
data2014 = pd.read_csv("daily-treasury-rates (12).csv").iloc[::-1]
data2015 = pd.read_csv("daily-treasury-rates (13).csv").iloc[::-1]
data2016 = pd.read_csv("daily-treasury-rates (14).csv").iloc[::-1]
data2017 = pd.read_csv("daily-treasury-rates (15).csv").iloc[::-1]
data2018 = pd.read_csv("daily-treasury-rates (16).csv").iloc[::-1]
```

- Also initialize three columns with all zeroes for beta1, beta2, beta3

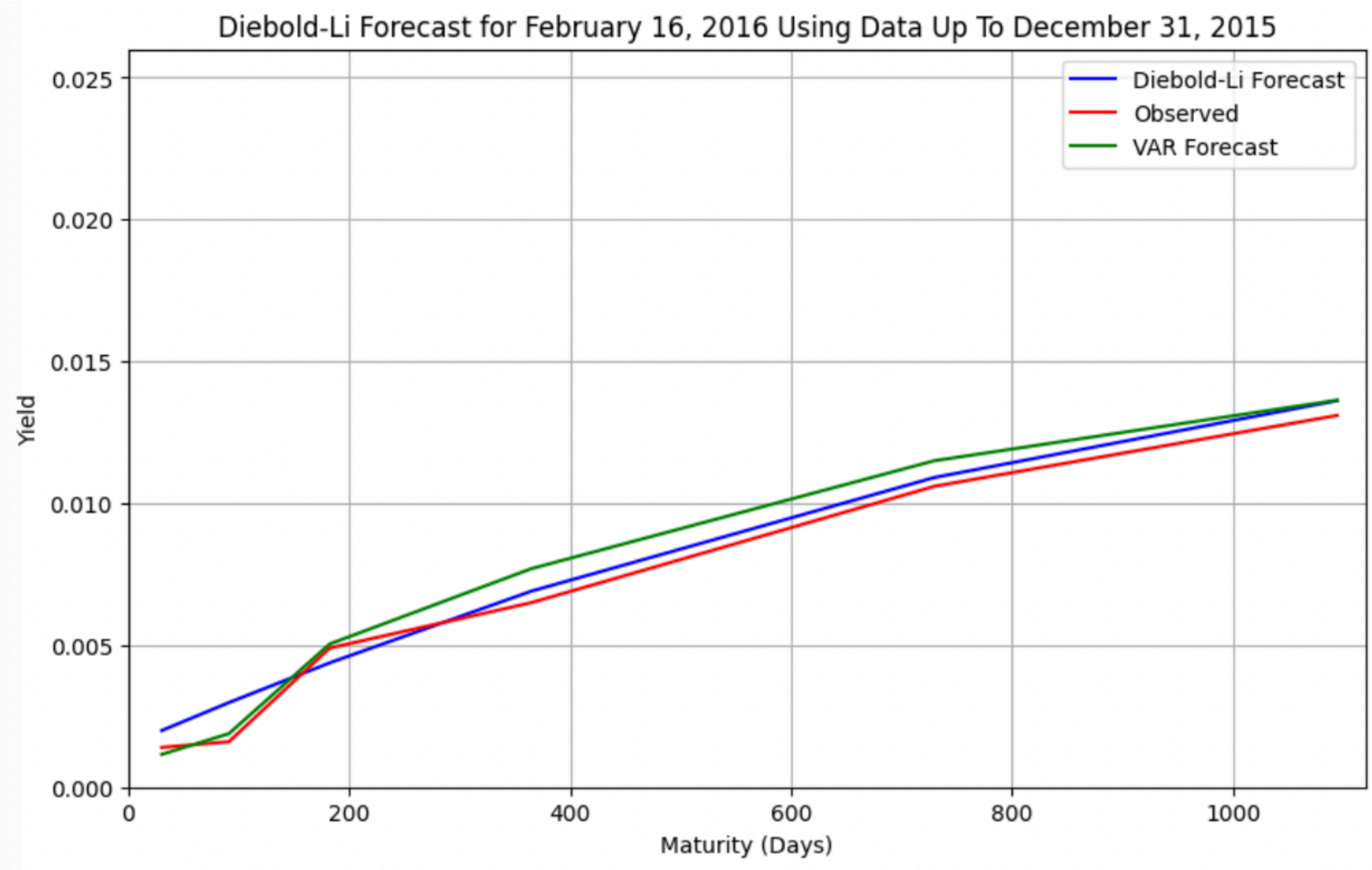

3D Plot of Treasury Yields
Yields
0.000
0.005
0.010
0.015
0.020
0.025
0.030
Dates
1e18
1.35
1.40
1.45
1.50
1.55
Maturities
1 Mo
3 Mo
6 Mo
1 Yr
2 Yr
3 Yr
0.005
0.010
0.015
0.020
0.025

```python
def NelsonSiegel(t,b1,b2,b3):
    lambd = 0.0609/365
    return b1 + b2*((1-np.exp(-lambd*t))/(lambd*t)) + b3*((1-np.exp(-lambd*t))/(lambd*t) - np.exp(-lambd*t))

def generateParams(df):
    for i in range(len(df)):
        params, pcov = optimize.curve_fit(NelsonSiegel,
            xdata = np.array([1,3,6,12,24,36]) * 30.4,
            ydata = df.iloc[i,1:7],
            p0 = [0.1,0.2,0.3])
        b1,b2,b3 = params
        df.at[i,'b1'] = b1
        df.at[i,'b2'] = b2
        df.at[i,'b3'] = b3
    return df.iloc[:,7:10].reset_index(drop=True)
```

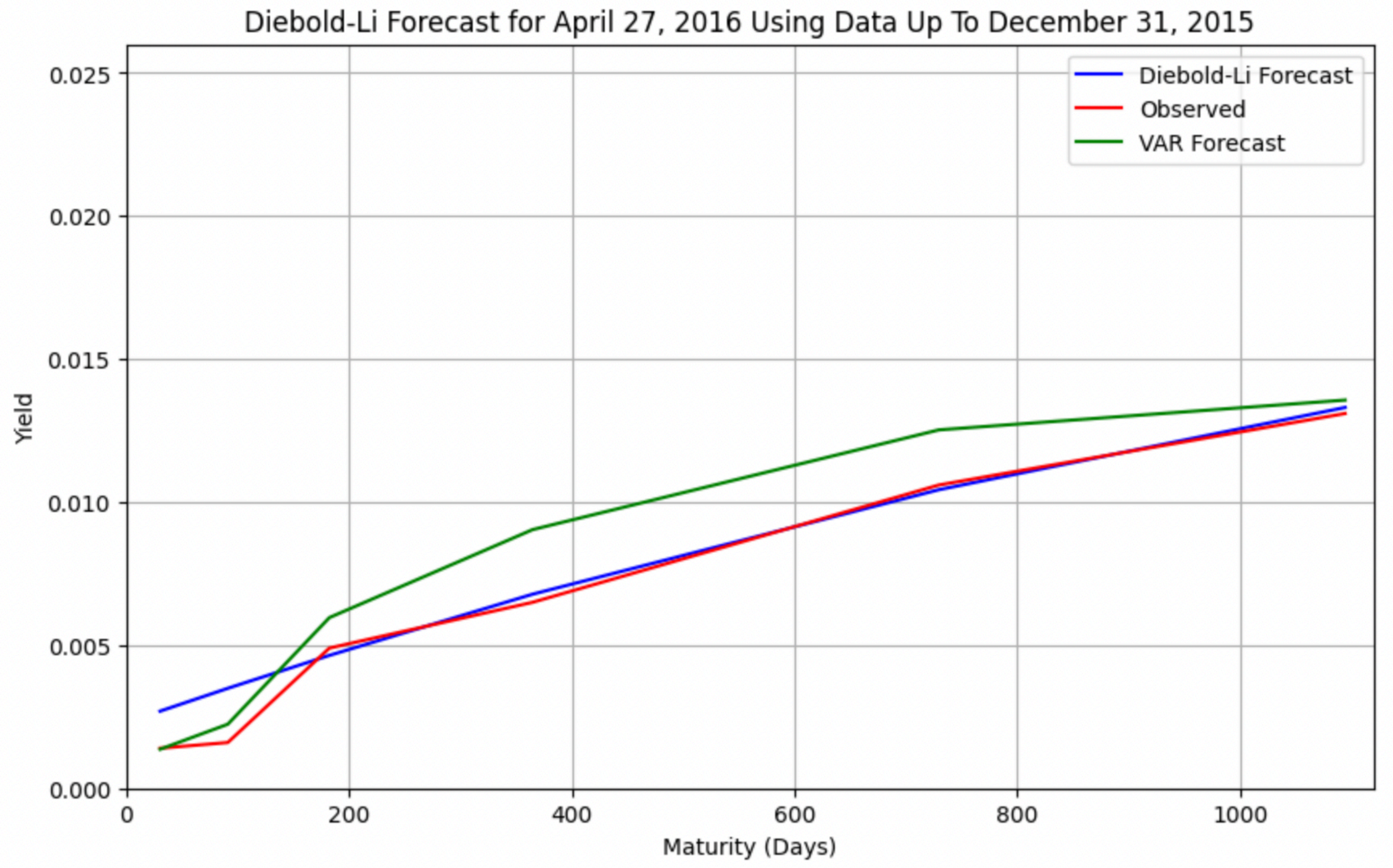
```python
result = adfuller(data['b1'])
print('p-val b1: ', result[1])

result = adfuller(data['b2'])
print('p-val b2: ', result[1])

result = adfuller(data['b3'])
print('p-val b3: ', result[1])
```

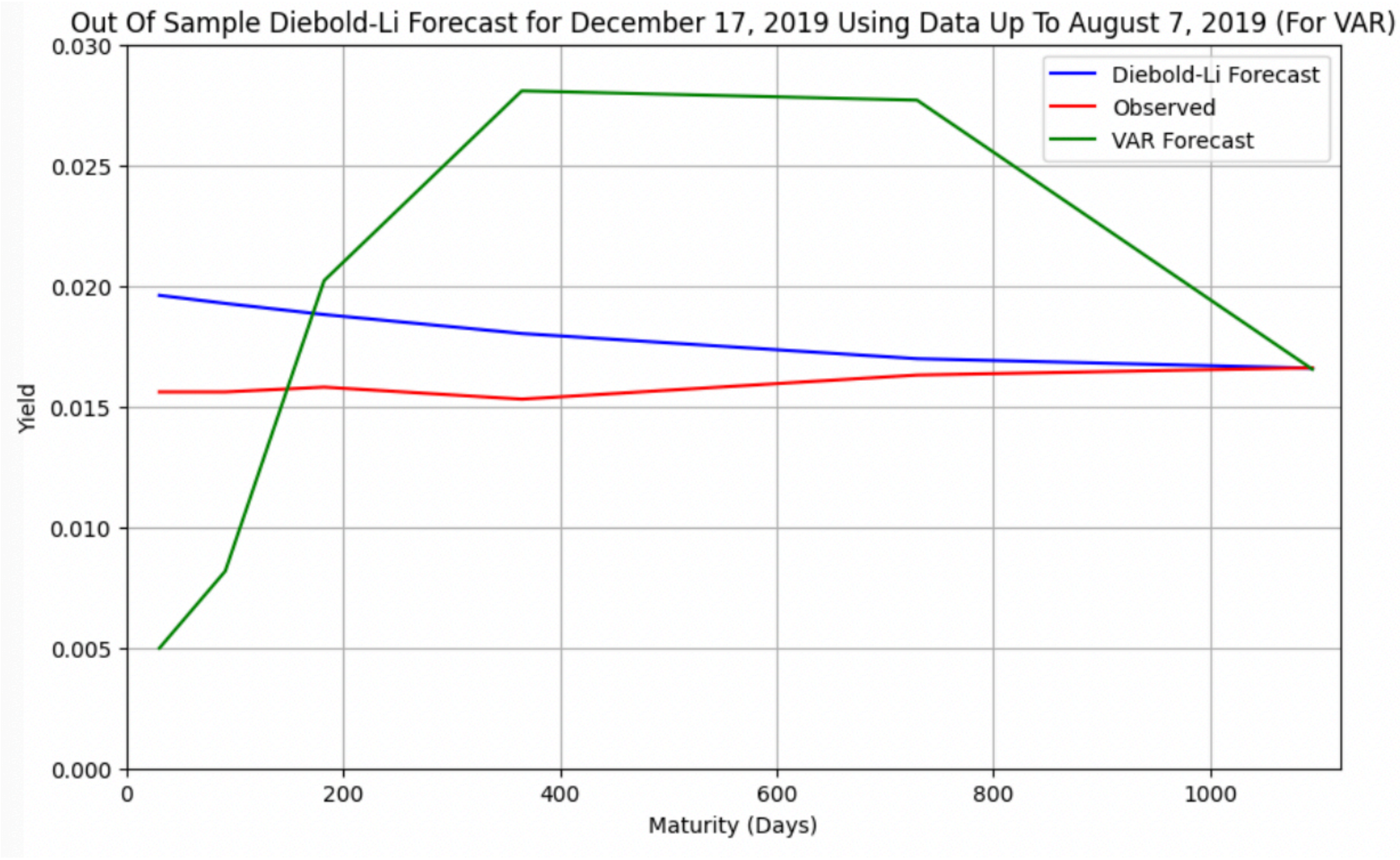
```
p-val b1:  0.6623063488365231
p-val b2:  0.6675027583701549
p-val b3:  0.650330917848073
```

- High autocorrelations in betas, which we expect
- High autocorrelation indicates persistence, strengthening the forecasting



Diebold-Li Forecast for February 16, 2016 Using Data Up To December 31, 2015
Diebold-Li Forecast
Observed
VAR Forecast
0.025
0.020
0.015
0.010
0.005
0.000
Yield
0
200
400
600
800
1000
Maturity (Days)


Diebold-Li Forecast for April 27, 2016 Using Data Up To December 31, 2015
Diebold-Li Forecast
Observed
VAR Forecast
Yield
0.025
0.020
0.015
0.010
0.005
0.000
0
200
400
600
800
1000
Maturity (Days)


Out Of Sample Diebold-Li Forecast for December 17, 2019 Using Data Up To August 7, 2019 (For VAR)
Diebold-Li Forecast
Observed
VAR Forecast
Yield
Maturity (Days)
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
200
400
600
800
1000

```python
warnings.simplefilter('ignore')

def RMSE(actual,pred):
    size = len(actual)
    accumulator = 0
    for i in range(len(actual)):
        accumulator += np.sqrt((actual[i] - pred[i])**2 / size)
    return accumulator

def MCSamplingRMSE(df,iterations):

    for i in range(iterations):
        ran = random.randint(20,219)
        VAR_rmse = 0
        DL_rmse = 0

        dfparams = generateParams(df)

        model_beta1 = AutoReg(dfparams['b1'][:ran],lags=1)
        fit_beta1 = model_beta1.fit()
        forecast_beta1 = fit_beta1.forecast(steps=30).iloc[-1]

        model_beta2 = AutoReg(dfparams['b2'][:ran],lags=1)
        fit_beta2 = model_beta2.fit()
        forecast_beta2 = fit_beta2.forecast(steps=30).iloc[-1]

        model_beta3 = AutoReg(dfparams['b3'][:ran],lags=1)
        fit_beta3 = model_beta3.fit()
        forecast_beta3 = fit_beta3.forecast(steps=30).iloc[-1]

        forecast_VAR = results.forecast(df.iloc[ran-1:ran,1:7].values,steps=30)[-1]
        forecast_DL = NelsonSiegel(np.array([1,3,6,12,24,36]) * 30.4,forecast_beta1,forecast_beta2,forecast_beta3)
        slice = df.iloc[ran+30,1:7]

        VAR_rmse += RMSE(slice,forecast_VAR)
        DL_rmse += RMSE(slice,forecast_DL)

    print("VAR out of sample RMSE: ", VAR_rmse)
    print("DL out of sample RMSE: ", DL_rmse)
```

```
MCSamplingRMSE(data2019,120)
```

```
VAR out of sample RMSE:  0.01609685312200426
DL out of sample RMSE:  0.005868541227778114
```

Black-Litterman Model

- Developed in 1990 by Fischer Black and Robert Litterman

- Takes a Bayesian approach to forecasting returns

- Given a set of stocks, the “prior” is the equilibrium return, and the “posterior” considers also investor views

- Key components:

- Key components:
  - Market capitalizations

- Key components:
    - Market capitalizations
    - Covariance shrinkage
        - “Technique to estimate covariance matrix of a set of variables when sample is small”

$$\Sigma_{\text{shrink}} = (1 - \lambda)\Sigma_{\text{sample}} + \lambda T$$

- Key components:
  - Market capitalizations
  - Covariance shrinkage
  - Market implied risk aversion
    - "Quantifies degree to which investors are willing to trade off risk and return"

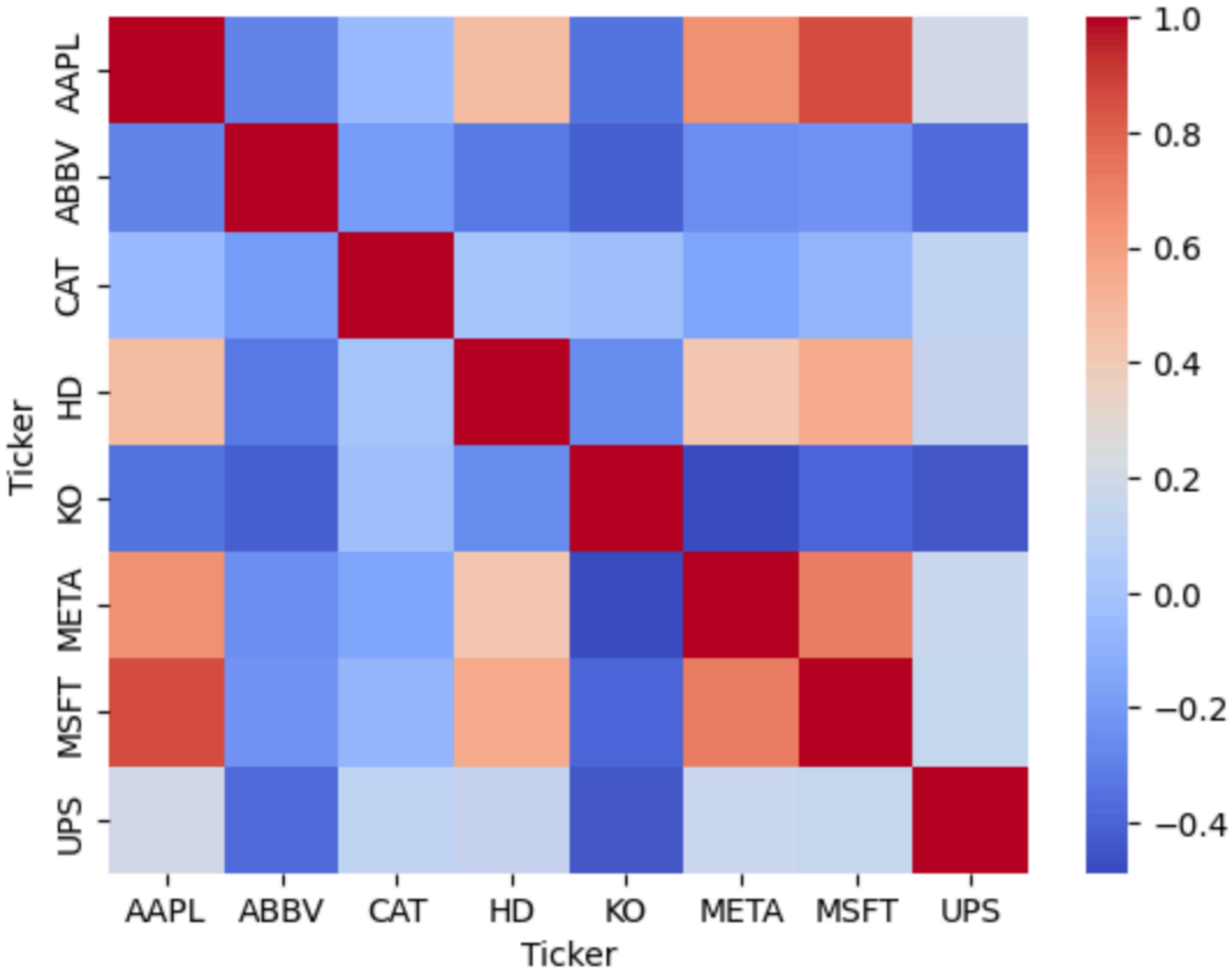
- Key components:
    - Market capitalizations
    - Covariance shrinkage
    - Market implied risk aversion
    - Market implied prior returns

- Key components:
    - Market capitalizations
    - Covariance shrinkage
    - Market implied risk aversion
    - Market implied prior returns
    - Omega

- Key components:
  - Omega
    - Uncertainty matrix, by specifying 1 s.d. confidence intervals for true return

- Key components:
    - Omega
        - Uncertainty matrix, by specifying 1 s.d. confidence intervals for true return
    - Efficient frontier optimizer
        - After finding BL returns and covariance, must optimize for max Sharpe

AAPL	ABBV	CAT	HD	KO	META	MSFT	UPS
40.524338	72.345779	133.842758	158.680954	36.539890	180.875397	79.633514	97.355186
40.517284	73.477913	134.047287	159.508026	36.459644	184.115646	80.004120	99.512161
40.705482	73.058861	135.888245	160.773895	36.973164	183.776672	80.708298	100.157684
41.168930	74.330681	138.035995	162.453278	36.965145	186.289108	81.708885	100.464714
41.016014	73.139740	141.504776	162.065048	36.908970	187.714813	81.792297	101.684891

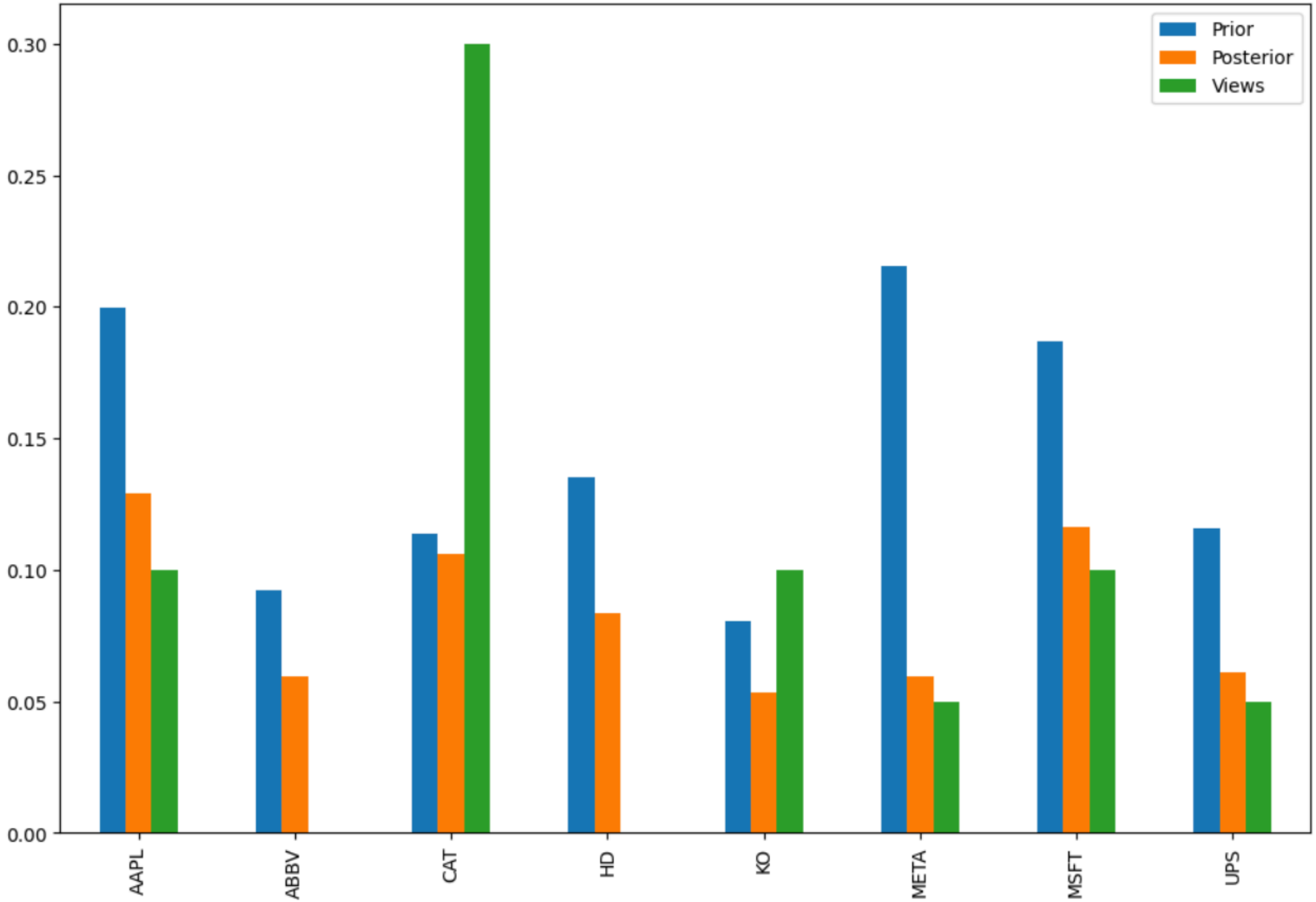

AAPL
ABBV
CAT
HD
KO
META
MSFT
UPS
Ticker
AAPL
ABBV
CAT
HD
KO
META
MSFT
UPS
Ticker
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

- Priors

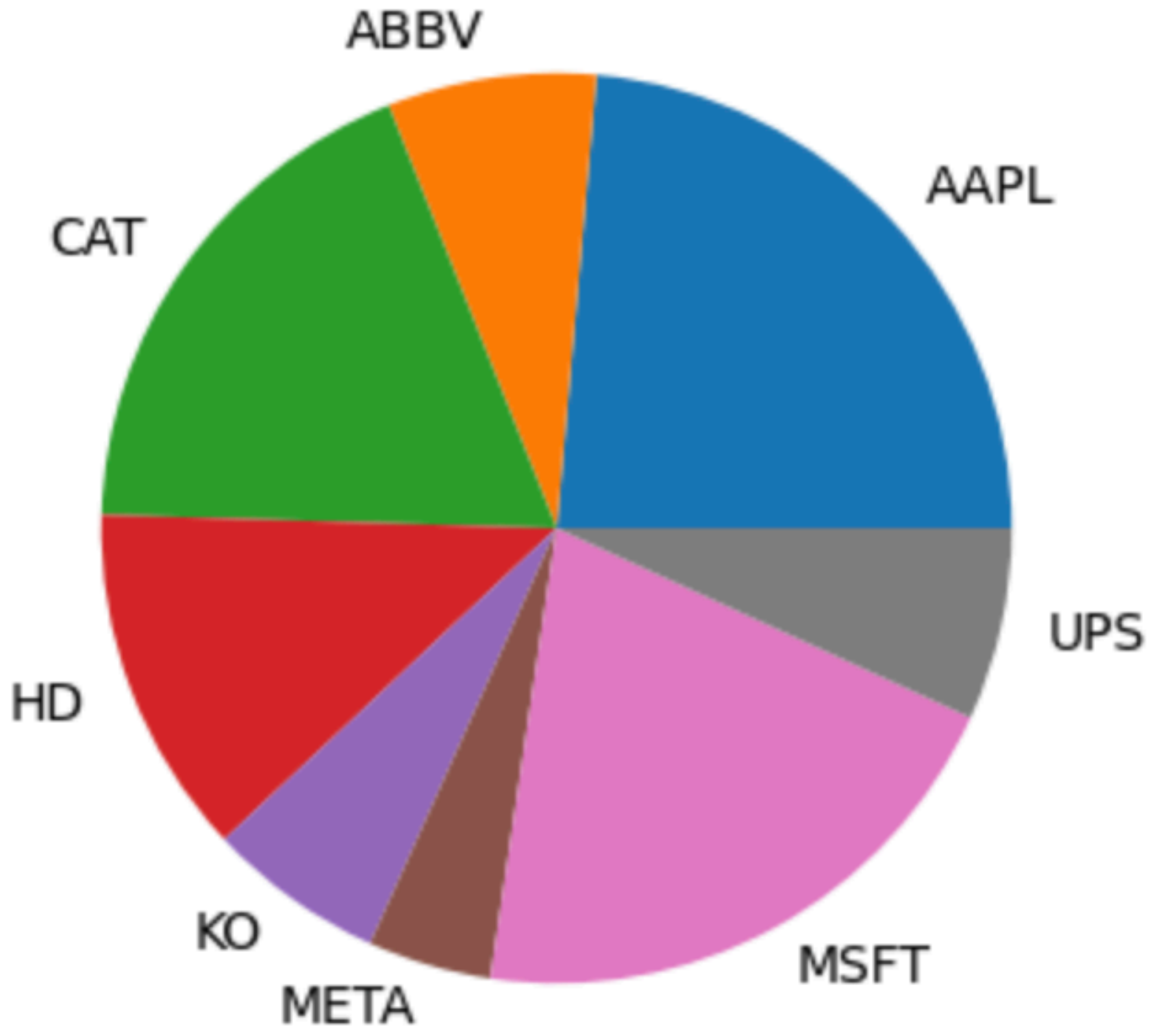
```
Ticker
AAPL     0.199520
ABBV     0.092113
CAT      0.113690
HD       0.134984
KO       0.080437
META     0.215492
MSFT     0.187026
UPS      0.115658
```

- Views

```
viewdict = {'AAPL': 0.10, 'MSFT': 0.10, 'META': 0.05,
            'KO': 0.1, 'UPS': 0.05, 'CAT': 0.3, 'HD': 0.1,}
```



Prior
Posterior
Views
0.00
0.05
0.10
0.15
0.20
0.25
0.30
AAPL
ABBV
CAT
HD
KO
META
MSFT
UPS




```
Expected annual return: 10.5%
Annual volatility: 23.8%
Sharpe Ratio: 0.36
```

```
OrderedDict([('AAPL', 0.22554),
             ('ABBV', 0.07669),
             ('CAT', 0.17773),
             ('HD', 0.15057),
             ('KO', 0.06745),
             ('META', 0.03699),
             ('MSFT', 0.19944),
             ('UPS', 0.06558)])
```