Homework 1

June 22, 2020

0.0.1 Homework 1: Mean Variance Optimization

FINM 25000

June 22, 2020

Julian LaNeve, Hao Zhu, Yimo Zheng, Kshitij Kashive

0.0.2 Setup

[51]:
```
import pandas as pd
import numpy as np
import pprint
pp = pprint.PrettyPrinter(indent=4)
```

First, import the data and display it to ensure it's loaded correctly.

[52]:
```
path_to_file = "C:/Users/lanev/OneDrive - Southern Methodist University/School/
↪UChicago/FINA 25500/Homework 1/assetclass_data_monthly_2009.xlsx"
raw_data = pd.read_excel(path_to_file)
raw_data.head(5)
```

[52]:
```
        Dates  Domestic Equity  Foreign Equity  Emerging Markets  \
0 2009-03-31         0.083484        0.083909          0.168616
1 2009-04-30         0.099347        0.115192          0.155561
2 2009-05-29         0.058453        0.131916          0.159411
3 2009-06-30        -0.000674       -0.014310         -0.022522
4 2009-07-31         0.074606        0.100413          0.110148

   Private Equity  Absolute Return  High Yield  Commodities  Real Estate  \
0        0.153477        -0.011116    0.019301     0.045049     0.035224
1        0.230201         0.022882    0.138431    -0.009187     0.296149
2        0.053890         0.027865    0.028495     0.196670     0.022728
3        0.045446        -0.003436    0.033359     0.005693    -0.025084
4        0.143248         0.015326    0.069164     0.004440     0.105799

   Domestic Bonds  Foreign Bonds  Inflation-Indexed      Cash
0        0.033104       0.047512           0.059060  0.001135
1       -0.027456       0.008993          -0.017967  0.000554
2       -0.020760       0.053668           0.020024 -0.000468
```

```
3       -0.005521       0.005150           0.002008  0.000598
4        0.008315       0.031284           0.000884 -0.000026
```

Calculate `excess_df`, a DataFrame of excess returns

[53]:
```python
df = raw_data.set_index("Dates")
risk_free = df.loc[:, "Cash"]
excess_rets = df.subtract(risk_free, axis = 0).drop("Cash", axis = 1)
excess_rets.head(5)
```

[53]:
```
            Domestic Equity  Foreign Equity  Emerging Markets  Private Equity  \
Dates
2009-03-31         0.082349        0.082774          0.167482        0.152342
2009-04-30         0.098793        0.114638          0.155007        0.229647
2009-05-29         0.058921        0.132384          0.159879        0.054357
2009-06-30        -0.001272       -0.014908         -0.023120        0.044847
2009-07-31         0.074632        0.100439          0.110174        0.143274

            Absolute Return  High Yield  Commodities  Real Estate  \
Dates
2009-03-31        -0.012250    0.018166     0.043914     0.034089
2009-04-30         0.022329    0.137877    -0.009741     0.295595
2009-05-29         0.028333    0.028963     0.197137     0.023195
2009-06-30        -0.004035    0.032761     0.005095    -0.025683
2009-07-31         0.015351    0.069189     0.004465     0.105825

            Domestic Bonds  Foreign Bonds  Inflation-Indexed
Dates
2009-03-31        0.031970       0.046377           0.057925
2009-04-30       -0.028010       0.008439          -0.018521
2009-05-29       -0.020293       0.054136           0.020491
2009-06-30       -0.006119       0.004552           0.001410
2009-07-31        0.008340       0.031310           0.000910
```

Define a formula to calculate a tangency portfolio

[72]:
```python
def tang_portfolio(rets, target_ret = None, scaling_factor = 1, mu_delta = 0,␣
↪diagonalize = False):
    mean_rets = np.mean(rets)
    mean_rets["Foreign Equity"] += mu_delta # only used in Q4

    vol_rets = np.std(rets, ddof=1)

    sigma = rets.cov()

    if diagonalize:
        sigma = np.diag(np.diag(sigma))
```

```
    sigma_inv = np.linalg.inv(sigma)
    weights = sigma_inv @ mean_rets
    weights = weights / weights.sum()
    w_tan = pd.Series(weights, index = mean_rets.index)

    if target_ret:
        scaling_factor = target_ret / (w_tan @ mean_rets)

    w_tan = w_tan * scaling_factor

    return w_tan.sort_values(ascending = False), {
        "monthly": {
            "mu": w_tan @ mean_rets,
            "vol": np.sqrt(w_tan @ sigma @ w_tan),
            "sharpe": (w_tan @ mean_rets) / np.sqrt(w_tan @ sigma @ w_tan)
        },
        "yearly": {
            "mu": w_tan @ mean_rets * 12,
            "vol": np.sqrt(w_tan @ sigma @ w_tan) * np.sqrt(12),
            "sharpe": (w_tan @ mean_rets) / np.sqrt(w_tan @ sigma @ w_tan) * np.
↪sqrt(12)
        }
    }
```

0.0.3 Q1: Summary Statistics

(a) Calculate and display the mean and volatility of each asset's excess return. (Recall we use volatility to refer to standard deviation.)

(b) Which assets have the best and worst Sharpe ratios?

[55]:
```
mean_rets = np.mean(excess_rets)
mean_rets.name = "Avg Monthly Excess Returns"

vol_rets = np.std(excess_rets, ddof=1)
vol_rets.name = "Std Dev of Monthly Excess Returns"

sharpe_ratios = mean_rets.divide(vol_rets)
sharpe_ratios.name = "Sharpe Ratio"

pd.concat([mean_rets, vol_rets, sharpe_ratios], axis = 1).sort_values("Sharpe␣
↪Ratio", ascending = False)
```

[55]:
```
                   Avg Monthly Excess Returns  \
Domestic Equity                      0.013028
High Yield                           0.007353
Real Estate                          0.014564
Private Equity                       0.013641
```

```
Inflation-Indexed                     0.002988
Domestic Bonds                        0.003093
Foreign Equity                        0.008126
Absolute Return                       0.001936
Emerging Markets                      0.008033
Foreign Bonds                         0.002109
Commodities                          -0.001676

                   Std Dev of Monthly Excess Returns  Sharpe Ratio
Domestic Equity                             0.037414      0.348220
High Yield                                  0.023927      0.307293
Real Estate                                 0.050511      0.288341
Private Equity                              0.057616      0.236753
Inflation-Indexed                           0.013942      0.214291
Domestic Bonds                              0.016780      0.184348
Foreign Equity                              0.045516      0.178535
Absolute Return                             0.012769      0.151593
Emerging Markets                            0.058230      0.137954
Foreign Bonds                               0.022195      0.095042
Commodities                                 0.055118     -0.030401
```

0.0.4 Q2: The MV Frontier

(a) Compute and display the weights of the tangency portfolios: w^{tan}.

[56]:
```python
weights, props = tang_portfolio(excess_rets)

print(weights)
```

```
Domestic Equity      1.100132
Domestic Bonds       0.799114
High Yield           0.791084
Inflation-Indexed    0.187910
Foreign Bonds       -0.022817
Foreign Equity      -0.045800
Commodities         -0.117513
Emerging Markets    -0.144565
Private Equity      -0.166304
Real Estate         -0.215180
Absolute Return     -1.166062
dtype: float64
```

(b) Compute the mean, volatility, and Sharpe ratio for the tangency portfolio corresponding to w^{tan}.

[57]:
```python
pp.pprint(props)
```

```
{   'monthly': {   'mu': 0.014138662446771718,
                   'sharpe': 0.6825681561145168,
```

```
                   'vol': 0.020713920390390474},
    'yearly': {   'mu': 0.1696639493612606,
                  'sharpe': 2.3644854520378966,
                  'vol': 0.0717551250801865}}
```

0.0.5 Q3: The Allocation

(a) Compute and display the weights of MV portfolios with target returns of $\mu^p = .0067$.

[58]:
```python
weights_adj, props_adj = tang_portfolio(excess_rets, target_ret = 0.0067)

print(weights_adj)
```

```
Domestic Equity      0.521328
Domestic Bonds       0.378682
High Yield           0.374877
Inflation-Indexed    0.089046
Foreign Bonds       -0.010812
Foreign Equity      -0.021704
Commodities         -0.055687
Emerging Markets    -0.068506
Private Equity      -0.078808
Real Estate         -0.101969
Absolute Return     -0.552571
dtype: float64
```

(b) What is the mean, volatility, and Sharpe ratio for w^p?

[60]:
```python
pp.pprint(props_adj)
```

```
{   'monthly': {   'mu': 0.006700000000000001,
                   'sharpe': 0.6825681561145174,
                   'vol': 0.009815869580174077},
    'yearly': {   'mu': 0.08040000000000001,
                  'sharpe': 2.3644854520378984,
                  'vol': 0.03400316966666257}}
```

(c) Discuss the allocation. In which assets is the portfolio most long? And short?

The allocation is very concentrated among Domestic Equity, Domestic Bonds, and High Yield with a large short in Absolute Return. The most long position is Domestic Equity.

(d) Does this line up with which assets have the strongest Sharpe ratios?

It partially lines up - Domestic Equity has the highest Sharpe ratio, and it's the highest concentration. The others aren't necessarily in the same order as the Sharpe ratio, and this is because weights are highly dependent on correlation, not returns

0.0.6 Q4: Long-Short Positions

(a) Consider an allocation between only domestic and foreign equities. (Drop all other return columns and recompute w^p for $\mu^p = .0067$.)

[66]:
```
weights_equities, props_equities = tang_portfolio(excess_rets[["Domestic␣
↪Equity", "Foreign Equity"]],
                                                    target_ret = 0.0067)
print(weights_equities, "\n")
pp.pprint(props_equities)
```

```
Domestic Equity     0.769695
Foreign Equity     -0.409531
dtype: float64

{   'monthly': {   'mu': 0.0066999999999999999,
                   'sharpe': 0.43093808955697505,
                   'vol': 0.015547476916899871},
    'yearly': {   'mu': 0.0804,
                  'sharpe': 1.4928133320586956,
                  'vol': 0.05385803989914 98}}
```

(b) What is causing the extreme long-short position?

The extreme long-short position is due to the high correlation between the Domestic Equity and Foreign Equity.

(c) Make an adjustment to $\mu^{foreign\ equities}$ of +0.001, (+0.012 annualized.) Recompute w^p for $\mu^p = .0067$ for these two assets. How does the allocation among the two assets change?

The allocation changes very insignificantly - the weights change very slightly to contain more Domestic Equity and a smaller short position in Foreign Equity.

[70]:
```
weights_equities_adj, props_equities_adj =␣
↪tang_portfolio(excess_rets[["Domestic Equity", "Foreign Equity"]],
                                                    target_ret = 0.0067, mu_delta␣
↪= 0.001)
print(weights_equities_adj, "\n")
pp.pprint(props_equities_adj)
```

```
Domestic Equity     0.780259
Foreign Equity     -0.379738
dtype: float64

{   'monthly': {   'mu': 0.006700000000000002,
                   'sharpe': 0.40620738944878665,
                   'vol': 0.016494037710864235},
    'yearly': {   'mu': 0.08040000000000003,
                  'sharpe': 1.4071436738704326,
                  'vol': 0.05713702267434782}}
```

(d) What does this say about the statistical precision of the MV solutions?

This shows that there is significant statistical precision - small changes in returns result in small changes in the weights.

0.0.7 Q5: Robustness

(a) Recalculate the full allocation, again with the unadjusted µ foreign equities and again for $\mu^p = 0.0067$. This time, make one change: in building w^{tan}, do not use Σ as given in the formulas in the lecture. Rather, use a diaganolized Σ^D, which zeroes out all non-diagonal elements of the full covariance matrix, Σ. How does the allocation look now?

[84]:
```
weights_no_cov, props_no_cov = tang_portfolio(excess_rets, target_ret = 0.0067,␣
 ↪diagonalize = True)
print(weights_no_cov)
pp.pprint(props_no_cov)
```

```
Inflation-Indexed    0.198569
High Yield           0.165913
Absolute Return      0.153376
Domestic Bonds       0.141925
Domestic Equity      0.120236
Real Estate          0.073746
Foreign Bonds        0.055320
Private Equity       0.053085
Foreign Equity       0.050674
Emerging Markets     0.030606
Commodities         -0.007125
dtype: float64
{   'monthly': {   'mu': 0.006700000000000001,
                   'sharpe': 0.7201577510976398,
                   'vol': 0.009303517166604248},
    'yearly': {   'mu': 0.08040000000000001,
                  'sharpe': 2.494699628731307,
                  'vol': 0.0322283288432956}}
```

(b) What does this suggest about the sensitivity of the solution to estimated means and estimated covariances?

It's extremely sensitive to the estimated covariances; the new allocation is vastly different and no where near as extremely long/short.

(c) HMC deals with this sensitivity by using explicit constraints on the allocation vector. Conceptually, what are the pros/cons of doing that versus modifying the formula with Σ^D?

Using the diagonal matrix to calculate the solution may be easier to calculate its inverse if we do it by hand. However, since we assume the covariances between any two assets are zero in the diagonal matrix, this gives us an imprecise solution. From the HMC case, we can see that covariances between assets can affect the sharpe ratio and optimality of the optimized solution significantly.

0.0.8 Q6: Out-of-Sample Performance

(a) Using only data through the end of 2016, compute w^p for $\mu^p = .0067$, allocating to all 11 assets.

[76]:
```
weights_thru_2016, props_thru_2016 = tang_portfolio(excess_rets[excess_rets.
↪index <= "2016-12-31"], target_ret = 0.0067)
print(weights_thru_2016)
```

```
Domestic Equity      0.469822
High Yield           0.318027
Domestic Bonds       0.297214
Inflation-Indexed    0.147943
Commodities         -0.037083
Foreign Equity      -0.041204
Private Equity      -0.062163
Foreign Bonds       -0.062232
Emerging Markets    -0.078779
Real Estate         -0.084349
Absolute Return     -0.340700
dtype: float64
```

(b) Calculate the portfolio's Sharpe ratio within that sample, through the end of 2016.

[78]:
```
pp.pprint(props_thru_2016)
```

```
{   'monthly': {   'mu': 0.0067,
                   'sharpe': 0.7520647641821946,
                   'vol': 0.008908807218598614},
    'yearly': {   'mu': 0.0804,
                  'sharpe': 2.6052287642917347,
                  'vol': 0.030861013474898346}}
```

(c) Calculate the portfolio's Sharpe ratio based on performance in 2017-2019.

[86]:
```
portfolio = excess_rets[excess_rets.index > "2016-12-31"] @ weights_thru_2016

pp.pprint({
    "monthly": { "sharpe": portfolio.mean() / portfolio.std() },
    "yearly": { "sharpe": portfolio.mean() / portfolio.std() * np.sqrt(12) }
})
```

```
{   'monthly': {'sharpe': 0.45506836842567955},
    'yearly': {'sharpe': 1.5764030700614993}}
```

(d) How does this out-of-sample Sharpe compare to the 2009-2016 performance of a portfolio optimized to μ^p using 2009-2016 data?

The Sharpe ratio is significantly higher in-sample than out-of-sample, meaning the portfolio performs significantly better in-sample compared to out-of-sample.

0.0.9 Q7: Robust Out-of-Sample Performance

Recalculate w^p on 2009-2016 data using the diaganolized covariance matrix, Σ^D. What is the performance of this portfolio in 2017-2019? Does it do better out of of sample than the portfolio constructed on 2009-2016 data using the full covariance matrix?

It still performs significantly worse out-of-sample compared to in-sample. It does not do better out-of-sample compared to the full covariance matrix.

[85]:
```
weights_thru_2016_no_cov, props_thru_2016_no_cov =␣
↪tang_portfolio(excess_rets[excess_rets.index <= "2016-12-31"],
                                                              target_ret =␣
↪0.0067, diagonalize = True)
pp.pprint(props_thru_2016_no_cov)

portfolio = excess_rets[excess_rets.index > "2016-12-31"] @␣
↪weights_thru_2016_no_cov

pp.pprint({
    "monthly": { "sharpe": portfolio.mean() / portfolio.std() },
    "yearly": { "sharpe": portfolio.mean() / portfolio.std() * np.sqrt(12) }
})
```

```
{   'monthly': {   'mu': 0.0067,
                   'sharpe': 0.763444439872619,
                   'vol': 0.008776015188633633},
    'yearly': {   'mu': 0.0804,
                  'sharpe': 2.644649117230678,
                  'vol': 0.030401008389419233}}
{   'monthly': {'sharpe': 0.2829572228020353},
    'yearly': {'sharpe': 0.9801925725234238}}
```