FINM 36700: Portfolio Management

TA Review Session 1

Autumn 2023

Agenda

1. Getting Started with Jupyter Notebook for homework 1
2. Mathematics and Implementation of MV portfolio for excess returns
3. Homework 1 Highlights

Getting Started with Jupyter: Importing Libraries

- Start by importing the necessary libraries and formatting your floating-point numbers to a 4 decimal precision, unless specified
- Some basic libraries used in almost all assignments are:

```
import pandas as pd
import numpy as np
pd.options.display.float_format = "{:,.4f}".format

import matplotlib.pyplot as plt
import seaborn as sns

import statsmodels.api as sm
from sklearn.linear_model import LinearRegression

import warnings
warnings.filterwarnings("ignore")
```

Getting Started with Jupyter: Loading Data

- Data can be loaded using the pd.read_excel() function by specifying the file path and the sheet name
- For example, in homework 1, we are working with excess returns:

```
multi_asset_etf_descriptions = pd.read_excel('multi_asset_etf_data.xlsx')
```

```
multi_asset_etf_excess_ret = pd.read_excel('multi_asset_etf_data.xlsx', sheet_name = 'excess returns')
multi_asset_etf_excess_ret.head(2)
```

	Date	BWX	DBC	EEM	EFA	HYG	IEF	IYR	PSP	QAI	SPY	TIP
0	2009-04-30	0.0084	-0.0016	0.1550	0.1146	0.1379	-0.0280	0.2956	0.2296	0.0223	0.0988	-0.0185
1	2009-05-31	0.0541	0.1631	0.1599	0.1324	0.0290	-0.0203	0.0232	0.0544	0.0283	0.0589	0.0204

Getting Started with Jupyter: Helper Functions

- It's extremely helpful to functionalize and parameterize your code to as they can be recursively used on subsequent assignments and exams

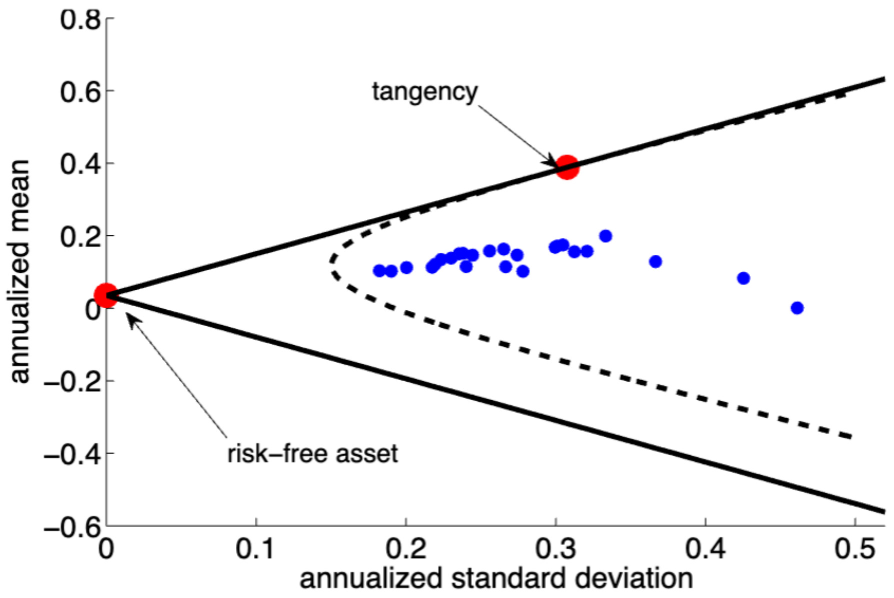
Helper Functions

```python
def summary_statistics_annualized(returns, annual_factor = 12):
    """This functions returns the summary statistics for the input total/excess returns passed
    into the function"""

    summary_statistics = pd.DataFrame(index=returns.columns)
    summary_statistics['Mean'] = returns.mean() * annual_factor
    summary_statistics['Vol'] = returns.std() * np.sqrt(annual_factor)
    summary_statistics['Sharpe'] = (returns.mean() / returns.std()) * np.sqrt(annual_factor)
    summary_statistics['Min'] = returns.min()
    summary_statistics['Max'] = returns.max()
    summary_statistics['Skewness'] = returns.skew()
    summary_statistics['Excess Kurtosis'] = returns.kurtosis()
    summary_statistics['VaR (0.05)'] = returns.quantile(.05, axis = 0)
    summary_statistics['CVaR (0.05)'] = returns[returns <= returns.quantile(.05, axis = 0)].mean()

    return summary_statistics

```

Total Returns vs Excess Returns

- Excess Returns : With a risk-free rate, the optimal $\widetilde{MV}$ portfolio is a combination of the tangency portfolio $\mathbf{w}^t$ and a position in the riskless asset (the two bold straight lines)

- Total Returns: In absence of the risk-free rate, the optimal MV portfolio is a combination of the global minimum variance portfolio and the tangency portfolio (the curved frontier from GMV to tangency)



Math behind Excess Returns: Notations

- n : Number of risky assets available
- **r** : returns of the risky assets
- **w** : n x 1 vector of portfolio allocations to n risky assets
- 1 – **w' 1** = Allocation to the risk-free rate (The assumption here is that any investment not made in the risky assets is invested in the risk-free rate)
- $\boldsymbol{\mu}$: Vector of mean returns of risky assets
- μ^p: Mean return on a portfolio
- $\tilde{\boldsymbol{\mu}}$: Excess Returns

Math behind Excess Returns: Mean Excess Returns and Variance of Returns

- Mean Return (μ^p)and Mean Excess Return ($\widetilde{\mu^p}$)
- Variance (σ_p^2) remains the same as before

$$\mu^p = (1 - \mathbf{w}'\mathbf{1})r^f + \mathbf{w}'\boldsymbol{\mu}$$

$$\mu^p = r^f - \mathbf{w}'\mathbf{1}r^f + \mathbf{w}'\boldsymbol{\mu}$$

$$\mu^p = r^f + \underbrace{\mathbf{w}'(\boldsymbol{\mu} - \mathbf{1}r^f)}_{\tilde{\boldsymbol{\mu}}}$$

$$\mu^p = r^f + \mathbf{w}'\tilde{\boldsymbol{\mu}}$$

$$\mu^p - r^f = \tilde{\mu^p} = \mathbf{w}'\tilde{\boldsymbol{\mu}}$$

$$\sigma_p^2 = \mathbf{w}'\boldsymbol{\Sigma}\mathbf{w}$$

Math behind Excess Returns: $\widetilde{MV}$ problem with a riskless asset

- A Mean-Variance portfolio with a risk-free asset is a vector $\boldsymbol{w}^*$ that solves the following optimization for some mean excess return value $\tilde{\mu}^p$

$$\min_{\boldsymbol{w}} \quad \boldsymbol{w}'\boldsymbol{\Sigma}\boldsymbol{w}$$
$$\text{s.t.} \quad \boldsymbol{w}'\tilde{\boldsymbol{\mu}} = \tilde{\mu}^p$$

Math behind Excess Returns: $\widetilde{MV}$ solution

$$\boldsymbol{w}^* = \tilde{\delta}\, \boldsymbol{w}^{t}$$

for the portfolio

$$\boldsymbol{w}^{t} = \underbrace{\left(\frac{1}{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\tilde{\mu}}\right)}_{\text{scaling}} \boldsymbol{\Sigma}^{-1}\tilde{\mu}$$

and allocation

$$\tilde{\delta} = \left(\frac{\mathbf{1}'\boldsymbol{\Sigma}^{-1}\tilde{\mu}}{(\tilde{\mu})'\boldsymbol{\Sigma}^{-1}\tilde{\mu}}\right)\tilde{\mu}^{p}$$

- The result is a combination of the tangency portfolio and a position in the riskless asset

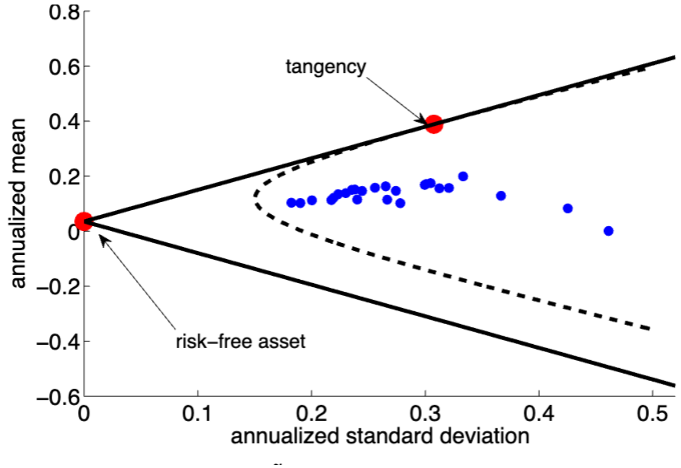
Math behind Excess Returns: *Sharpe Ratio*

- Sharpe Ratio is a measure of risk-adjusted excess mean return of a portfolio. Investors seek to maximize the SR.

- The tangency portfolio is the portfolio on the risky MV frontier with maximum Sharpe ratio.

$$\mathrm{SR}(\boldsymbol{w}^*) = \pm\sqrt{(\tilde{\boldsymbol{\mu}})' \boldsymbol{\Sigma}^{-1} \tilde{\boldsymbol{\mu}}}$$

- It is the slope of the line



Homework 1: Key Questions Section 2

- Summary Statistics:

```python
def summary_statistics_annualized(returns, annual_factor = 12):
    """This functions returns the summary statistics for the input total/excess returns passed
    into the function"""

    summary_statistics = pd.DataFrame(index=returns.columns)
    summary_statistics['Mean'] = returns.mean() * annual_factor
    summary_statistics['Vol'] = returns.std() * np.sqrt(annual_factor)
    summary_statistics['Sharpe'] = (returns.mean() / returns.std()) * np.sqrt(annual_factor)
    summary_statistics['Min'] = returns.min()
    summary_statistics['Max'] = returns.max()
    summary_statistics['Skewness'] = returns.skew()
    summary_statistics['Excess Kurtosis'] = returns.kurtosis()
    summary_statistics['VaR (0.05)'] = returns.quantile(.05, axis = 0)
    summary_statistics['CVaR (0.05)'] = returns[returns <= returns.quantile(.05, axis = 0)].mean()

    return summary_statistics

```

Homework 1: Key Questions Section 2

- Question during OH: Why do we need to annualize Sharpe ratio again if mean and vol are annualized?
- If you are dividing the monthly mean returns by monthly vol, you need to scale it by $\sqrt{n}$ to annualize it. However, if you are using annualized metrics you don't need to scale it again

Homework 1: Key Questions Section 2

```
annual_factor = 12
excess_returns = multi_asset_etf_excess_ret[multi_asset_etf_excess_ret.columns[1:]]
annualized_mean = excess_returns.mean()* annual_factor
annualized_vol = excess_returns.std()* np.sqrt(annual_factor)
sharpe_ratio = annualized_mean/ annualized_vol
print(sharpe_ratio)
```

```
BWX   -0.0221
DBC    0.1422
EEM    0.3302
EFA    0.4916
HYG    0.7197
IEF    0.2287
IYR    0.6920
PSP    0.3516
QAI    0.3734
SPY    0.9732
TIP    0.4332
dtype: float64
```

]:

```
excess_returns = multi_asset_etf_excess_ret[multi_asset_etf_excess_ret.columns[1:]]
sharpe_ratio = (excess_returns.mean() / excess_returns.std())*np.sqrt(annual_factor)
print(sharpe_ratio)
```

```
BWX   -0.0221
DBC    0.1422
EEM    0.3302
EFA    0.4916
HYG    0.7197
IEF    0.2287
IYR    0.6920
PSP    0.3516
QAI    0.3734
SPY    0.9732
TIP    0.4332
dtype: float64
```

Homework 1: Key Questions Section 2

- Tangency Weights:

```
def tangency_weights(returns, cov_mat = 1):

    if cov_mat ==1:
        cov_inv = np.linalg.inv((returns.cov()*12))
    else:
        cov = returns.cov()
        covmat_diag = np.diag(np.diag((cov)))
        covmat = cov_mat * cov + (1-cov_mat) * covmat_diag
        cov_inv = np.linalg.inv((covmat*12))

    ones = np.ones(returns.columns[1:].shape)
    mu = returns.mean()*12
    scaling = 1/(np.transpose(ones) @ cov_inv @ mu)
    tangent_return = scaling*(cov_inv @ mu)
    tangency_wts = pd.DataFrame(index = returns.columns[1:], data = tangent_return, columns = ['Tangent Weights'] )

    return tangency_wts
```

- To determine the performance of the tangent portfolio, multiply the returned tangency weights with the excess returns and pass it to the summary function – this way you can avoid writing multiple summary statistic function

- Revisiting Sharpe Ratio

Homework 1: Key Questions Section 2

- **4. TIPS**
- Assess how much the tangency portfolio (and performance) change if...
- TIPS are dropped completely from the investment set.
- The expected excess return to TIPS is adjusted to be 0.0012 higher than what the historic sample shows.
- Based on the analysis, do TIPS seem to expand the investment opportunity set, implying that Harvard should consider them as a separate asset?