RICE UNIVERSITY

HOMEWORK 2: MPT AND CAPM

STAT 682 - Fall 2024

Authors:

Dolese, Ryker

Fang, Judy

Hao, Sirui

Hecht, Martin

Thetford, Jackson

Professors:

Dr. Michael D Jackson

Max Lee

Data Overview

Data Sources

- **Fama-French 5-Factor Data**: This data was retrieved from the Fama-French Data Library. This dataset includes five risk factors: market excess return (Mkt-RF), size factor (SMB), value factor (HML), profitability factor (RMW), and investment factor (CMA). This dataset was primarily used in the analysis for **Question 6**.
- **Fama-French Daily Data**: A CSV file titled *F-F_Research_Data_Factors_daily.CSV* was used to retrieve daily data for the three-factor model: Mkt-RF, SMB, HML, and the risk-free rate (RF). This data was mainly used for other questions.
- **Stock Prices Dataset**: An Excel file titled *Homework 2 data.xlsx* was used, containing daily stock price data for various stocks such as AAPL, MSFT, NVDA, and others.

Data Frequency

- **Fama-French 5-Factor Data**: Available on a daily frequency, providing daily returns for the U.S. market.
- **Fama-French Daily Data**: Daily data for the three-factor model, used to explain stock returns.
- **Stock Prices**: Daily stock price data used to calculate daily returns for selected stocks.

Data Coverage

- **Fama-French 5-Factor Data**: Covers several decades of daily data for the U.S. equity market, enabling analysis of factor returns over time. This was used for a deep dive in **Question 5**.
- **Fama-French Daily Data**: Includes daily market, size, and value factors for analyzing U.S. stock returns. The timeframe overlaps with stock price data to ensure consistency in the analysis.
- **Stock Price Data**: Covers daily stock prices for major U.S. stocks from various sectors, including technology, financial, and retail. Used to calculate returns and perform factor model regressions.

Data Treatment and Cleaning

To ensure the data used in the analysis is accurate and free of missing values, the following steps were applied to clean and prepare the data:

- The Fama-French daily data (three-factor model) was loaded and cleaned. The original dataset contains various columns, but only the necessary factors (Mkt-RF, SMB, HML, and RF) were retained.
- Rows with missing data were dropped using the ‘dropna()‘ function to ensure the accuracy of the model.
- The dates in the dataset, which were initially in ‘YYYYMMDD‘ format, were converted to a ‘DateTime‘ object for easier merging with stock price data.
- The ‘Unnamed: 0‘ column (which contained dates) was renamed and set as the index to facilitate merging with other datasets.
- Daily stock price data was also loaded from the provided Excel file. This data was used to calculate daily returns, which were later merged with the Fama-French factors for the regression analysis.

In the analysis, we cleaned the data as described in Appendix A.1.

Question 1

Select 10 stocks from the top 20 companies in the S&P 500 by market cap, and compute the traditional CAPM and Fama-French (FF) 3-Factor CAPM coefficients.

1. Use percent returns, not log returns.

2. Describe the relevance of Beta for each company from each model.

Answer:

Table of the coefficients

We run the Python code to calculate the percent returns and apply the CAPM and Fama-French 3-Factor models. See Appendix A.2 for the full details of the code used to calculate the coefficients.

After running the code, we obtain the following coefficients for each of the selected stocks. The tables below summarize the Alpha and Beta values for each stock from both models. CAPM Beta measures the sensitivity of the stock to the overall market (Mkt-RF), while the Fama-French 3-Factor model further breaks down the Beta coefficients by including the size factor (SMB) and the value factor (HML).

Table 1: CAPM Coefficients for Selected Stocks

Stock	**CAPM Alpha**	**CAPM Beta (Mkt-RF)**
AAPL	0.050	1.233
MSFT	0.055	1.190
NVDA	0.144	1.661
GOOG	0.023	1.114
MA	0.073	1.127
HD	0.054	1.096
PG	0.011	0.628
COST	0.036	0.823
JPM	-0.005	1.343
V	0.046	0.973

Table 2: Fama-French 3-Factor Coefficients for Selected Stocks

Stock	FF3 Alpha	FF3 Beta (Mkt-RF)	FF3 Beta (SMB)	FF3 Beta (HML)
AAPL	0.059	1.193	-0.049	-0.650
MSFT	0.060	1.156	-0.301	-0.612
NVDA	0.147	1.632	0.507	-0.863
GOOG	0.017	1.125	-0.261	-0.415
MA	0.072	1.149	-0.210	-0.015
HD	0.055	1.090	-0.083	-0.062
PG	0.012	0.608	-0.602	-0.030
COST	0.038	0.827	-0.234	-0.246
JPM	-0.020	1.414	-0.243	1.049
V	0.045	0.989	-0.206	0.022

Relevance of Beta for Each Company

Now based on the results we got, we can further analyze the relevance of Beta for each company from each model.

- **Apple Inc. (AAPL)**
  - **CAPM Model:** The CAPM Beta (Mkt-RF) for AAPL is 1.233, indicating that Apple's stock is more volatile than the market. A Beta above 1 suggests that Apple's returns move more than the market, implying higher risk and potential for higher returns.
  - **Fama-French Model:**
    - The FF3 Beta (Mkt-RF) for AAPL is 1.193, slightly lower than the CAPM Beta, reflecting a similar sensitivity to market risk.
    - The SMB Beta is -0.049, showing that Apple behaves more like a large-cap stock.
    - The HML Beta is -0.650, indicating that Apple is more of a growth stock rather than a value stock.
- **Microsoft Corp. (MSFT)**
  - **CAPM Model:** The CAPM Beta for MSFT is 1.190, implying Microsoft's stock has a slightly higher volatility compared to the market, exposing it to market movements.
  - **Fama-French Model:**

    * The FF3 Beta (Mkt-RF) for MSFT is 1.156, close to the CAPM Beta, reflecting similar market risk.
    * The SMB Beta of -0.301 suggests that Microsoft is aligned with large-cap stocks.
    * The HML Beta of -0.612 implies that Microsoft is more growth-oriented than value-oriented.

- **NVIDIA Corp. (NVDA)**
  - **CAPM Model:** NVIDIA's CAPM Beta is 1.661, indicating significant sensitivity to market fluctuations, typical for the high-growth tech sector.
  - **Fama-French Model:**
    * The FF3 Beta (Mkt-RF) is 1.632, showing strong market exposure.
    * The SMB Beta of 0.507 reflects small-cap characteristics.
    * The HML Beta of -0.863 shows NVIDIA is growth-oriented rather than value-focused.
- **Alphabet Inc. (GOOG)**
  - **CAPM Model:** Alphabet's CAPM Beta is 1.114, showing slight volatility above the market.
  - **Fama-French Model:**
    * The FF3 Beta (Mkt-RF) is 1.125, similar to the CAPM Beta.
    * The SMB Beta is -0.261, indicating Alphabet behaves as a large-cap stock.
    * The HML Beta of -0.415 suggests Alphabet is more growth-oriented.
- **Mastercard Inc. (MA)**
  - **CAPM Model:** Mastercard's CAPM Beta is 1.127, indicating higher volatility than the market.
  - **Fama-French Model:**
    * The FF3 Beta (Mkt-RF) is 1.149, showing similar market sensitivity.
    * The SMB Beta of -0.210 reflects its large-cap characteristics.
    * The HML Beta of -0.015 indicates Mastercard is growth-oriented, but not strongly.
- **The Home Depot Inc. (HD)**
  - **CAPM Model:** Home Depot's CAPM Beta is 1.096, showing slightly higher market volatility.
  - **Fama-French Model:**
    * The FF3 Beta (Mkt-RF) is 1.090, confirming its market risk.
    * The SMB Beta of -0.083 shows large-cap behavior.

    * The HML Beta of -0.062 indicates mild growth orientation.

- **Procter & Gamble Co. (PG)**
  - **CAPM Model:** Procter & Gamble's CAPM Beta is 0.628, suggesting it is less volatile than the market, making it a more defensive stock.
  - **Fama-French Model:**
    * The FF3 Beta (Mkt-RF) is 0.608, confirming its lower market sensitivity.
    * The SMB Beta of -0.602 aligns P&G with large-cap stocks.
    * The HML Beta of -0.030 shows minimal value or growth tendencies.
- **Costco Wholesale Corp. (COST)**
  - **CAPM Model:** Costco's CAPM Beta is 0.823, indicating lower volatility compared to the market.
  - **Fama-French Model:**
    * The FF3 Beta (Mkt-RF) is 0.827, consistent with the CAPM result.
    * The SMB Beta of -0.234 confirms its large-cap nature.
    * The HML Beta of -0.246 suggests growth tendencies.
- **JPMorgan Chase & Co. (JPM)**
  - **CAPM Model:** JPMorgan's CAPM Beta is 1.343, indicating higher volatility typical for financial institutions.
  - **Fama-French Model:**
    * The FF3 Beta (Mkt-RF) is 1.414, reinforcing its high market exposure.
    * The SMB Beta of -0.243 shows large-cap characteristics.
    * The HML Beta of 1.049 reflects value-stock tendencies.
- **Visa Inc. (V)**
  - **CAPM Model:** Visa's CAPM Beta is 0.973, indicating market-like volatility.
  - **Fama-French Model:**
    * The FF3 Beta (Mkt-RF) is 0.989, similar to the CAPM result.
    * The SMB Beta of -0.206 shows its large-cap characteristics.
    * The HML Beta of 0.022 suggests neutrality between growth and value.

Question 2

Devise some non-cluttered way to report the coefficients and the significance, perhaps by quoting the t-statistics.

Answer:

We calculated the CAPM and Fama-French 3-Factor model coefficients along with their significance using t-statistics. The table below reports the coefficients with corresponding t-statistics for each model. Coefficients with significant t-statistics (above 1.96 or below -1.96) are marked with an asterisk (*), indicating they are significant at the 5% level.

For the details of the code used to calculate the coefficients and t-statistics, please refer to Appendix A.3.

CAPM Model Results

Table 3: CAPM Coefficients and t-Statistics for Alpha and Market Risk Factor (Mkt-RF)

Stock	**FF3 Alpha**	**FF3 Beta (Mkt-RF)**	**t-Statistics (Mkt-RF)**
AAPL	0.059	1.193	57.499*
MSFT	0.060	1.156	83.200*
NVDA	0.147	1.632	50.501*
GOOG	0.017	1.125	54.669*
MA	0.072	1.149	61.357*
HD	0.055	1.090	71.121*
PG	0.012	0.608	59.424*
COST	0.038	0.827	54.358*
JPM	-0.020	1.414	108.322*
V	0.045	0.989	56.828*

Fama-French 3-Factor Model Results

Table 4: Fama-French 3-Factor Coefficients and t-Statistics for Alpha and Market Risk Factor (Mkt-RF)

Stock	FF3 Alpha	FF3 Beta (Mkt-RF)	t-Statistics (Mkt-RF)
AAPL	0.059	1.193	58.865*
MSFT	0.060	1.156	56.379*
NVDA	0.147	1.632	72.543*
GOOG	0.017	1.125	53.216*
MA	0.072	1.149	55.439*
HD	0.055	1.090	51.875*
PG	0.012	0.608	32.325*
COST	0.038	0.827	45.679*
JPM	-0.020	1.414	60.124*
V	0.045	0.989	50.413*

Table 5: Fama-French 3-Factor Coefficients and t-Statistics for SMB and HML Factors

Stock	FF3 Beta (SMB)	t-Statistics (SMB)	FF3 Beta (HML)	t-Statistics (HML)
AAPL	-0.049	-1.291	-0.650	-18.535*
MSFT	-0.301	-11.759*	-0.612	-26.049*
NVDA	0.507	8.133*	-0.863	-16.925*
GOOG	-0.261	-7.314*	-0.415	-15.926*
MA	-0.210	-5.744*	-0.015	-0.545
HD	-0.083	-2.943*	-0.062	-2.397*
PG	-0.602	-32.085*	-0.030	-1.709
COST	-0.234	-8.110*	-0.246	-10.025*
JPM	-0.243	-10.175*	1.049	47.549*
V	-0.206	-6.101*	0.022	0.873

Key Insights:

- **Market Risk Factor (Mkt-RF):** The t-statistics for Beta (Mkt-RF) are high across most stocks, as the market explains much of the return variance, especially for large-caps like MSFT, GOOG, and JPM, where the market risk factor plays a dominant role in their performance.

- **SMB (Small Minus Big):** The size factor shows varying significance, with negative coefficients for large-cap stocks such as MSFT and PG, indicating their large-cap characteristics, while NVDA has a positive SMB value, suggesting some small-cap behavior.
- **HML (High Minus Low):** The value/growth factor (HML) has strong negative t-statistics for growth stocks like AAPL and NVDA, confirming their growth stock tendencies.

Question 3

For the stock of your choosing from the top 20 companies in the S&P 500 by market cap, obtain and plot a year rolling beta from 2000-present (i.e., on each day regress the previous 252 trade days to calculate your betas over time).

Answer:

We run the code to get the results. For code details, refer to Appendix A.4.

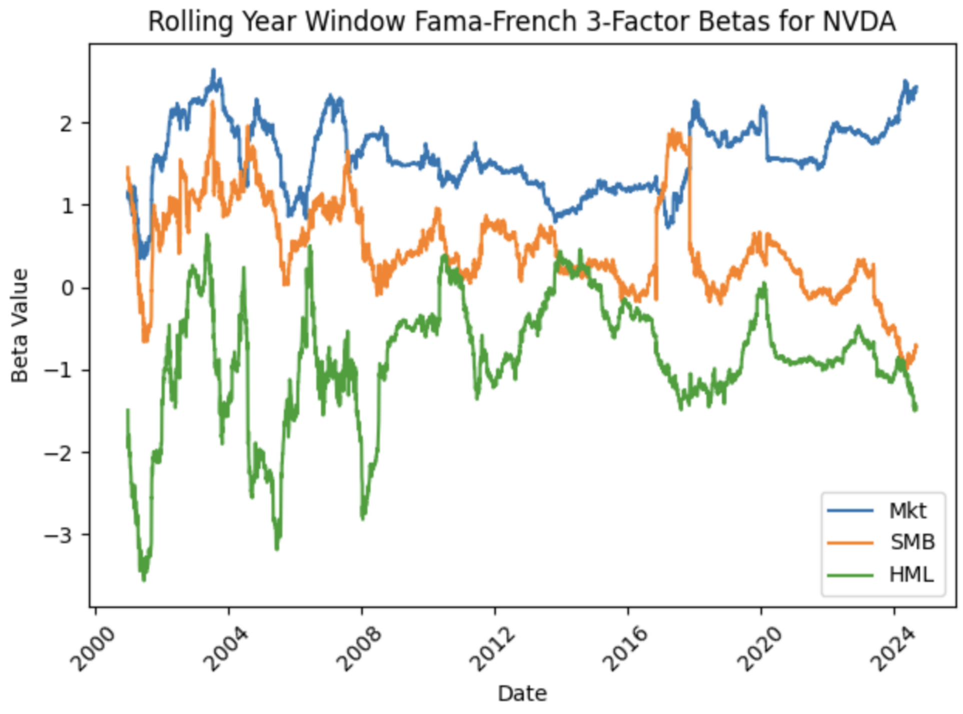


Figure 1: NVDA Beta Values (2000-Present)

Nvidia's β_{Mkt} value tends to hover between 1 and 2, suggesting it is a more agressive stock relative to the market. It's sensitivity to the size factor β_{SMB}, typically falls between 0 and 1, suggesting there is a smaller, positive relationship between the difference in performance of small and large cap firms. Meanwhile, the β_{HML} is typically negative, suggesting the size factor and Nvidia returns typically move in opposite directions. However, it is important to note that β_{HML} was previously much more volatile from 2000-2010 but has since become more steady, typically staying around -1.

Question 4

Summarize how the Fama-French factors are derived and maintained.

Answer:

The Fama-French 3 Factor model can be written as follows:

$$R_i - R_f = \alpha_i + \beta_{\text{Mkt}}(R_{\text{Mkt}} - R_f) + \beta_{\text{SMB}} \cdot \text{SMB} + \beta_{\text{HML}} \cdot \text{HML} + \epsilon_i \tag{1}$$

Where:

- R_i is the return of stock i,
- R_f is the risk-free rate
- R_M is the market return
- - R_M is the market return
- α_i is the intercept, or stock-specific alpha
- β_{iM} is the sensitivity to the market risk premium
- β_{iSMB} is the sensitivity to the size factor (SMB)
- β_{iHML} is the sensitivity to the value factor (HML)
- ϵ_i is the error term

The factors:

- Market Risk Premium: $(R_M - R_f)$ represents the excess return of the market over the risk-free rate.
- SMB (Small Minus Big): Captures the size effect (small-cap stocks tend to outperform large-cap stocks).
- HML (High Minus Low): Captures the value effect (value stocks tend to outperform growth stocks).

Derivations and Maintenance

1. Market Risk Premium (Mkt-RF)

   - Derivation: This is the excess return of the overall stock market over the risk-free rate.
   - Market Return is usually represented by a broad market index like the S&P 500 or CRSP total market index, which includes all listed U.S. stocks.
   - Risk-Free Rate typically comes from short-term U.S. Treasury bills (often the 1-month or 3-month T-bill rate). The market risk premium measures the additional return an investor earns by investing in the stock market over a risk-free asset.

2. Small Minus Big (SMB) - Size Factor

   - Derivation: SMB captures the size effect, where historically, small-cap stocks (small companies) tend to outperform large-cap stocks (large companies). To derive SMB, stocks are sorted by market capitalization and divided into small and large categories.
   - Steps for calculation:
     - Rank all stocks in the market based on their market capitalization at the start of the period.
     - Divide stocks into small (bottom 50% by market cap) and large (top 50% by market cap).
     - SMB is the difference in the average returns of the small-cap stock portfolio minus the large-cap stock portfolio:

$$SMB = \frac{1}{3}(\text{Small-Value} + \text{Small-Neutral} + \text{Small-Growth}) - \frac{1}{3}(\text{Big-Value} + \text{Big-Neutral} + \text{Big-Growth})$$

   - Maintenance: SMB is updated periodically (often monthly) to reflect changes in the size classifications of stocks, based on market capitalizations. Stocks can shift from large to small and vice versa as their market capitalizations change over time.

3. High Minus Low (HML) - Value Factor

   - Derivation: HML captures the value effect, where value stocks (companies with high book-to-market ratios) tend to outperform growth stocks (companies with low book-to-market ratios). To derive HML, stocks are sorted based on their book-to-market (B/M) ratio, which is:

$$\text{Book-to-Market} = \frac{\text{Market Value of Equity}}{\text{Book Value of Equity}}$$

   - Steps for calculation:

  - Rank stocks based on their book-to-market ratio.
  - Divide stocks into value (top 30% of book-to-market ratios) and growth (bottom 30% of book-to-market ratios).
  - HML is the difference between the returns of high book-to-market stocks (value) and low book-to-market stocks (growth):

$$HML = \frac{1}{2}\,(\text{Small-Value} + \text{Big-Value}) - \frac{1}{2}\,(\text{Small-Growth} + \text{Big-Growth})$$

- Maintenance:
  - HML is updated monthly as companies' book values and market values change.
  - Stocks move between value and growth categories based on shifts in their book-to-market ratios.

Question 5

Make some assessment of the regression assumption in both the CAPM and 3-Factor models.

Answer:

- **Linearity**

  CAPM: Assumes a linear relationship between the excess return of an asset and the excess return of the market portfolio.

  3-Factor Model: Assumes a linear relationship between excess returns and three factors: the market excess return, the size factor (SMB – small minus big), and the value factor (HML – high minus low).

  Assessment: Both models generally satisfy linearity well, but for some stocks, especially those in volatile markets, linearity may not perfectly hold.

- **Homoscedasticity**

  Homoscedasticity assumes constant variance of the residuals across all levels of the independent variables.

  CAPM: Assumes that the error terms have constant variance across all levels of market returns.

  3-Factor Model: Assumes homoscedasticity across the three factors.

  Assessment: Financial time series often exhibit heteroscedasticity, especially during market volatility, which violates this assumption. ARCH or GARCH models are sometimes used to address this.

- **Independence of Errors**

  CAPM & 3-Factor Model: Both models assume that the error terms are uncorrelated (i.e., no autocorrelation).

  Assessment: In stock return data, this assumption is often violated. Autocorrelation can lead to biased estimates, especially in high-frequency data. Time-series models may be needed to correct this.

- **Normality of Errors**

  CAPM & 3-Factor Model: Both models assume that the residuals follow a normal distribution.

  Assessment: Stock returns, particularly over short periods, tend to exhibit fat tails and skewness, indicating that errors are not normally distributed. This affects hypothesis testing but is less critical for parameter estimation.

- **No Multicollinearity**

  CAPM: With only one independent variable, multicollinearity is not an issue.

  3-Factor Model: Multicollinearity can be a concern if the factors (market return, SMB, HML) are highly correlated.

  Assessment: Multicollinearity is generally not a severe issue in the 3-Factor Model since the factors capture different aspects of stock returns. However, expanded models (e.g., 5-Factor Model) may introduce multicollinearity risks.

- **Exogeneity**

  CAPM & 3-Factor Model: Both models assume that the independent variables (market return, SMB, HML) are not correlated with the error terms (i.e., exogenous).

  Assessment: This assumption typically holds, as the factors are based on external market information, though endogeneity can occur if asset returns feedback into the factors.

Question 6

Describe a method to modernize the Fama-French model and implement this model.

1. Describe your analysis and the results.

Answer:

As the market has evolved, more factors are added to better capture modern financial dynamics. A common approach to modernize the Fama-French Model is to add two more factors, resulting in the Fama-French

5-Factor Model. The added factors are:

- **Profitability**: Robust minus Weak (RMW), which compares the returns of firms with robust profitability to those with weak profitability.
- **Investment**: Conservative minus Aggressive (CMA), which compares the returns of firms that are conservative to those with aggressive investment strategies.

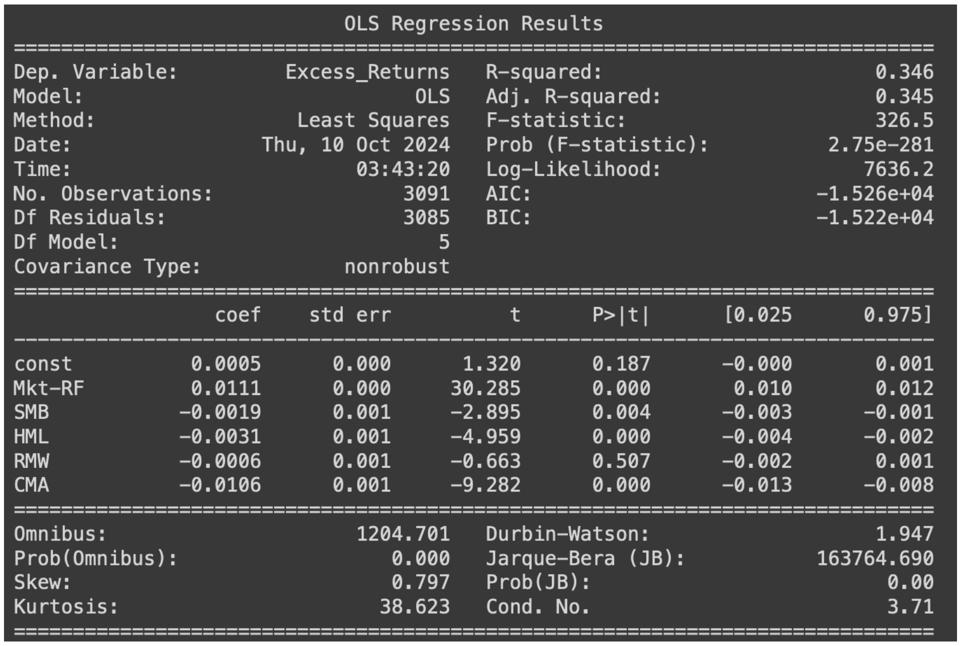
The Fama-French 5-Factor Model is represented as:

$$R_i - R_f = \alpha + \beta_m(R_m - R_f) + \beta_{SMB} \cdot SMB + \beta_{HML} \cdot HML + \beta_{RMW} \cdot RMW + \beta_{CMA} \cdot CMA + \epsilon$$

Where:

- R_i is the return of the asset/portfolio i,
- R_f is the risk-free rate,
- R_m is the return of the market portfolio,
- SMB is the size factor (Small minus Big),
- HML is the value factor (High minus Low),
- RMW is the profitability factor (Robust minus Weak),
- CMA is the investment factor (Conservative minus Aggressive),
- β's represent the sensitivities of the stock returns to these factors, and
- ϵ is the error term.

Then we ran the python code to get the result, see Appendix A.5. for the code. We chose daily returns for META as an example. Below is the analysis and the results.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:         Excess_Returns   R-squared:                       0.346
Model:                            OLS   Adj. R-squared:                  0.345
Method:                 Least Squares   F-statistic:                     326.5
Date:                Thu, 10 Oct 2024   Prob (F-statistic):          2.75e-281
Time:                        03:43:20   Log-Likelihood:                 7636.2
No. Observations:                3091   AIC:                        -1.526e+04
Df Residuals:                    3085   BIC:                        -1.522e+04
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0005      0.000      1.320      0.187      -0.000       0.001
Mkt-RF         0.0111      0.000     30.285      0.000       0.010       0.012
SMB           -0.0019      0.001     -2.895      0.004      -0.003      -0.001
HML           -0.0031      0.001     -4.959      0.000      -0.004      -0.002
RMW           -0.0006      0.001     -0.663      0.507      -0.002       0.001
CMA           -0.0106      0.001     -9.282      0.000      -0.013      -0.008
==============================================================================
Omnibus:                     1204.701   Durbin-Watson:                   1.947
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           163764.690
Skew:                           0.797   Prob(JB):                         0.00
Kurtosis:                      38.623   Cond. No.                         3.71
==============================================================================
```

Figure 2: OLS Regression Results for META

- The R-squared number is 0.346, meaning that approximately 34.6% of the dependent variable is explained by the five factors (independent variables).
- The p-value of Mkt-RF is 0, which is much smaller than 0.05, meaning that the coefficient of Mkt-RF is statistically significant. A positive and significant coefficient indicates that the stock's return has a strong and positive correlation with market returns. For every 1% increase in the market excess return, the stock's excess return increases by 1.11%.
- The p-value of SMB is 0.004, which is much smaller than 0.05, meaning that the coefficient of SMB is statistically significant. A negative coefficient for SMB suggests that the stock is more aligned with larger-cap stocks rather than small-cap stocks.
- The p-value of HML is 0, which is much smaller than 0.05, meaning that the coefficient of HML is statistically significant. The negative coefficient indicates that the stock has a growth orientation, rather than a value orientation.
- The p-value of RMW is 0.507, which is much larger than 0.05, meaning that the coefficient of RMW is not statistically significant. This suggests that the stock has little to no relationship with firms with robust profitability compared to those with weak profitability.
- The p-value of CMA is 0, which is much smaller than 0.05, meaning that the coefficient of CMA is statistically significant. A negative coefficient for CMA suggests that the stock is more aligned with aggressive investment, rather than conservative investment.

Question 7

Use the top 10 companies in the S&P 500 by market cap to calculate the efficient frontier and find and label the tangent portfolio.

1. Give the optimal weights and describe which risk-free rate you used and why.

Answer:

We write the code to deal with this question, for the details of the code refer to Appendix A.6.

- Initial Notes: for this exercise, we are using the top 10 companies in the S&P 500 by market cap. In order, these are: Apple, Microsoft, Nvidia, Google, Amazon, Meta, Berkshire Hathaway, Eli Lilly, Broadcom and Tesla. Furthermore, our answer implemented the expected return model as seen in class:

$$E(R_p) = \sum \omega_i E(R_i)$$

  $E(R_p)$ = Expected return of the portfolio

  ω_i = weight of an asset, i

  $E(R_i)$ = Expected return of an asset, i

- The real challenge this question posed was the implementation of the weight optimization of the portfolio. As discussed, this can be computationally intensive and cannot scale in large portfolios, especially considering that modifying the portfolio implies a rerun of the whole program. For this reason, we implemented our solution using a Monte Carlo simulation of 100,000 portfolios to ensure a critical mass of points.

- After running the simulation, we computed the Sharpe ratio as a basis to determine what was the weight optimization, and seeked to maximize the ratio to determine this precisely.

- Throughout the whole simulation, the risk free rate used was the mean for the last 504 days (approximately 2 trading years), found in the CSV provided. We took this window in particular as it represented a recent and relevant period for the analysis, capturing both short-term market dynamics and ensuring that the risk-free rate aligns with the same timeframe as the stock data. In turn, providing a more accurate reflection of prevailing interest rate conditions during this period. The average daily effective risk rate in this period was 0.0187%.

- Finally, our results are represented in the following table and graph, where we can see the final weights for the winning portfolio in the simulation, and a visualization of the efficient frontier cloud, and its tangent.

Stock	**Weight**
AAPL	0.067
MSFT	0.010
NVDA	0.254
GOOG	0.038
AMZN	0.039
META	0.069
BRK/B	0.091
LLY	0.345
AVGO	0.066
TSLA	0.020

Table 6: Best Portfolio Composition

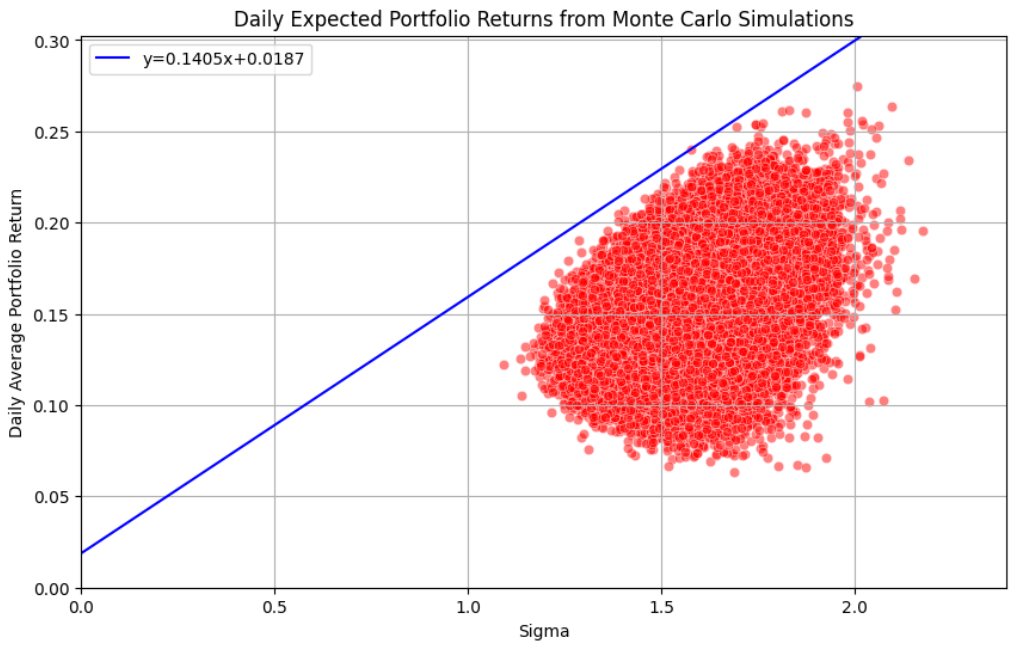


Figure 3: Efficient Frontier, n=100000

Question 8 (For 682 only)

Attempt the CAPM and the 3-Factor Model on at least one stock with:

1. lasso regression,
2. ridge regression,
3. elastic net regression.
4. Discuss the differences between each approach and which you feel is most appropriate.

Answer:

We write the code to deal with this question, for the details of the code refer to Appendix A.7.

- This question is very interesting as it addresses the nature of each of the three regressions and why they came to be. As a brief introduction, each regression model looks to penalize the betas in the linear regression to achieve different effects. We will explain how each works in the next bullets, but as a small clarification, the models make more sense when studying the FF3, rather than the CAPM. The reason behind this is very simple, which is that FF3 has 3 betas to optimize, whereas CAPM has only one, which disallows some of the models to alter the values too much.
- **Lasso:** the Lasso Regression model will study the absolute value of each of the Betas and shrink (and nullify) them by a certain coefficient to simplify the model. It is the most aggressive of the three regression models we are exploring and it is particularly useful in dimension reduction problems for high n-dimensional functions (not our case precisely).
- **Ridge:** the Ridge regression will use a similar approach in adding the coefficients and determining a penalization value, but in this case, the addition would be done on the square values of Betas, meaning that the penalization can never be greater than any of the coefficients, which results in a shrinking of the weights, but not in a nullification.
- **Elastic Net:** the Elastic Net is a weighted combination of both models. To implement it, we determine which relative weight to apply to each of the both models and in turn we will receive a penalization that, when extreme, may nullify Betas, but in other cases may not. It is useful when we predict some irrelevant factors but are not completely sure about them. In functions with correlated Betas, we may want to drop some, and models such as the Elastic Net could be a good way to reduce dimensionality.

- The following table contains the results for the FF3 for the stock of Apple. What we can see from the table is that the Lasso Model found insignificant the SMB beta and eliminated it. This is also confirmed by the fact that Ridge asigned it very little weight to it (shrank it but can't eliminate it) and Elastic Net eliminated it altogether likely because of the effect of the Lasso penalty component. Compared to the results of question 2, the coefficient for the Mkt-RF was proportionally exacerbated, whilst the other two coefficients were diluted (or nullified in some cases). The final reflection for these 3 regression models in the Fama French are that neither SMB or HML play a significant role to Apple's stock, and therefore the expected behaviour is that CAPM and FF3 models output similar results. Computationally, this favors the CAPM model, which due to its simplicity, will have better time-complexity results.

Model	**Mkt-RF**	**SMB**	**HML**
Lasso	1.32	0.00	-0.31
Ridge	1.43	-0.04	-0.42
Elastic Net	1.30	0.00	-0.34

Table 7: FF3 Weights under Lasso, Ridge, and Elastic Net regression models

- For the sake of consistency, the following table shows the results obtained using the three regression alternatives in the CAPM model. As expected, the weights for the Mkt-Rf are very similar than in the Fama French model, with only subtle changes, which likely suggests that this is the most important component for this asset. We can back this argument also by comparing the magnitude of the weights in the previous table. We can see that in all models Mkt-RF has the highest beta.

Model	**Mkt-RF**
Lasso	1.31
Ridge	1.41
Elastic Net	1.29

Table 8: CAPM Weights under Lasso, Ridge, and Elastic Net regression models

- Finally, lets discuss what each of the 3 coefficients represent and what conclusions they allow us to draw out from our results, as a means of connecting our results to question 4.

  1. The Market Risk Premium coefficient is the sensibility of the asset to a variation in the S&P 500. When this coefficient is close to 1, it means that a 1% increase in the market, creates a 1%

increase in the asset. In our case, our coefficient is higher than 1, meaning it is highly sensitive to market movements. Given this model uses a rearview-mirror approach, the coefficient is possibly recreating the effect that the Apple Stock has beaten the market when the market was on the rise (the majority of the cases in the lifecycle of the asset).

2. The Small Minus Big coefficient measures the excess of small cap stocks over large cap stocks. High values of the SMB coefficient mean the asset's value is correlated with smaller capped company's movements. Small SMB values mean there is no correlation with smaller assets, and negative correlation would imply that the increase of small-cap assets is detrimental for the asset being studied. In the case of Apple, we obtained the expected result: the SMB coefficeint was small or null, as it is the largest capped stock, and it would be very unlikely for its expected returns to be explained by the variation of less valuable assets. Therefore, it makes sense that Lasso and Elastic Net methods nullified this component and Ridge minimized it.
3. The High Minus Low coefficient basically measures the asset's return compared to their book value and earnings. Companies with high HML are perceived to be undervalued, and are therefore forecasted to increase in value. For this precise reason is that the FF3 model incorporates it as part of the expected return equation. In the case of the Apple stock, all 3 regression models returned an HML smaller than 0, meaning that Apple is thought of as a 'growth stock'. Growth stocks are expected to grow in the long run, which is why the market sentiment tends to be that they are overpriced, as opposed to high HML values which imply the opposite. In Apple's case, the negative HML is relevant in 'correcting' the expected return due to the pereived overpricing of the asset.

Question 9 (For 682 only)

Use the top 20 companies in the S&P 500 by market cap to calculate the efficient frontier and find and label the tangent portfolio.

1. Give the optimal weights and describe which risk-free rate you used and why.

Answer:

- Initial Notes: We are using a portfolio of the top 20 companies and are evaluating various portfolio returns with the following:

$$E(R_p) = \sum \omega_i E(R_i)$$

$E(R_p)$ = Expected return of the portfolio

ω_i = weight of an asset, i

$E(R_i)$ = Expected return of an asset, i

- From a Monte Carlo simulation resulting in 100,000 possible portfolios with varying weights for the stocks of the top 20 companies, we found the optimal weights of a portfolio for the last 504 trading days to be as shown in Table 9. See the Efficient Frontier figure below for a visual of the Monte Carlo simulation. See Appendix A.8. for the relevant code.

- The risk free rate that was used was the average risk free rate given in the "F-F_Research_Data_Factors_daily.CSV" for our window, which came out to be 0.018$\%$ for the past 504 days. Note that this is the daily risk free rate, and corresponds to the T-Bill daily returns that correlate to the 1-month T-Bill rate.

Stock	**Weight**
AAPL	0.032
MSFT	0.053
NVDA	0.101
GOOG	0.024
AMZN	0.001
META	0.055
BRK/B	0.061
LLY	0.100
AVGO	0.108
TSLA	0.008
WMT	0.114
JPM	0.058
V	0.002
UNH	0.019
XOM	0.103
ORCL	0.066
MA	0.023
HD	0.017
PG	0.033
COST	0.019

Table 9: Best Portfolio Composition

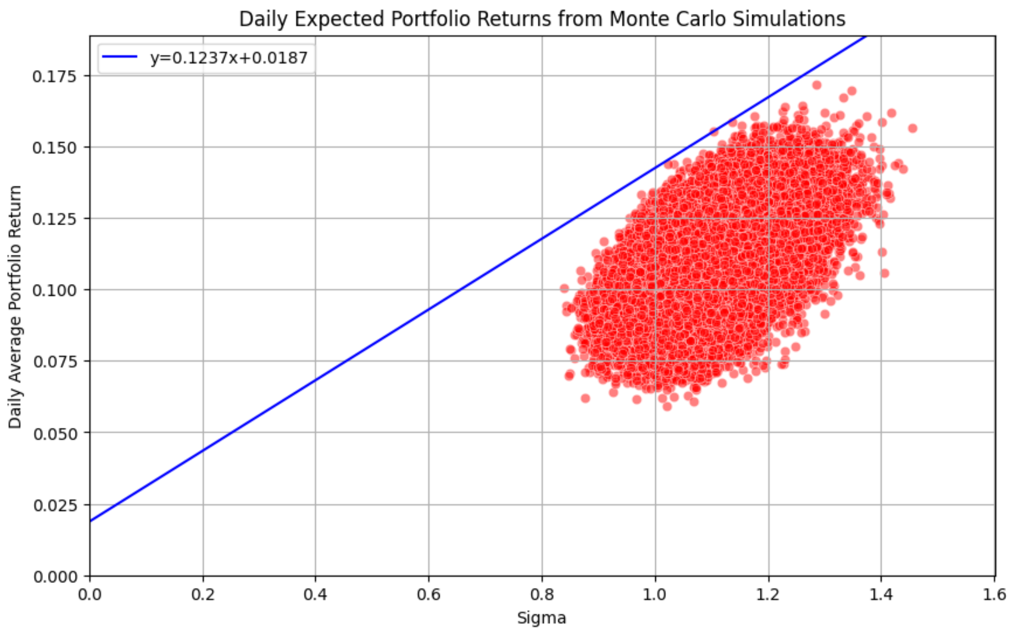


Figure 4: Efficient Frontier, n=100000

References

- Fama-French Data Library: U.S. Research Returns
- Fama and French Three-Factor Model on Investopedia

A Appendix: Python Code Listings

A.1 Data Loading and Cleaning Code

Listing 1: Code for data loading and cleaning

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load Fama-French 3-Factor data
ff_data = pd.read_csv("/content/F-F_Research_Data_Factors_daily.CSV",
    skiprows=4)

```

```
# Drop rows with missing values
ff_cleaned_data = ff_data.dropna()

# Convert the 'Unnamed: 0' column to datetime format
ff_cleaned_data['Date'] = pd.to_datetime(ff_cleaned_data['Unnamed: 0'],
    format='%Y%m%d')

# Remove 'Unnamed: 0' column
ff_cleaned_data = ff_cleaned_data.drop(columns=['Unnamed: 0'])

# Set the index as the 'Date' for easier merging
ff_cleaned_data = ff_cleaned_data.set_index('Date')

# Display cleaned data
print(ff_cleaned_data)

# Load stock data from Excel
stock_data = pd.read_excel("Homework 2 data.xlsx", sheet_name="Data")

# Display stock data
print(stock_data)
```

A.2 Question 1 Code

Listing 2: Code for calculating the coefficients

```
# Define the columns of date and the corresponding price for selected
    10 stocks
stock_columns = {
    'AAPL': ['AAPL US Equity', 'AAPL US Equity Closing Price'],
    'MSFT': ['MSFT US Equity', 'MSFT US Equity Closing Price'],
    'NVDA': ['NVDA US Equity', 'NVDA US Equity Closing Price'],
    'GOOG': ['GOOG US Equity', 'GOOG US Equity Closing Price'],
    'MA': ['MA US Equity', 'MA US Equity Closing Price'],
    'HD': ['HD US Equity', 'HD US Equity Closing Price'],
    'PG': ['PG␣US␣Equity', 'PG␣US␣Equity␣Closing␣Price'],
    'COST': ['COST␣UW␣Equity', 'COST␣UW␣Equity␣Closing␣Price'],
    'JPM': ['JPM␣US␣Equity', 'JPM␣US␣Equity␣Closing␣Price'],
    'V': ['V␣US␣Equity', 'V␣US␣Equity␣Closing␣Price']
}

from sklearn.linear_model import LinearRegression
import numpy as np

# Dictionary to store regression results
regression_results = {}

# Clean the stock data, calculate percent returns, and merge with Fama-
   French data
for stock, cols in stock_columns.items():
    date_col, price_col = cols

    # Extract date and closing price for the stock
    stock_df = stock_data[[date_col, price_col]].dropna()

    # Convert date column to datetime
    stock_df[date_col] = pd.to_datetime(stock_df[date_col])

    # Set date as index
    stock_df = stock_df.set_index(date_col)

    # Calculate percent returns
    stock_returns = stock_df.pct_change().dropna() * 100
    rename_col = stock + '␣' + 'US␣Equity␣Percent␣Returns'
    stock_returns.columns = [rename_col]

    # Align stock returns with Fama-French data by date
    stock_returns.index = pd.to_datetime(stock_returns.index)

    # Merge stock returns with Fama-French data on the date index
    merged_data = stock_returns.merge(ff_cleaned_data, left_index=True,
        right_index=True)

    # Subtract risk-free rate to get excess returns
    excess_returns = merged_data[rename_col] - merged_data['RF']

    # Prepare data for regression (X and y)
    X_capm = merged_data[['Mkt-RF']].values  # CAPM uses only market
       excess return
    X_ff3 = merged_data[['Mkt-RF', 'SMB', 'HML']].values  # FF3 uses
       market, size, and value factors
    y = excess_returns.values.reshape(-1, 1)

    # CAPM: Linear regression on Mkt-RF
    capm_model = LinearRegression().fit(X_capm, y)

    # Fama-French 3-Factor: Linear regression on Mkt-RF, SMB, HML
    ff3_model = LinearRegression().fit(X_ff3, y)

    # Store the results for CAPM and FF3
    regression_results[stock] = {
        'CAPM': capm_model,
        'FF3': ff3_model
    }

# Extract coefficients (Betas) for each stock and display them
for stock, models in regression_results.items():
    print(f"{stock}␣CAPM␣Coefficients:\nIntercept␣(Alpha):␣{models['
       CAPM'].intercept_[0]:.3f}\n"
          f"Beta␣(Mkt-RF):␣{models['CAPM'].coef_[0][0]:.3f}\n")

    print(f"{stock}␣Fama-French␣3-Factor␣Coefficients:\nIntercept␣(
       Alpha):␣{models['FF3'].intercept_[0]:.3f}\n"
```

```
            f"Beta␣(Mkt-RF):␣{models['FF3'].coef_[0][0]:.3f}\n"
            f"Beta␣(SMB):␣{models['FF3'].coef_[0][1]:.3f}\n"
            f"Beta␣(HML):␣{models['FF3'].coef_[0][2]:.3f}\n")

    print("-" * 50)
```

A.3 Question 2 Code

Listing 3: Code for calculating the coefficients and t-statistics

```
from statsmodels.api import OLS, add_constant

# Dictionary to store rounded t-statistics and coefficients
regression_results_with_tstats = {}

# Loop through the stocks to calculate coefficients and t-statistics
for stock, cols in stock_columns.items():
    date_col, price_col = cols

    # Extract date and closing price for the stock
    stock_df = stock_data[[date_col, price_col]].dropna()
    stock_df[date_col] = pd.to_datetime(stock_df[date_col])
    stock_df = stock_df.set_index(date_col)

    # Calculate percent returns
    stock_returns = stock_df.pct_change().dropna() * 100
    rename_col = stock + '␣Percent␣Returns'
    stock_returns.columns = [rename_col]

    # Merge stock returns with Fama-French data
    merged_data = stock_returns.merge(ff_cleaned_data, left_index=True,
        right_index=True)

    # Calculate excess returns
    excess_returns = merged_data[rename_col] - merged_data['RF']

    # Prepare data for CAPM (Mkt-RF) and FF3 (Mkt-RF, SMB, HML)
    X_capm = merged_data[['Mkt-RF']].values
    X_ff3 = merged_data[['Mkt-RF', 'SMB', 'HML']].values
    y = excess_returns.values.reshape(-1, 1)

    # CAPM: OLS to get coefficients and t-statistics
    capm_model = OLS(y, add_constant(X_capm)).fit()
    capm_tstats = np.round(capm_model.tvalues, 3)

    # Fama-French 3-Factor: OLS to get coefficients and t-statistics
    ff3_model = OLS(y, add_constant(X_ff3)).fit()
    ff3_tstats = np.round(ff3_model.tvalues, 3)

    # Store coefficients and t-statistics
    regression_results_with_tstats[stock] = {
        'CAPM␣Coefficients': np.round(capm_model.params, 3),
        'CAPM␣t-Statistics': capm_tstats,
        'FF3␣Coefficients': np.round(ff3_model.params, 3),
        'FF3␣t-Statistics': ff3_tstats
    }

# Extract coefficients and t-statistics for display
for stock, results in regression_results_with_tstats.items():
    print(f"{stock}␣CAPM␣Coefficients:\n{results['CAPM␣Coefficients']}\
        n"
          f"CAPM␣t-Statistics:\n{results['CAPM␣t-Statistics']}\n")
    print(f"{stock}␣FF3␣Coefficients:\n{results['FF3␣Coefficients']}\n"
          f"FF3␣t-Statistics:\n{results['FF3␣t-Statistics']}\n")
    print("-" * 50)
```

A.4 Question 3 Code

Listing 4: Code for calculating rolling beta

```
stock = 'NVDA' # choose stock

def process_stock_data(stock):

  stock_chosen = f'{stock}␣US␣Equity␣Closing␣Price' if stock != 'COST' else 'COST␣UW␣Equity␣Closing␣Price'
  stock_symbol = f'{stock}␣US␣Equity' if stock != 'COST' else 'COST␣UW␣Equity'

  chosen_stock_data = stock_data[[stock_symbol, stock_chosen]].copy()

  chosen_stock_data.columns = ['Date', f'{stock}␣Price']
  ## merge with FF factors
  chosen_stock_data = chosen_stock_data.merge(ff_cleaned_data, how = 'left', on = 'Date')

  ## filter data after 2000
  chosen_stock_data = chosen_stock_data[chosen_stock_data['Date'] >'2000-01-01']

  chosen_stock_data[f'{stock}␣Return'] = chosen_stock_data[f'{stock}␣Price'].pct_change()*100

  ## calculate excess returm
  chosen_stock_data[f'{stock}␣Excess␣Return'] = chosen_stock_data[f'{stock}␣Return'] - chosen_stock_data['RF'] # is this supposed to be *100?
  chosen_stock_data.dropna(inplace=True)

  return chosen_stock_data

chosen_stock_data = process_stock_data(stock)
chosen_stock_data

```

```python
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns
def rolling_regression_FF3(df, stock, window=252):
  ## make beta dictionary
  betas = {
        'Beta_Mkt': [],
        'Beta_SMB': [],
        'Beta_HML': [],
        'Date' : [],
        }

  lin_reg = LinearRegression()

  # make FF3F model for each 252 window after year 2000
  for i in range(len(df) - window):
    window_data = df.iloc[i:i+window]
    X = window_data[['Mkt-RF', 'SMB', 'HML']]
    y = window_data[f'{stock}␣Excess␣Return']
    lin_reg.fit(X, y)
    ## append coefficients to dictionary
    betas['Beta_Mkt'].append(lin_reg.coef_[0])
    betas['Beta_SMB'].append(lin_reg.coef_[1])
    betas['Beta_HML'].append(lin_reg.coef_[2])
    betas['Date'].append(df.iloc[i+window]['Date'])
  return betas


rolling_betas = rolling_regression_FF3(chosen_stock_data, stock = stock
    )

betas_df = pd.DataFrame(rolling_betas)

import matplotlib.pyplot as plt
import seaborn as sns
sns.lineplot(data=betas_df, x='Date', y='Beta_Mkt', label='Mkt')
sns.lineplot(data=betas_df, x='Date', y='Beta_SMB', label='SMB')
sns.lineplot(data=betas_df, x='Date', y='Beta_HML', label='HML')

# Add the legend automatically based on the labels
plt.legend()

# Add title and labels
plt.title(f'Rolling␣Year␣Window␣Fama-French␣3-Factor␣Betas␣for␣{stock}'
    )
plt.xlabel('Date')
plt.ylabel('Beta␣Value')

# Rotate x-axis labels for readability
plt.xticks(rotation=45)

# Display the plot
plt.tight_layout()
plt.show()
```

A.5 Question 6 Code

Listing 5: Code for Fama-French 5-Factor Model on META data

```

import pandas as pd
import statsmodels.api as sm

ff_factors_url = 'https://mba.tuck.dartmouth.edu/pages/faculty/ken.
    french/ftp/F-F_Research_Data_5_Factors_2x3_daily_CSV.zip'
ff_factors = pd.read_csv(ff_factors_url, skiprows=3)

# Step 1: Clean up the data
ff_factors.columns = ['Date', 'Mkt-RF', 'SMB', 'HML', 'RMW', 'CMA', 'RF
```

```
    ']
ff_factors['Date'] = pd.to_datetime(ff_factors['Date'], format='%Y%m%d'
    ) # Convert 'Date' column to datetime
# Assuming stock_data has been loaded as shown in your screenshot
# Calculate daily returns for META as an example
stock_data['META_Returns'] = stock_data['META US Equity Closing Price'
    ].pct_change()

# Remove any rows with missing values (for simplicity)
stock_data = stock_data.dropna(subset=['META_Returns'])
# Align the dates and merge the data
stock_data['Date'] = pd.to_datetime(stock_data['META US Equity'])
merged_data = pd.merge(stock_data, ff_factors, on='Date')

# Calculate excess returns (META returns - risk-free rate)
merged_data['Excess_Returns'] = merged_data['META_Returns'] -
    merged_data['RF'] / 100

# Set up the independent variables (factors) and dependent variable (
    Excess Returns)
X = merged_data[['Mkt-RF', 'SMB', 'HML', 'RMW', 'CMA']]
y = merged_data['Excess_Returns']

# Add a constant (intercept) to the model
X = sm.add_constant(X)

# Run the regression
model = sm.OLS(y, X).fit()

# Print the summary of the results
print(model.summary())
```

A.6 Question 7 Code

Listing 6: Code for top 10 companies to calculate the efficient frontier and find and label the tangent portfolio.

```
stock_list = [
    'AAPL', 'MSFT', 'NVDA', 'GOOG', 'AMZN', 'META', 'BRK/B', 'LLY',
    'AVGO', 'TSLA'
]

def process_stock_data(stock):

  stock_chosen = f'{stock}␣US␣Equity␣Closing␣Price' if stock != 'COST'
      else 'COST␣UW␣Equity␣Closing␣Price'
  stock_symbol = f'{stock}␣US␣Equity' if stock != 'COST' else 'COST␣UW␣
      Equity'

  chosen_stock_data = stock_data[[stock_symbol, stock_chosen]].copy()

  chosen_stock_data.columns = ['Date', f'{stock}␣Price']
  ## merge with FF factors
  chosen_stock_data = chosen_stock_data.merge(ff_cleaned_data, how = '
      left', on = 'Date')

  ## filter data after 2000
  chosen_stock_data = chosen_stock_data[chosen_stock_data['Date'] >'
      2000-01-01']

  chosen_stock_data[f'{stock}␣Return'] = chosen_stock_data[f'{stock}␣
      Price'].pct_change()*100

  ## calculate excess returm
  chosen_stock_data[f'{stock}␣Excess␣Return'] = chosen_stock_data[f'{
      stock}␣Return'] - chosen_stock_data['RF'] # is this supposed to be
       *100?
  chosen_stock_data.dropna(inplace=True)
  return chosen_stock_data
```

```

def get_stock_returns(window=252):
    '''
    Returns a DataFrame of all 20 stock returns
    '''
    excess_returns_df = pd.DataFrame()

    for stock in stock_list:
        stock_data_processed = process_stock_data(stock)[-window:]

        stock_excess_returns = stock_data_processed[['Date', f'{stock}␣
            Excess␣Return']].copy()
        stock_excess_returns.set_index('Date', inplace=True)

        stock_excess_returns = stock_excess_returns.rename(columns={f'{
            stock}␣Excess␣Return': stock})


        if excess_returns_df.empty:
            excess_returns_df = stock_excess_returns
        else:
            excess_returns_df = excess_returns_df.join(
                stock_excess_returns, how='outer')

    return excess_returns_df

def get_portfolio_return(weights: np.array, asset_returns: np.array) ->
      float:
    '''
    Returns EV of a portfolio, given EV of assets and weights
    '''
    return weights.T @ asset_returns

def get_weights(n = 10):
    '''
    Returns n (default=10) random numbers that sum to 1
    '''
    weights = np.random.rand(n)
    weights /= np.sum(weights)
    return weights

def efficient_frontier(window=504, n = 10000):

    '''
    Inputs:
      expected_returns: np.array of expected returns correlating to df['
          Stock']
      weights: np.array of weights correlating to df['Expected Return']
      n: number of portfolios to generate, default = 10000

    Returns:
      best_weights: np.array of best weights
      best_return: float of best return
    '''


    best_return = 0
    best_weights = None
    portfolio_returns = []
    portfolio_weights = []
    portfolio_std = []
    best_sharpe_index = 0
    best_return = 0.0
    best_weights = np.array([])
    sharpe_ratio = []
    best_sharpe_index = 0


```

```
# Getting current data
stock_returns = get_stock_returns(window=window)

risk_free_rate = ff_cleaned_data[-(window):]['RF'].mean()
expected_returns = stock_returns.mean()
cov_returns = stock_returns.cov()


# each n defines a portfolio, each portfolio has
# # sharpe, E(return), weights, and stdev that need to be calc/
    plotted

for i in range(n):
  # get and save weights
  weights = get_weights()
  portfolio_weights.append(weights)

  # calculate return and std
  portfolio_return = get_portfolio_return(weights, expected_returns)
  portfolio_returns.append(portfolio_return)

  stdev = np.sqrt(np.dot(weights.T, np.dot(cov_returns, weights)))
  portfolio_std.append(stdev)

  sharpe_ratio.append((portfolio_return - risk_free_rate) / stdev)

  if sharpe_ratio[i] > sharpe_ratio[best_sharpe_index]:
     best_sharpe_index = i
     best_return = portfolio_return
     best_weights = weights
     best_stdev = stdev
     best_sharpe = sharpe_ratio[i]

# Slope for CAL
```

```
slope = best_sharpe # ((best_return - risk_free_rate )/ best_stdev)

# prints
print('*'*50)
print('Efficient Frontier')
print('*'*50)
print(f'Ran {n} simulations\n')
print(f'Best Return:\n{best_return}\n')
print(f'Best Weights:\n{best_weights}\n')
print(f'Best Sharpe Ratio (slope): ', round(slope, 5),'\n')
print(f'Risk Free rate (intercept) {round(risk_free_rate, 5)}\n')


print('*'*50)
print('Best Portfolio Composition:\n')
for i in range(len(portfolio_weights[best_sharpe_index])):
  print(stock_returns.columns[i], round(portfolio_weights[
      best_sharpe_index][i], 5))
print('*'*50)


# plot
plt.figure(figsize=(10,6))
sns.scatterplot(x=portfolio_std, y=portfolio_returns, color='red',
    alpha=0.5)
plt.xlabel('Sigma')
plt.ylabel('Daily Average Portfolio Return')
plt.title('Daily Expected Portfolio Returns from Monte Carlo 
    Simulations')


# plotting CAL
x_line = np.linspace(0, 7, 1000)
y_line = x_line * slope + risk_free_rate
```

```
    plt.plot(x_line, y_line, color='blue', label=f'y={round(slope, 4)}x+{
        round(risk_free_rate, 4)}')


    plt.xlim(0, np.max(portfolio_std )*1.1) # maybe change this? idk bc
        we wanna be able to see (0,0)
    plt.ylim(0, np.max(portfolio_returns)*1.1) # maybe change this? idk
        bc we wanna be able to see (0,0)
    plt.grid(True)
    plt.legend(loc='upper left')
    plt.show()


    return {'best_weights': portfolio_weights[best_sharpe_index], '
        best_return': best_return}


efficient_frontier(window=504, n = 100000)
```

A.7 Question 8 Code

Listing 7: Code for CAPM and 3 factor model with different regression

```
def process_stock_data(stock):

    stock_chosen = f'{stock} US Equity Closing Price' if stock != 'COST'
        else 'COST UW Equity Closing Price'
    stock_symbol = f'{stock} US Equity' if stock != 'COST' else 'COST UW
        Equity'

    chosen_stock_data = stock_data[[stock_symbol, stock_chosen]].copy()

    chosen_stock_data.columns = ['Date', f'{stock} Price']
    ## merge with FF factors
    chosen_stock_data = chosen_stock_data.merge(ff_cleaned_data, how = '
        left', on = 'Date')

    ## filter data after 2000
    chosen_stock_data = chosen_stock_data[chosen_stock_data['Date'] >'
        2000-01-01']

    chosen_stock_data[f'{stock}␣Return'] = chosen_stock_data[f'{stock}␣
        Price'].pct_change()*100

    ## calculate excess returm
    chosen_stock_data[f'{stock}␣Excess␣Return'] = chosen_stock_data[f'{
        stock}␣Return'] - chosen_stock_data['RF'] # is this supposed to be
         *100?
    chosen_stock_data.dropna(inplace=True)
    return chosen_stock_data

import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.linear_model import Lasso, Ridge, ElasticNet
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

stock = 'AAPL'
chosen_stock_data = process_stock_data(stock)

# Independent variables (Mkt-RF, SMB, HML) - Fama-French 3-Factor
X_ff3 = chosen_stock_data[['Mkt-RF', 'SMB', 'HML']]

# Dependent variable (Excess Return of the stock)
y = chosen_stock_data[f'{stock}␣Excess␣Return']

# CAPM: Only Market Excess Return
X_capm = chosen_stock_data[['Mkt-RF']]
X_capm = sm.add_constant(X_capm)  # Add constant

# Fama-French 3-Factor Model: Add constant to independent variables
X_ff3 = sm.add_constant(X_ff3)

# Split data into training and testing sets (80/20 split)
X_train_capm, X_test_capm, y_train, y_test = train_test_split(X_capm, y
    , test_size=0.2, random_state=42)
X_train_ff3, X_test_ff3, y_train_ff3, y_test_ff3 = train_test_split(
    X_ff3, y, test_size=0.2, random_state=42)

# Standardize the data for regularization
scaler = StandardScaler()
X_train_capm_scaled = scaler.fit_transform(X_train_capm)
X_test_capm_scaled = scaler.transform(X_test_capm)
X_train_ff3_scaled = scaler.fit_transform(X_train_ff3)
X_test_ff3_scaled = scaler.transform(X_test_ff3)

# Step 1: Apply Regularization on CAPM Model
# Lasso Regression (L1 regularization) on CAPM
lasso_capm = Lasso(alpha=0.1)
lasso_capm.fit(X_train_capm_scaled, y_train)
lasso_capm_pred = lasso_capm.predict(X_test_capm_scaled)

# Ridge Regression (L2 regularization) on CAPM
ridge_capm = Ridge(alpha=0.1)
ridge_capm.fit(X_train_capm_scaled, y_train)
ridge_capm_pred = ridge_capm.predict(X_test_capm_scaled)

# Elastic Net Regression (Combination of L1 and L2) on CAPM
elastic_net_capm = ElasticNet(alpha=0.1, l1_ratio=0.5)
elastic_net_capm.fit(X_train_capm_scaled, y_train)
elastic_net_capm_pred = elastic_net_capm.predict(X_test_capm_scaled)

```

```
# Step 2: Apply Regularization on Fama-French 3-Factor Model
# Lasso Regression (L1 regularization) on FF3
lasso_ff3 = Lasso(alpha=0.1)
lasso_ff3.fit(X_train_ff3_scaled, y_train_ff3)
lasso_ff3_pred = lasso_ff3.predict(X_test_ff3_scaled)

# Ridge Regression (L2 regularization) on FF3
ridge_ff3 = Ridge(alpha=0.1)
ridge_ff3.fit(X_train_ff3_scaled, y_train_ff3)
ridge_ff3_pred = ridge_ff3.predict(X_test_ff3_scaled)

# Elastic Net Regression (Combination of L1 and L2) on FF3
elastic_net_ff3 = ElasticNet(alpha=0.1, l1_ratio=0.5)
elastic_net_ff3.fit(X_train_ff3_scaled, y_train_ff3)
elastic_net_ff3_pred = elastic_net_ff3.predict(X_test_ff3_scaled)

# Results: Print Coefficients for CAPM and FF3 Model Regularization
print("CAPM␣Lasso␣Coefficients:", lasso_capm.coef_)
print("CAPM␣Ridge␣Coefficients:", ridge_capm.coef_)
print("CAPM␣Elastic␣Net␣Coefficients:", elastic_net_capm.coef_)

print("\nFF3␣Lasso␣Coefficients:", lasso_ff3.coef_)
print("FF3␣Ridge␣Coefficients:", ridge_ff3.coef_)
print("FF3␣Elastic␣Net␣Coefficients:", elastic_net_ff3.coef_)
```

A.8 Question 9 Code

Listing 8: Helper Functions for Data Processing and Portfolio Simulation

```
# List of stocks in the portfolio
stock_list = [
    'AAPL', 'MSFT', 'NVDA', 'GOOG', 'AMZN', 'META', 'BRK/B', 'LLY',
    'AVGO', 'TSLA', 'WMT', 'JPM', 'V', 'UNH', 'XOM', 'ORCL',
    'MA', 'HD', 'PG', 'COST'
]
```

```python

def process_stock_data(stock):
    """Processes stock data to calculate excess returns."""
    stock_chosen = f'{stock} US Equity Closing Price' if stock != 'COST' else 'COST UW Equity Closing Price'
    stock_symbol = f'{stock} US Equity' if stock != 'COST' else 'COST UW Equity'
    chosen_stock_data = stock_data[[stock_symbol, stock_chosen]].copy()
    chosen_stock_data.columns = ['Date', f'{stock} Price']
    chosen_stock_data = chosen_stock_data.merge(ff_cleaned_data, how='left', on='Date')
    chosen_stock_data = chosen_stock_data[chosen_stock_data['Date'] > '2000-01-01']
    chosen_stock_data[f'{stock} Return'] = chosen_stock_data[f'{stock} Price'].pct_change() * 100
    chosen_stock_data[f'{stock} Excess Return'] = chosen_stock_data[f'{stock} Return'] - chosen_stock_data['RF']
    chosen_stock_data.dropna(inplace=True)
    return chosen_stock_data

def get_stock_returns(window=252):
    """Generates a DataFrame of excess returns for all stocks over the specified window."""
    excess_returns_df = pd.DataFrame()
    for stock in stock_list:
        stock_data_processed = process_stock_data(stock)[-window:]
        stock_excess_returns = stock_data_processed[['Date', f'{stock} Excess Return']].copy()
        stock_excess_returns.set_index('Date', inplace=True)
        stock_excess_returns.rename(columns={f'{stock} Excess Return': stock}, inplace=True)
        excess_returns_df = excess_returns_df.join(stock_excess_returns, how='outer') if not excess_returns_df.empty else stock_excess_returns
    return excess_returns_df

def get_portfolio_return(weights, asset_returns):
    """Calculates the expected return of a portfolio given asset
       returns and weights."""
    return weights.T @ asset_returns

def get_weights(n=20):
    """Generates random portfolio weights that sum to 1."""
    weights = np.random.rand(n)
    return weights / np.sum(weights)
```

Listing 9: Efficient Frontier Simulation

```
def efficient_frontier(window=500, n=10000):
    """Simulates the efficient frontier to find the best portfolio
       composition."""
    best_sharpe_index, portfolio_returns, portfolio_std = 0, [], []
    sharpe_ratio, portfolio_weights = [], []
    stock_returns = get_stock_returns(window=window)
    risk_free_rate = ff_cleaned_data[-window:]['RF'].mean()
    expected_returns, cov_returns = stock_returns.mean(), stock_returns
       .cov()

    for _ in range(n):
        weights = get_weights()
        portfolio_weights.append(weights)
        portfolio_return = get_portfolio_return(weights,
           expected_returns)
        portfolio_stdev = np.sqrt(np.dot(weights.T, np.dot(cov_returns,
             weights)))

        portfolio_returns.append(portfolio_return)
        portfolio_std.append(portfolio_stdev)
        sharpe = (portfolio_return - risk_free_rate) / portfolio_stdev
```

```
            sharpe_ratio.append(sharpe)

            if sharpe > sharpe_ratio[best_sharpe_index]:
                best_sharpe_index = _
                best_return, best_weights = portfolio_return, weights
                best_stdev, best_sharpe = portfolio_stdev, sharpe

    print(f'Best␣Return:␣{best_return}\nBest␣Weights:␣{best_weights}\
        nBest␣Sharpe␣Ratio:␣{round(best_sharpe,␣5)}')
    print(f'Risk-Free␣Rate:␣{round(risk_free_rate,␣5)}\n')

    # Plot results
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=portfolio_std, y=portfolio_returns, color='red',
        alpha=0.5)
    plt.xlabel('Sigma')
    plt.ylabel('Daily␣Average␣Portfolio␣Return')
    plt.title('Efficient␣Frontier␣from␣Simulations')
    slope = best_sharpe
    x_line, y_line = np.linspace(0, np.max(portfolio_std)*1.1, 1000),
        np.linspace(0, 7, 1000) * slope + risk_free_rate
    plt.plot(x_line, y_line, color='blue', label=f'CAL:␣y={round(slope,
        ␣4)}x+{round(risk_free_rate,␣4)}')
    plt.grid(True)
    plt.legend(loc='upper␣left')
    plt.show()

    return {'best_weights': portfolio_weights[best_sharpe_index], '
        best_return': best_return}

# Run simulation
efficient_frontier(window=504, n=100000)
```