STAT417 Final Project

Lauren Huang

2025-03-21

Section 0

Please see "Section 0. Read in data" in the Appendix to read in all necessary data for the code.

Problem 1 Part a

An investor would like to construct different portfolios and rebalance them every 6 months, comprising of the stocks mentioned above. Specifically, the investor starting in the second half of the year 2015 and ending on December 31, 2024, would like to create the following types of portfolios, based on the historical data of the preceding semester.

1. The minimum variance portfolio
2. The tangency portfolio.
3. A portfolio whose annual expected return is 8.5%.

If for a particular semester this is not possible, mark it as NA. Plot the Sharpe ratios over time (for each semester) of the three portfolios. Also, plot the aggregate weight of the technology sector and the banking sector over time (for each semester) for the three portfolios. Comment on the results.

i

The first step is to load in the stock/ETF data (Homework 1), financial stock data (Homework 2), and combined them together across common dates. Then 13 stocks of interest can be extracted from the data. This is the data used to make calculations and will be referred to as the combined stock data.

Also load in the annualized risk-free rate data in DGS6MO.csv. This will be referred to as the risk free rate data.

Daily log returns are calculated using daily prices the from the combined stock data. Log return formula: $r_t = log(1 + R_t) = log(\frac{P_t}{P_{t-1}})$

Daily log returns are then converted to Semester log returns by averaging the daily log returns across each semester, and multiplying by 126.

The risk free rate data contains daily annualized rates. The conversion to semester scale is done by averaging the daily annualized rates and diving by 2.

Portfolio construction code is obtained from Homework 2.

Sharpe ratio formula: $\frac{R_p - R_{rf}}{\sigma_P}$

ii

Begin by calculating daily log returns using the combined stock data. Then, create a Semester column and average daily log returns per semester. Multiply the average daily log returns by 126 to convert to the Semester scale.

Meanwhile, also calculate the covariance matrix of the daily returns, and multiply by 126 to convert to a Semester scale.

The risk free rate data has daily annualized interest rates, so again add a Semester column, average the daily annualized rates and divide the rates by 2 and group by Semester to convert to Semester scale.

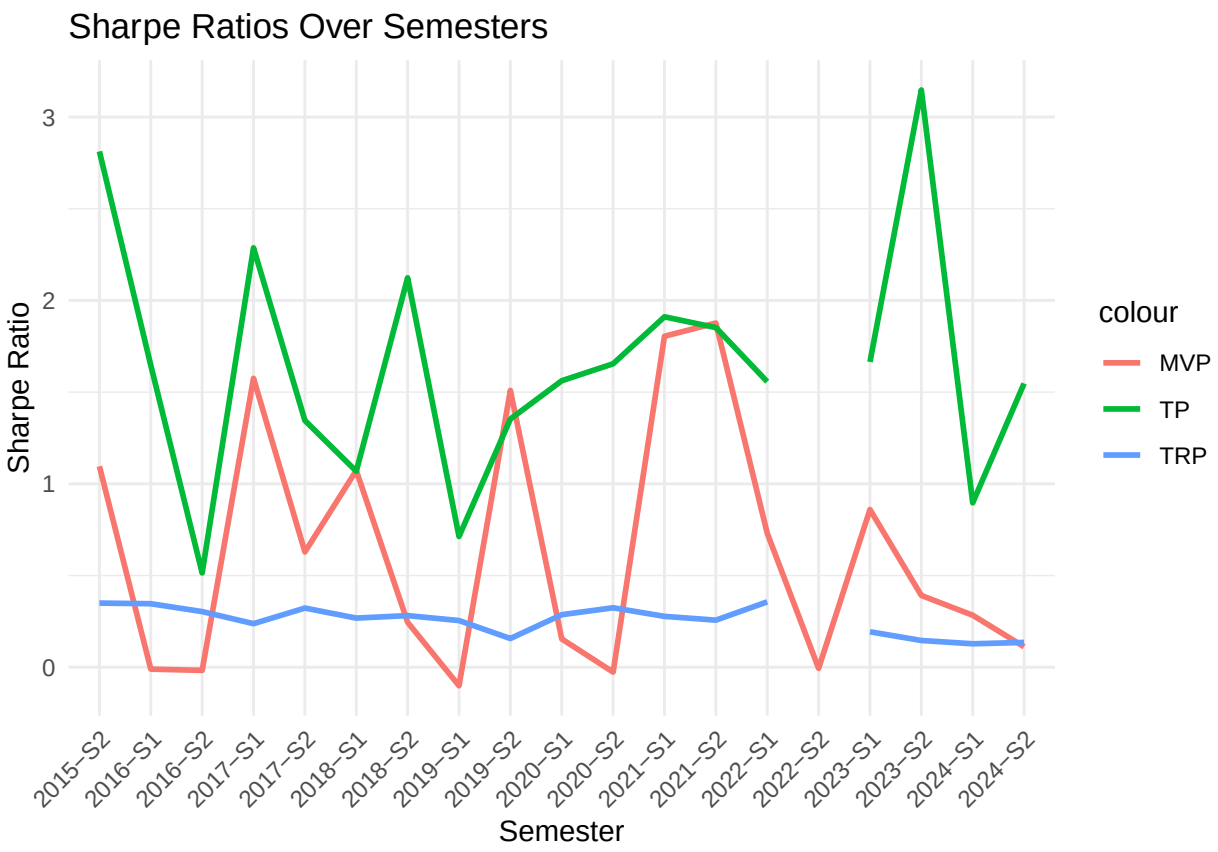
Then utilize the portfolio construction code (from the homeworks) to loop through the 19 semesters from 2015 S2 to 2024 S2. In each iteration, extract the weights, calculate the Sharpe ratio for each of the 3 portfolios (minimum variance, tangency, target return (if it exists).

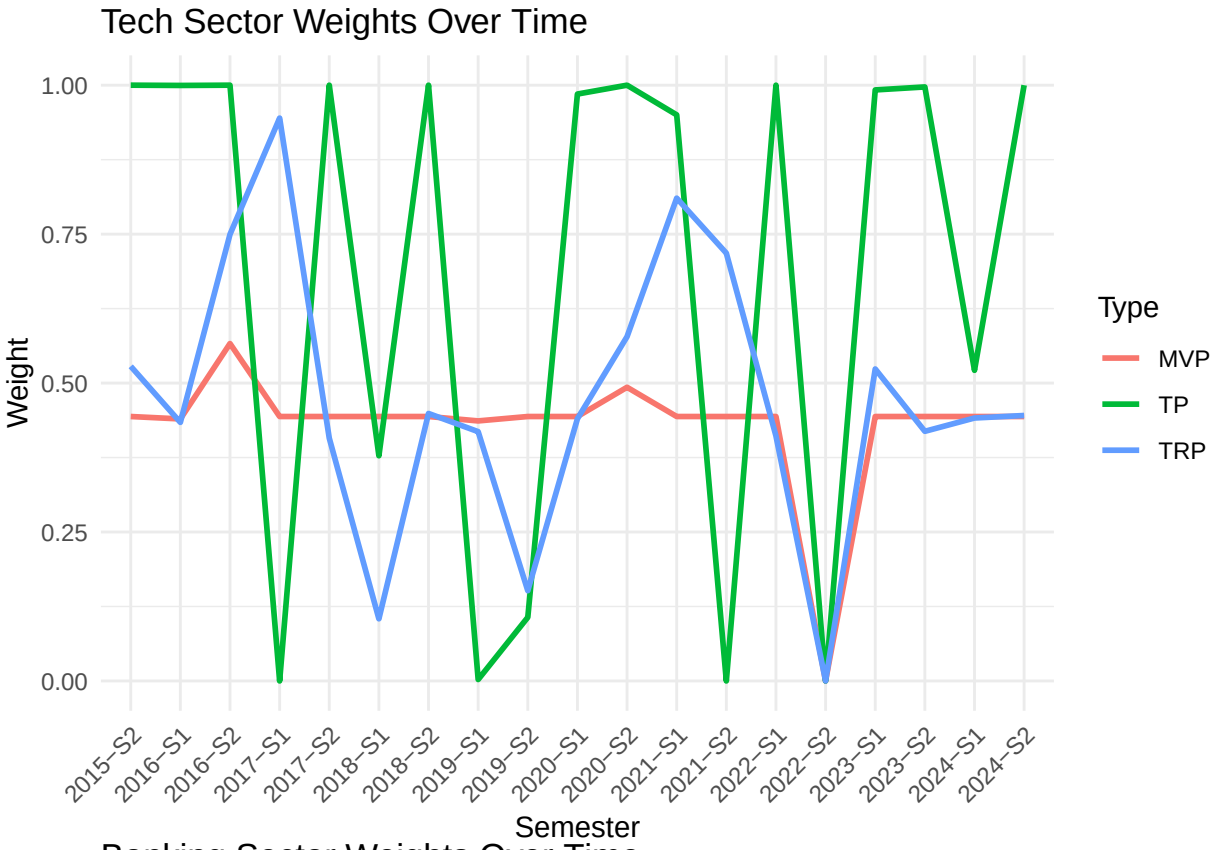
Note that in this loop, each time the returns of the previous semester are used to calculate the returns of the current semester. We collect the results in a table, and begin by plotting the Sharpe ratios of the 3 portfolios.

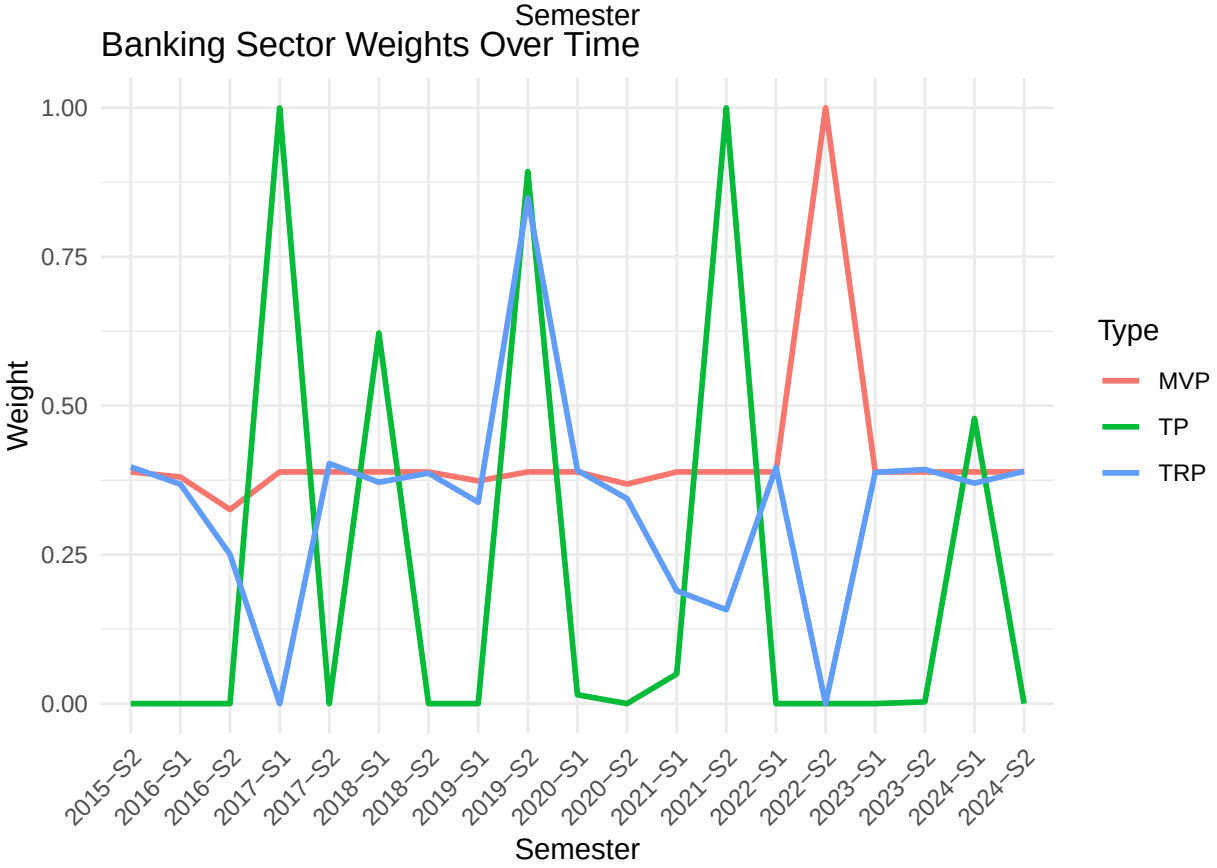
Then, taking the calculated weights, separate them by Tech and Bank sector, aggregate them respectively, and plot.

iii

```
## Warning in (function (..., deparse.level = 1) : number of rows of result is not
## a multiple of vector length (arg 1)
```




Tech Sector Weights Over Time
Weight
1.00
0.75
0.50
0.25
0.00
2015–S2
2016–S1
2016–S2
2017–S1
2017–S2
2018–S1
2018–S2
2019–S1
2019–S2
2020–S1
2020–S2
2021–S1
2021–S2
2022–S1
2022–S2
2023–S1
2023–S2
2024–S1
2024–S2
Semester
Type
MVP
TP
TRP


Banking Sector Weights Over Time
Weight
1.00
0.75
0.50
0.25
0.00
2015–S2
2016–S1
2016–S2
2017–S1
2017–S2
2018–S1
2018–S2
2019–S1
2019–S2
2020–S1
2020–S2
2021–S1
2021–S2
2022–S1
2022–S2
2023–S1
2023–S2
2024–S1
2024–S2
Semester
Type
MVP
TP
TRP

iv

From the Sharpe ratios plot, as expected the tangency portfolio has the highest Sharpe ratio out of the 3 for the entire time period. For some semesters like 2018-S1, the tangency portfolio has the same Sharpe Ratio as the minimum variance portfolio. For the 2nd semester of 2022, the portfolio with an annual target return of 8.5% was not able to be achieved.

Comparing the plots of the tech sector weights and bank sector weights, they show almost a inverse/reverse pattern. The tech sector weights shows sharp downward fluctuations, while the bank sector weights show sharp upward fluctuations. The semesters where tech sector weights spike upwards, the bank sector weights spike downwards. (for example 2022-S2) This is reasonable, as the tech and bank sectors have very different stock patterns.

Problem 1 Part b

According to the CAPM model, are the three portfolios constructed in Part (a) correctly priced for each semester. Consider the SPY ETF as a representative of the market portfolio. Organize the results in a Table and comment on them. Hint: For this problem, you use the weights obtained from Part (a) and the daily data for the components of the portfolio and the SPY for each semester in the regression model.

i

SPY prices are retrieved from the Homework 1 dataset containing the stocks and ETFs.

Daily log returns are calculated using daily prices the from the combined stock data. Log return formula: $r_t = log(1 + R_t) = log(\frac{P_t}{P_{t-1}})$

Daily log returns are then converted to semester log returns by averaging the daily log returns across each semester, and multiplying by 126.

Excess returns formula: Excess returns = Portfolio value - risk free return

CAPM test for mispricing: $H_0 : \alpha = 0$ versus $H_a : \alpha \neq 0$

ii

Extract the SPY stock prices from the stock/ETF dataset, and calculate daily log returns. Then, create a Semester column, average the daily log returns and group by semester. Multiply by 126 to get the SPY returns on semester scale.

Separately, retrieve daily prices from the combined dataset and add a column called semester. This column maps the corresponding semester label to each daily price. (For example, all prices with dates Jan 2015-Jun 2015 will get the label 2015-S1)

Retrieve the weights of the 3 portfolios calculated in part 1a, and join them with the daily prices on the Semester column.

Now, for each of the 3 portfolios, multiply the daily price by the corresponding semester weights to get the portfolio values.

Then combine the values of the 3 portfolios together to get combined dataframe of daily portfolio values. Use the Semester column created earlier to aggregate the daily portfolio values into semester portfolio values.

Using the semester portfolio values, subtract the semester risk free rates (calculated in part 1a) to get semester excess returns.

Lastly, run the CAPM model on the semester excess returns to test if the portfolios constructed in part 1a were mispriced ($\alpha = 0$ or $\alpha \neq 0$).

iii

```
## Warning: There was 1 warning in ‘summarise()‘.
## i In argument: ‘alpha = coef(lm(Excess_Return ~ Semester_Return, data =
##   cur_data()))[1]‘.
## i In group 1: ‘Type = "MVP"‘.
## Caused by warning:
## ! ‘cur_data()‘ was deprecated in dplyr 1.1.0.
## i Please use ‘pick()‘ instead.
```

```
## # A tibble: 3 x 4
##   Type    alpha   beta p_value_alpha
##   <chr>   <dbl>  <dbl>         <dbl>
## 1 MVP    0.0626 -0.939         0.671
## 2 TP     0.145  -3.02          0.437
## 3 TRP   -0.0159  0.722         0.921
```

iv

Based on the CAPM model results table, the α values have p-values that are much larger than 0.05. This suggests that there is no evidence of mispricing based on the weights from part 1a.

Problem 2 Part a

Based on the stocks considered in Problem 1, an investor would like to construct different portfolios and rebalance them every month. In the construction of each portfolio, the investor would use a predictive model for calculating the expected log-return and risk for each stock for the upcoming month. For example for the MSFT stock, the investor has monthly return data from Jan 2015 to Dec 2023. The investor would use these data to predict the expected return for Jan 2024 and so forth. Similarly for the risk free rate. For the risk, use the historical average risk based on the Jan 2015 to Dec 2023 period. Plot the Sharpe ratios over time (for each month) of the three portfolios. Also, plot the aggregate weight of the technology sector and the banking sector over time (for each semester) for the three portfolios. Comment on the results.

i

Continue using the combined stock data, risk free rate data read in from problem 1.

Daily log returns are calculated using daily prices the from the combined stock data. Log returns formula: $r_t = log(1 + R_t) = log(\frac{P_t}{P_{t-1}})$

Daily log returns are then converted to monthly log returns by aggregating daily log returns across each month.

Daily annualized risk free rates are converted to monthly risk free rates by taking the mean across months, and diving by 12.

The ARIMA model is used via the functions arima() and auto.arim().

ii

Instead of daily log returns as calculated in problem 1, now aggregate them by month to calculate monthly log returns.

Filter the monthly log returns to the desired historical period (Jan 2015 - Dec 2023).

Similarly, aggregate the daily risk free rate data to monthly risk free rate data, and filter to the same historical period (Jan 2015 - Dec 2023).

Use the historical monthly log returns to fit an ARIMA model and forecast the expected returns for 12 months of 2024 (Jan 2024 - Dec 2024).

Using the forecasted expected returns from ARIMA, calculate covariance matrix of the monthly returns.

Similarly, use the historical risk free rate data to fit an ARIMA model and forecast the risk free rates for 12 months (Jan 2024 - Dec 2024)

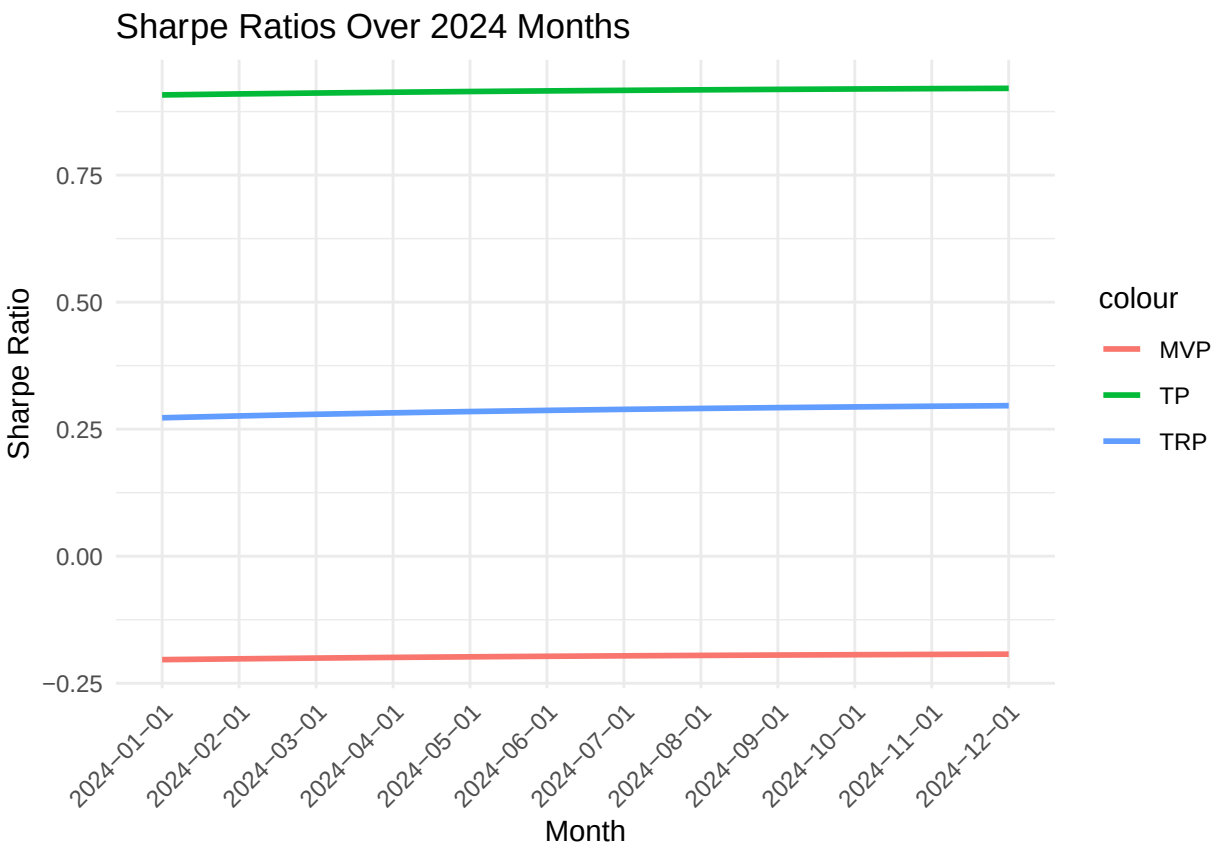
Now, again utilize the portfolio construction code (from the homeworks) to loop through each month from Jan 2024 to Dec 2024 (12 months).

In each iteration, extract the weights, calculate the Sharpe ratio for each of the 3 portfolios (minimum variance, tangency, target return (if it exists).

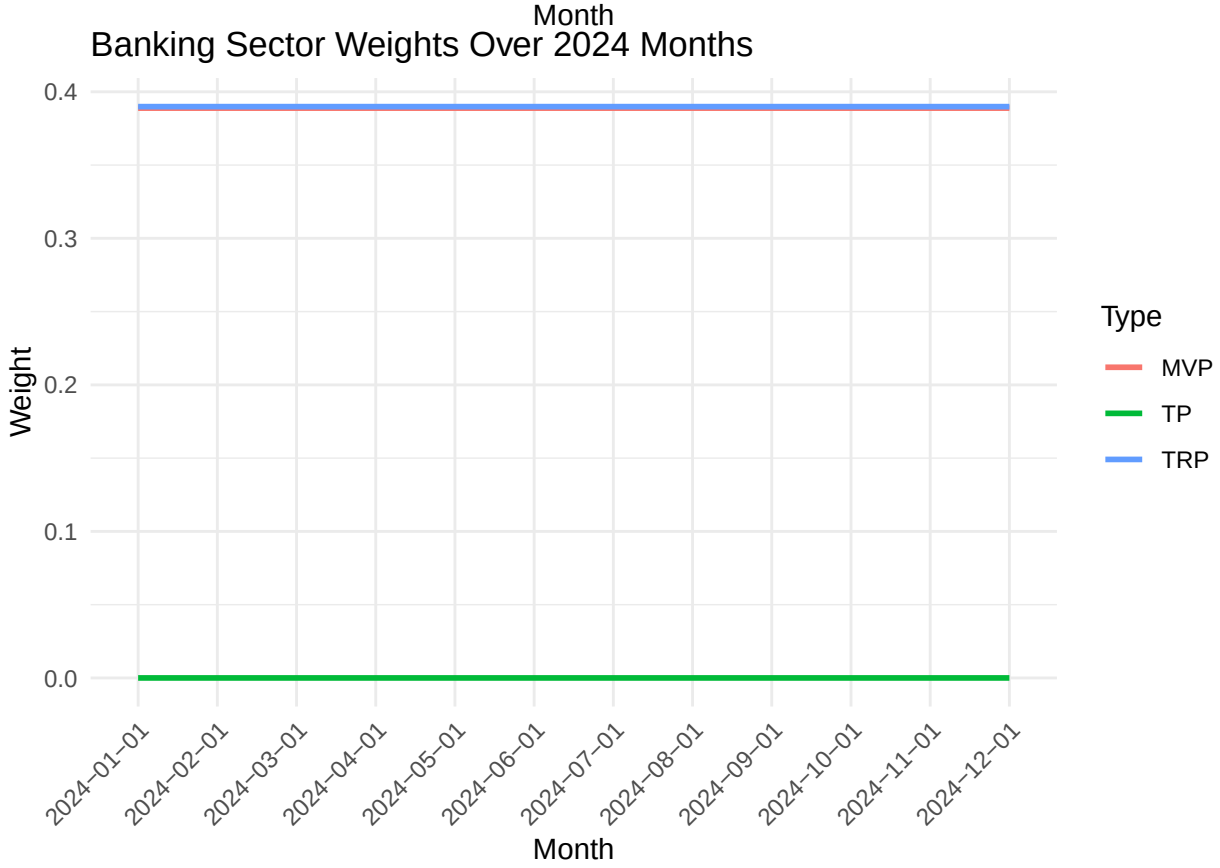
Note that in this loop, each time the forecasted monthly returns from the ARIMA model are used to calculate the returns of the current semester.

Collect the results in a table, and begin by plotting the Sharpe ratios of the 3 portfolios. Then, taking the calculated weights, separate them by Tech and Bank sector, aggregate them respectively, and plot.

iii




Tech Sector Weights Over 2024 Months
1.0
0.8
0.6
Weight
2024–01–01
2024–02–01
2024–03–01
2024–04–01
2024–05–01
2024–06–01
2024–07–01
2024–08–01
2024–09–01
2024–10–01
2024–11–01
2024–12–01
Month
Type
MVP
TP
TRP


Banking Sector Weights Over 2024 Months
0.4
0.3
0.2
0.1
0.0
Weight
2024–01–01
2024–02–01
2024–03–01
2024–04–01
2024–05–01
2024–06–01
2024–07–01
2024–08–01
2024–09–01
2024–10–01
2024–11–01
2024–12–01
Month
Type
MVP
TP
TRP

iv

The Sharpe ratio plots of the 3 portfolios under this method (method A) show very constant/slightly increasing trends across the months - this is somewhat expected because we made predictions on all 12 months of 2024 at once. An alternative approach might be to use a rolling mean - for example adding on one month at a time to predict the next. (using historical data Jan 2015- Dec 2023 to predict Jan 2024, then adding Jan 2024 to historical data to predict Feb 2024)

Separating the weight plots by tech and bank sector again, note that the weights of the 3 portfolios show similarly constant trends. However, an important observation is that the weights of the distinct sectors again show an inverse/reverse pattern. Where the tangency portfolio has the highest weight values in the tech sector, it has the lowest weight values in the bank sector. For the minimum variance and target return portfolio weights, this similar inverse pattern is present.

Problem 2 Part b

Would the investor be better off instead of predicting the expected return for the next month, use the average return for the previous month in the construction of the three portfolios. For example, instead of predicting the expected return for MSFT for the month of Jan 2024 and construct the Jan 2024 portfolios, use instead the monthly return of December 2023 and construct the portfolios. Hint: Based on the closing prices of the stocks at the end of the month (end of Jan, end of Feb and so forth), calculate the value of the three portfolios based on the weights assigned to each stock, according to the mechanism used in Part (a) and the new mechanism in Part (b).

i

Instead of daily log returns as calculated in problem 1, now aggregate them by month to calculate monthly log returns.

Filter the monthly log returns to the desired period (Dec 2023 - Nov 2024).

Similarly, aggregate the daily risk free rate data to monthly risk free rate data, and filter to the desired period (Dec 2023 - Nov 2024)

Retrieve the daily prices read in for problem 1, group by month, and extract the price at the end of each month to get monthly prices.

Use the formula: portfolio values = monthly weights * monthly prices

ii

Using the monthly log returns (same as from part 2a, calculated by aggregating daily log returns)

Instead, filter monthly log returns to have a range from Dec 2023 to Nov 2024. This "lag" of one month is so the prior month can be used to predict the following month. For example, Dec 2023 will be used for Jan 2024, and we will end with Nov 2024 to predict Dec 2024.

Similarly, get the monthly risk free data (same as from part 2a) and filter to the range Dec 2023 to Nov 2024.

For the portfolio construction, set up a start date of Jan 2024 and end date of Dec 2024. This is the range of dates to calculate returns for.

Utilize the portfolio code (from the homeworks) to loop through each month from Jan 2024 to Dec 2024 (12 months) In each iteration, extract the weights, calculate the Sharpe ratio for each of the 3 portfolios

(minimum variance, tangency, target return (if it exists). Note that in this loop, each time the previous month's returns are used to calculate the returns of the current month.

Collect the results in a table, and begin by plotting the Sharpe ratios of the 3 portfolios. Then, taking the calculated weights, separate them by Tech and Bank sector, aggregate them respectively, and plot.

The last step of this part is to collect the portfolio values, multiply weights in part 2a and 2b by the monthly prices, then compare values of portfolios.

The objective of part 2a was to use historical data in a predictive model to forecast expected returns for the 12 months of 2024.

The objective of the method in part 2b is to use the previous month's data to calculate next month return. For example, use Dec 2023 to forecast Jan 2024, use Jan 2024 to build Feb 2024, etc.

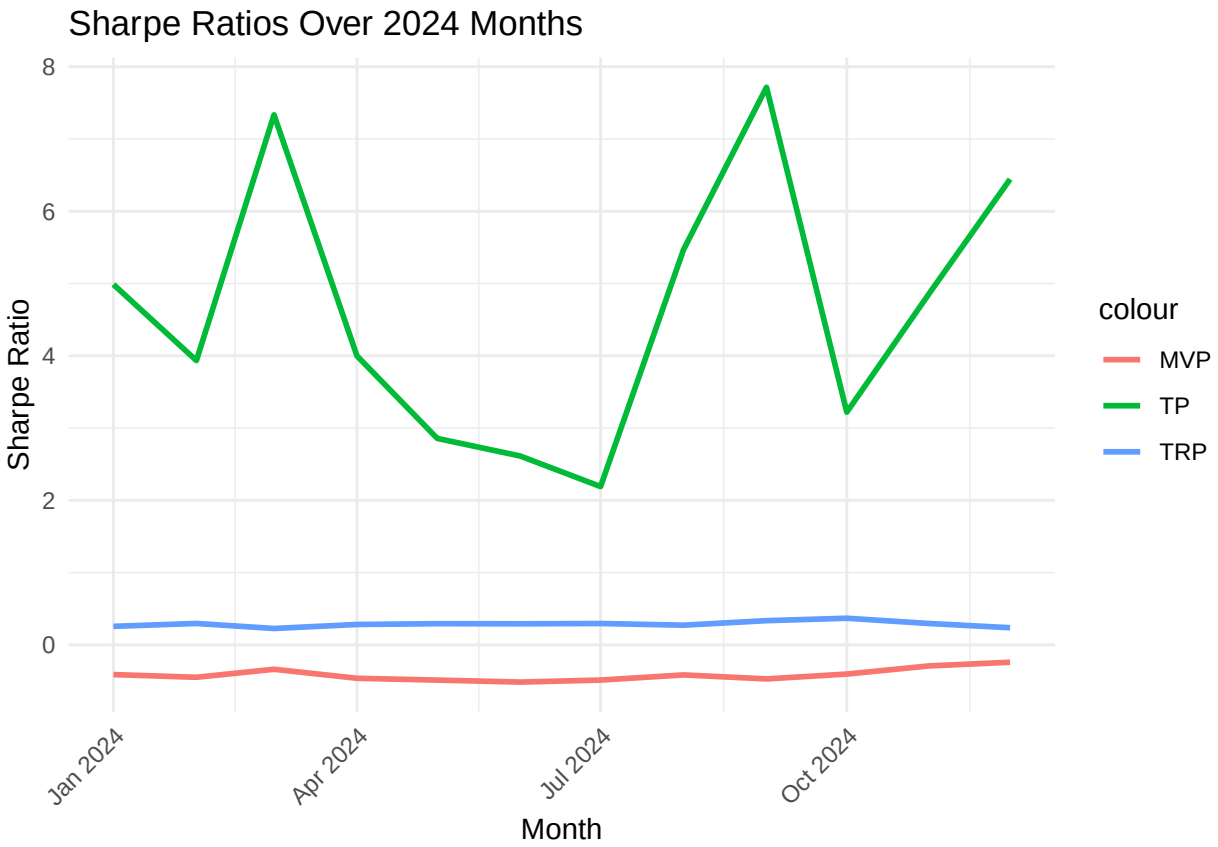
To calculate the portfolio values, begin by retrieving the daily prices from the data combined dataset. Then aggregate to get monthly prices, and filter for only the desired months (Jan 2024-Dec 2024).

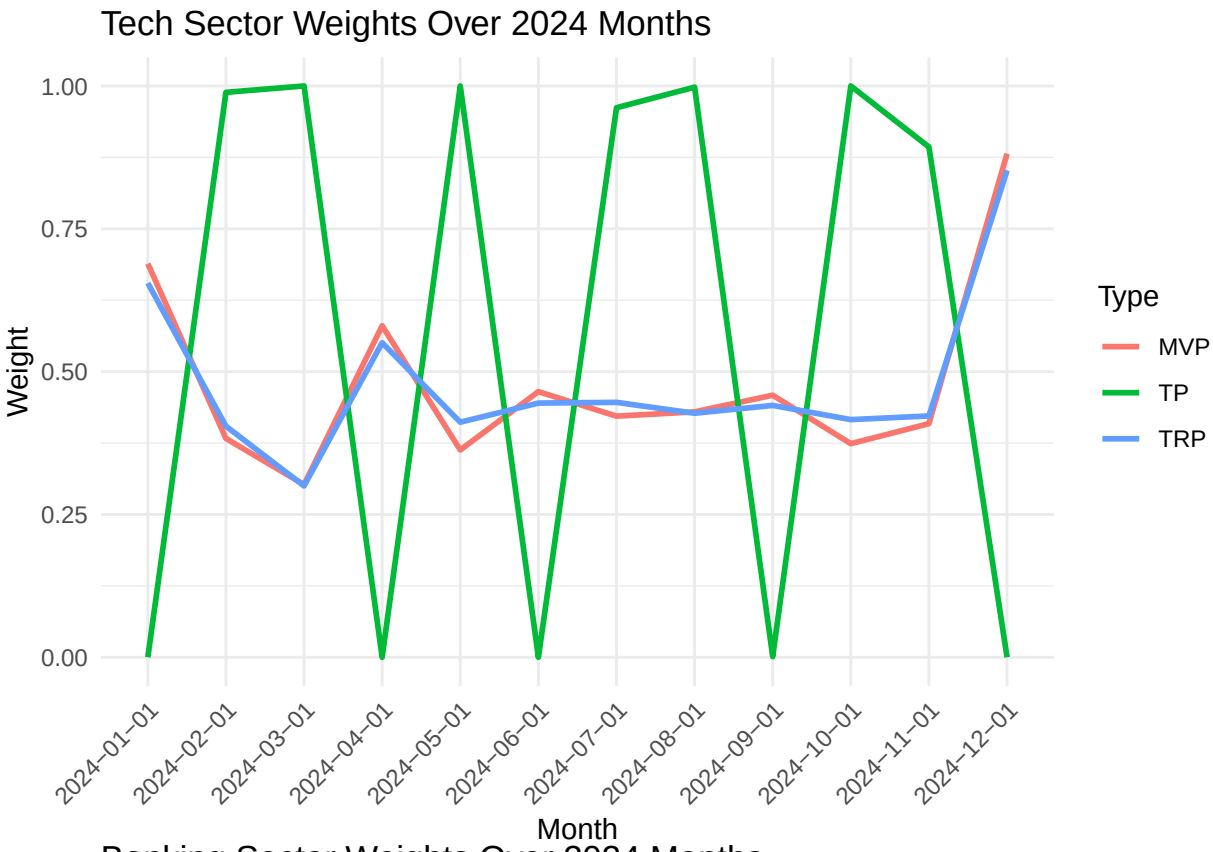
Next, pull the portfolio weights from method A (in part 2a) and method B (in the first part of part 2b). Join and multiply the each set of weights (method A, method B) with the monthly prices that were just calculated on the Month column.

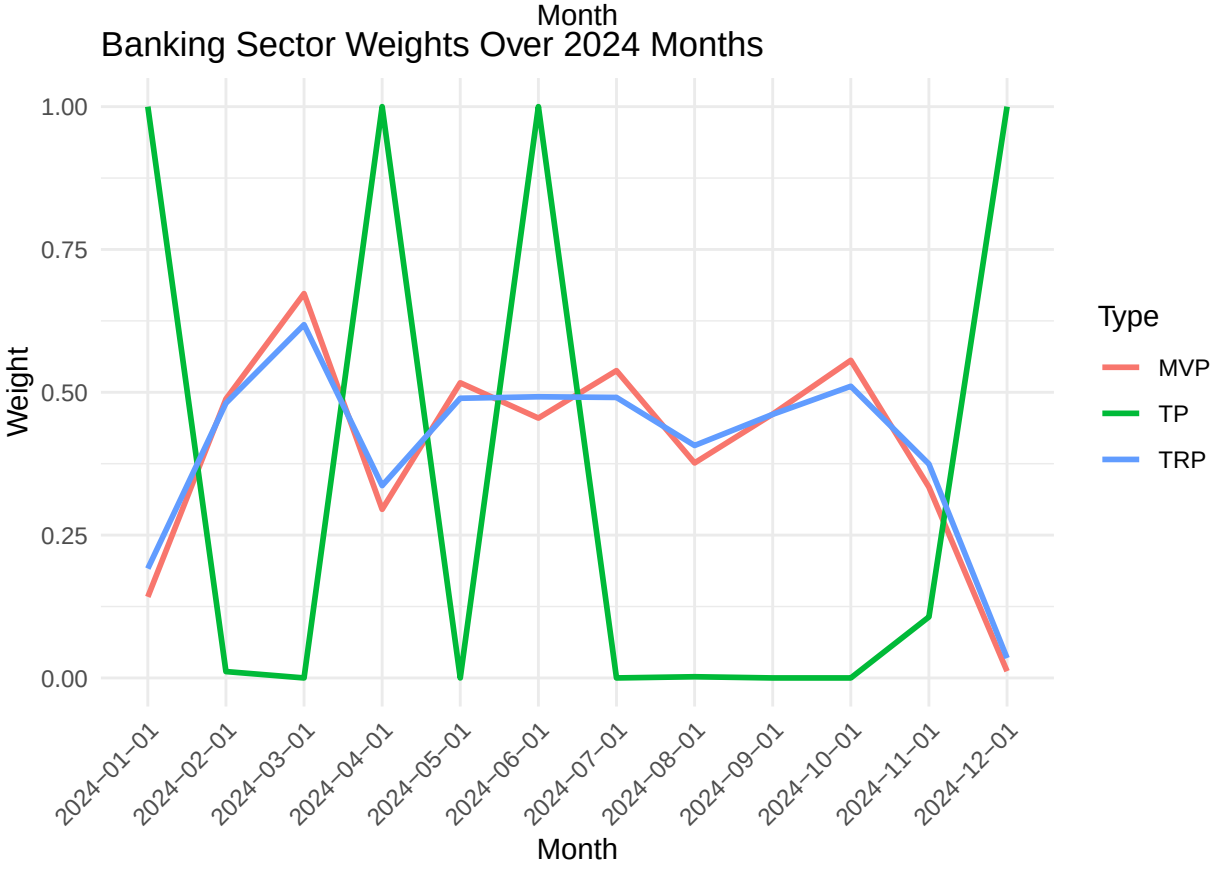
There should be 6 sets of portfolio values: each portfolio under method A (MVP, TP, TRP), each portfolio under method B (MVP, TP, TRP).

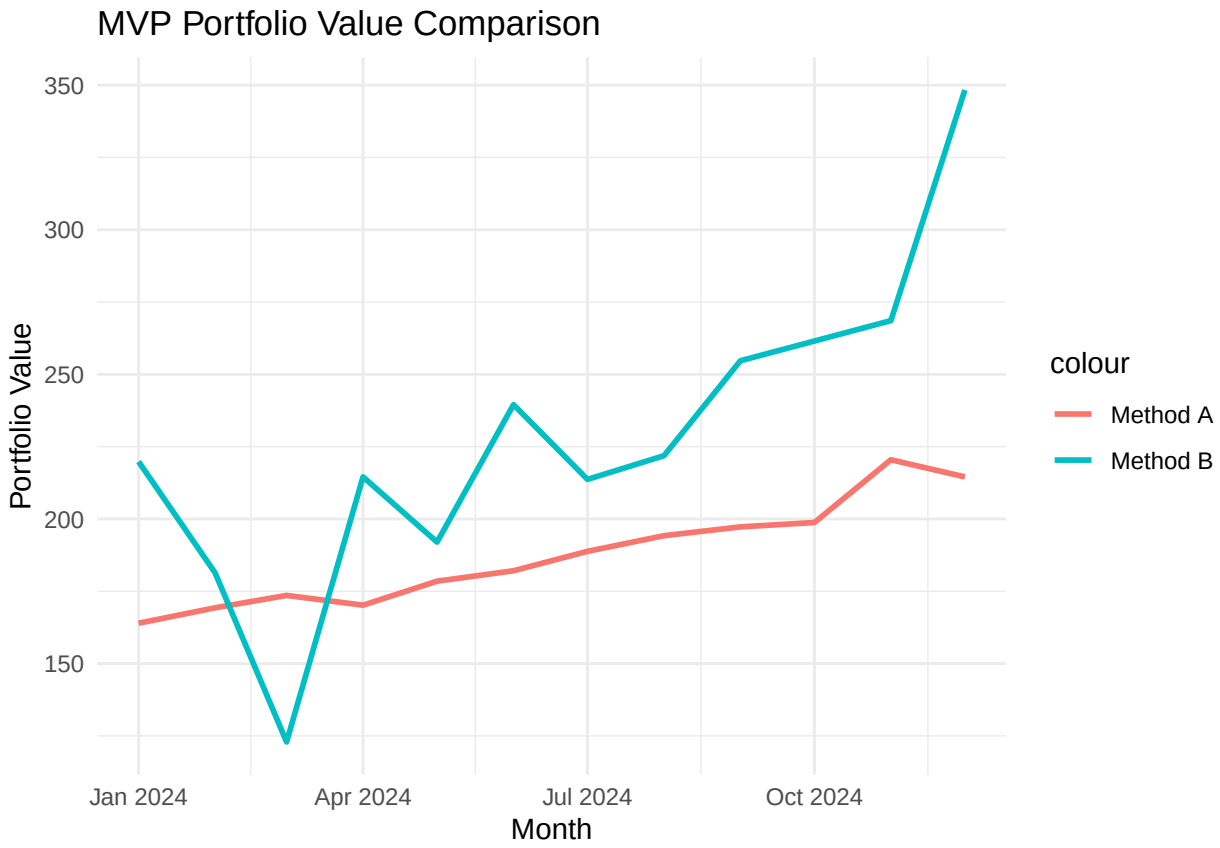
Lastly, plot a comparison of the portfolio values over the 12 months of 2024.

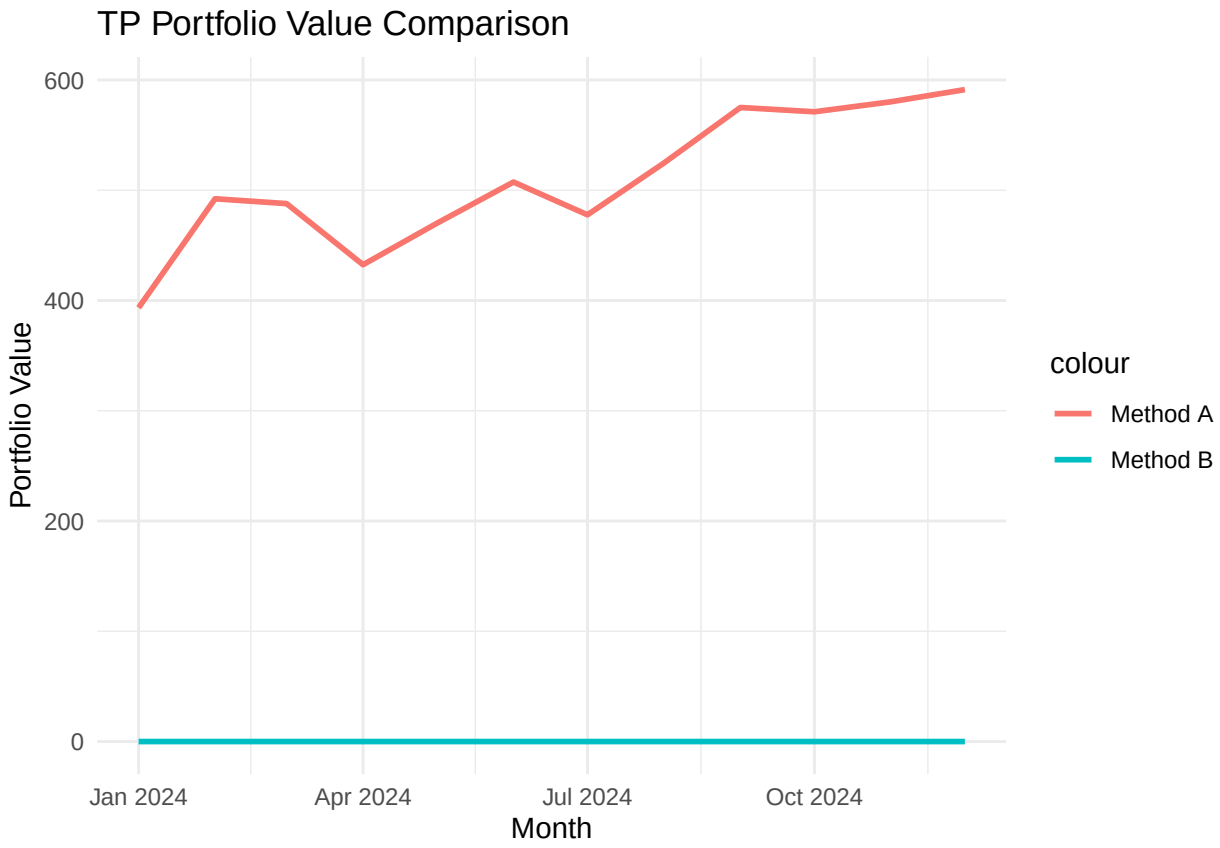
iii

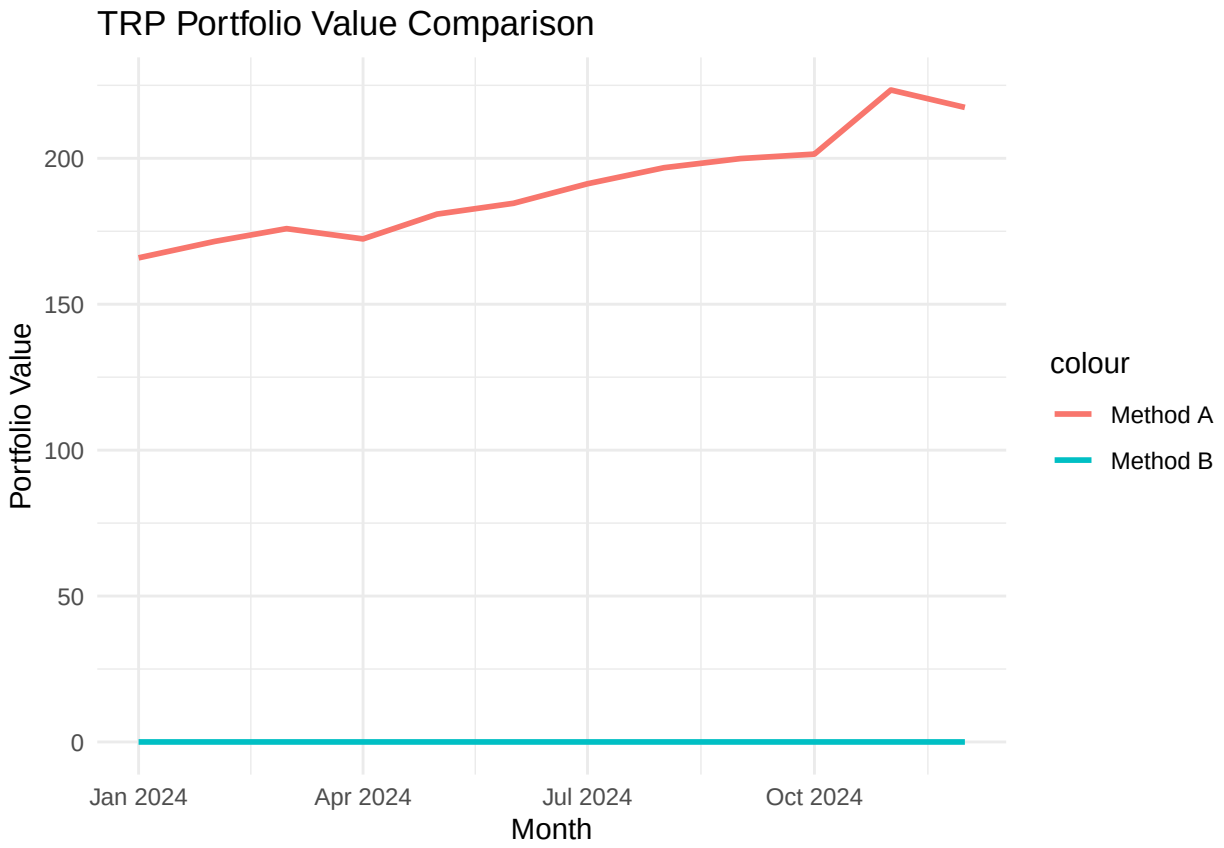



Tech Sector Weights Over 2024 Months
Weight
1.00
0.75
0.50
0.25
0.00
2024-01-01
2024-02-01
2024-03-01
2024-04-01
2024-05-01
2024-06-01
2024-07-01
2024-08-01
2024-09-01
2024-10-01
2024-11-01
2024-12-01
Month
Type
MVP
TP
TRP


Banking Sector Weights Over 2024 Months
Weight
1.00
0.75
0.50
0.25
0.00
2024-01-01
2024-02-01
2024-03-01
2024-04-01
2024-05-01
2024-06-01
2024-07-01
2024-08-01
2024-09-01
2024-10-01
2024-11-01
2024-12-01
Month
Type
MVP
TP
TRP


MVP Portfolio Value Comparison
350
300
250
200
150
Portfolio Value
Jan 2024
Apr 2024
Jul 2024
Oct 2024
Month
colour
Method A
Method B


TP Portfolio Value Comparison
600
400
200
0
Portfolio Value
Jan 2024
Apr 2024
Jul 2024
Oct 2024
Month
colour
Method A
Method B



iv

The Sharpe ratio plot of the portfolios under this method (method B) show slightly different patterns from method A. The ratio of the tangency portfolio fluctuates greatly across months, whereas the ratios of the minimum variance and target return portfolios stay relatively constant over months.

This variation pattern is reflected in the weights plots - for both tech and bank sectors, the tangency weight values spike upwards and downwards greatly in an inverse/reverse pattern. The weights of the minimum variance portfolios and target return portfolios show similar inverse patterns between sectors, but fluctuate less across months.

Lastly, looking at the porfolio value comparison plots of method A (problem 2a) and method B (problem 2b). Method A overall produces more constant portfolio values across all 3 portfolios. Method B produces more variation in minimum variance portfolio values, but produces near constant values for the tangency and target return portfolios.

Method A might be more suitable for long term forecasts if the portfolio values are known to show steady/constant trends. It would also be more effective if one were to implement a rolling mean forecasting strategy (appending each month onto the historical data as one predicts the next month).

In contrast, method B has a simpler implementation, and captures the variation from month to month better. This strategy is suitable if the portfolio values one is trying to forecast does not show a steady pattern across long periods of time.

Problem 3

An investor plans to invest $ 100 million on each of the three portfolios constructed at the beginning of each month in the period starting from January 2024 and ending in the December 2024 in Part (a) of Problem 2. The investor is interested in calculating the VaR and a 95% confidence interval for a one month horizon, assuming that the returns of the assets in the three portfolios can be modeled by a multivariate t-distribution based on monthly data from 2015-2023. Assume confidence level for VaR to be 0.99. Report the VaR of the three portfolios over time (for each month), together with their confidence intervals and comment on the results.

i

Daily log returns are then converted to monthly log returns by aggregating daily log returns across each month. (calculated the same way as in problem 2)

Daily annualized risk free rates are converted to monthly risk free rates by taking the mean across months, and diving by 12. (calculated the same way as in problem 2)

The parametric VaR code was retrieved from Lecture 6, slides 33,36

ii

Begin by filtering monthly log returns to the desired historical date range (2015-2023)

Then initialize variables, including the range of 12 months of 2024 that VaR and confidence intervals will be calculated for.

Using the parametric VaR code (from the lecture notes), create 2 loops. The first loop iterates over the set of the portfolio weights (MVP, TP, TRP), the second iterates over each of the 12 months.

Within the two loops, calculate VaR with 99% confidence. This is followed by using bootstrap to calculate a 95% confidence interval of the VaR for a one month horizon.

Then store each of the VaR values and confidence intervals for the 12 months in a table to display.

iii

```
##         Month Portfolio      VaR CI_95_lower CI_95_upper
## 1  2024-01-01       MVP 13134433     9282108    19785126
## 2  2024-02-01       MVP 13134433     9363394    19951131
## 3  2024-03-01       MVP 13134433     8861771    19544024
## 4  2024-04-01       MVP 13134433     8986857    21218293
## 5  2024-05-01       MVP 13134433     9649628    19347143
## 6  2024-06-01       MVP 13134433     8754270    20096599
## 7  2024-07-01       MVP 13134433     9145021    20789395
## 8  2024-08-01       MVP 13134433     9169953    21829339
## 9  2024-09-01       MVP 13134433     9027488    18857910
## 10 2024-10-01       MVP 13134433     9031252    19769279
## 11 2024-11-01       MVP 13134433     8995688    21730701
## 12 2024-12-01       MVP 13134433     8508725    19537584
## 13 2024-01-01        TP 24391554    15610770    36586840
## 14 2024-02-01        TP 24391554    16865188    38440590
## 15 2024-03-01        TP 24391554    16603408    38200203
## 16 2024-04-01        TP 24391554    16491024    35507935
```

```
## 17 2024-05-01        TP 24391554    16926746    34737469
## 18 2024-06-01        TP 24391554    17226565    39653151
## 19 2024-07-01        TP 24391554    16189372    35952178
## 20 2024-08-01        TP 24391554    16683143    37609955
## 21 2024-09-01        TP 24391554    16453250    36323186
## 22 2024-10-01        TP 24391554    17086188    34686266
## 23 2024-11-01        TP 24391554    16755385    37340723
## 24 2024-12-01        TP 24391554    16476972    35384900
## 25 2024-01-01       TRP 13125115     8978581    19029212
## 26 2024-02-01       TRP 13125115     9238605    19668411
## 27 2024-03-01       TRP 13125115     9261959    18870454
## 28 2024-04-01       TRP 13125115     9155698    18880899
## 29 2024-05-01       TRP 13125115     9351942    19359323
## 30 2024-06-01       TRP 13125115     9148740    20801610
## 31 2024-07-01       TRP 13125115     8893160    20364245
## 32 2024-08-01       TRP 13125115     8877108    19873417
## 33 2024-09-01       TRP 13125115     9443737    21244882
## 34 2024-10-01       TRP 13125115     9191080    19386333
## 35 2024-11-01       TRP 13125115     8857319    18841772
## 36 2024-12-01       TRP 13125115     9143325    19038466
```

iv

The VaR and respective confidence intervals for each of the 3 portfolios across 12 months are shown in the table.

The VaR for the minimum variance and target return portfolios are similar, whereas the VaR for the tangency portfolio is much higher. This makes sense because the tangency portfolio aims for the highest return, which usually also means higher risk is involved.

Note that the VaR values for each portfolio is nearly identical across the 12 months. This is reasonable given that the weights used to calculate the VaR were from problem 2a (method A), which produced very constant/similar expected returns across the 12 months of 2024.

Problem 4

An investor is interested in understanding how the price of a zero-coupon bond fluctuates as a function of the interest rate. To that end, the investor plans to invest in a zero-coupon bond at the beginning of each semester starting in he first semester of 2015 and ending in the second semester of 2024, whose par value is $1000 and has a 6-month horizon. The investor plans to use as a proxy for the prevailing interest rate, the average of the daily annualized interest rates available in the file DGS6MO.csv. Can you help the investor to explore the relationship between the price of a 6-month zero-coupon bond and the corresponding interest rate? Explain in detail your approach and comment on your result.

i

The risk-free data was obtained from DGS6MO.csv. Semester interest rates were calculated by aggregating daily annualized interest rates, grouping by semester and dividing by 2. (Similarly to what was done in problem 1a)

The semi annual compounding formula : $P = \frac{V}{(1+r/2)^{2T}}$ In this case, $V = 1000$, r is the average risk free rate for the given semester, and $T = 0.5$.

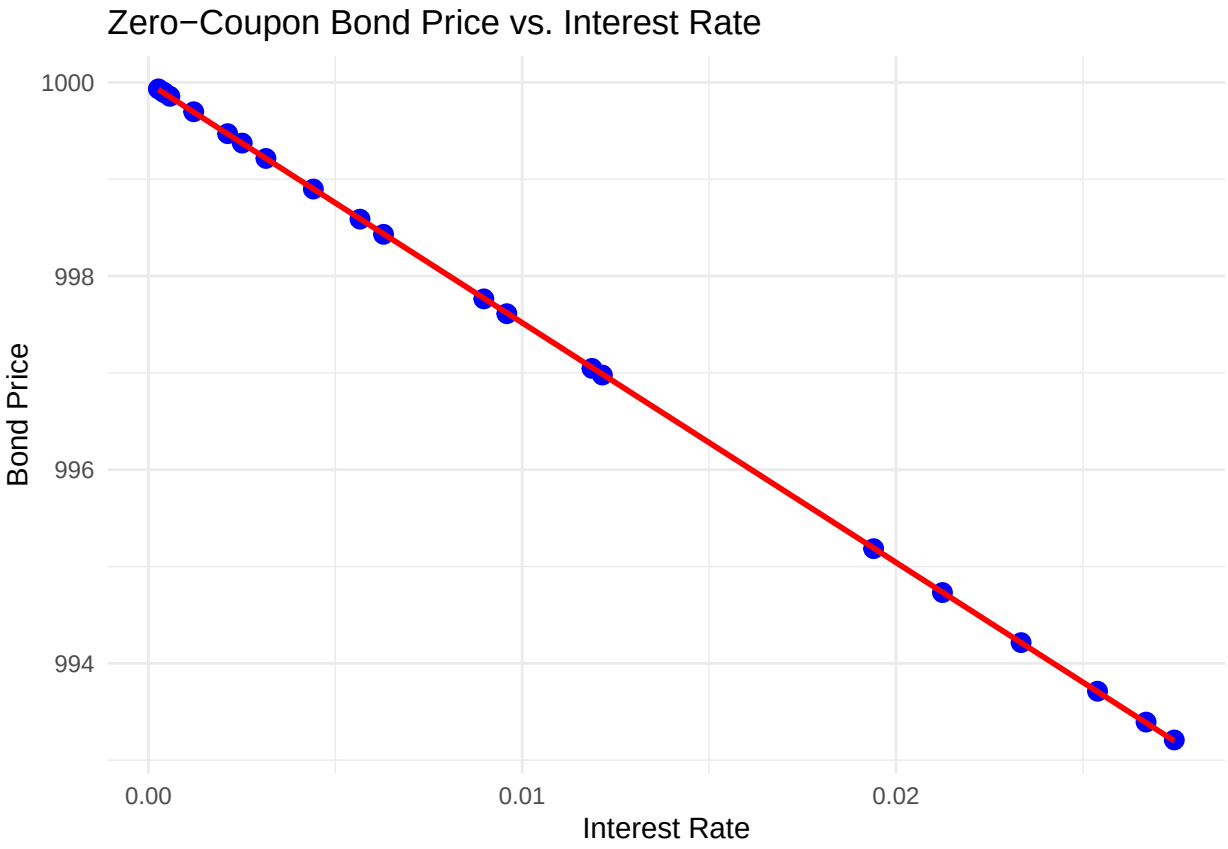
ii

Begin with the risk free data that contains the annualized rates. Define the desired date range, making sure to group by semester and scaling the daily annualized risk free rates to a semester scale. (Similarly to what was done in problem 1a)

Then use the semi-annual compounding formula to calculate the bond prices during each semester.

To explore the relationship between bond price and interest rate, first plot the bond prices against a line that represents interest rate. In addition, calculate the correlation between bond price and interest rate.

iii




```
## [1] "Correlation between interest rates and bond prices: -0.999996982487639"
```

iv

In the plot of bond price and interest rate, there is a very clear 45 degree downward sloping line. This suggests bond price and interest rate have an inverse relationship. To confirm this, correlation can be calculated between bond price and interest rate. As expected, the correlation is -0.9999, which means the two are almost perfectly inversely related. This aligns with the concept that when interest rates go up, generally bond prices go down, and vice versa.

Appendix

```r
# necessary packages
library(dplyr)
library(ggplot2)
library(tidyr)
library(quadprog)
library(lmtest)
library(zoo)
library(MASS)
library(mvtnorm)
library(lubridate)
library(stats)
library(tibble)
library(stringr)
library(forecast)
library(fitHeavyTail)
```

Section 0. Read in data

```r
# stock data
folder_path <- "C:/Users/cupca/Desktop/UCLA/Winter2025/STAT417/hw1_data/"
all_files <- list.files(folder_path)
filtered_files <- all_files[!grepl("-", all_files)]
filtered_files <- all_files[-1]
#print(filtered_files)

# clean data
datasets <- lapply(filtered_files, function(x){
  curr_data <- read.table(paste0(folder_path,x), sep=",", h=1)
  curr_data$Date <- as.Date(curr_data$Date, "%m/%d/%Y")
  curr_data <- curr_data[order(curr_data$Date),]
  curr_data$Close.Last <- as.numeric(gsub("\\$", "", curr_data$Close.Last))
  price <- curr_data$Close.Last
  names(price) <- curr_data$Date
  return(price)
})

file_names <- gsub("\\.csv$", "", filtered_files)
names(datasets) <- file_names

# financial stock data
folder_path2 <- "C:/Users/cupca/Desktop/UCLA/Winter2025/STAT417/hw2_data/"
all_files2 <- list.files(folder_path2)
filtered_files2 <- all_files2[!grepl("-", all_files2)]
#print(filtered_files2)

# clean data
datasets2 <- lapply(filtered_files2, function(x){
  curr_data <- read.table(paste0(folder_path2,x), sep=",", h=1)
```

```
  curr_data$Date <- as.Date(curr_data$Date, "%m/%d/%Y")
  curr_data <- curr_data[order(curr_data$Date),]
  curr_data$Close.Last <- as.numeric(gsub("\\$", "", curr_data$Close.Last))
  price <- curr_data$Close.Last
  names(price) <- curr_data$Date
  return(price)
})

file_names2 <- gsub("\\.csv$", "", filtered_files2)
names(datasets2) <- file_names2
financial_stocks <- datasets2

# add financial stocks to the org stocks
datasets_combined <- c(datasets, financial_stocks)

# label combined dataset
names(datasets_combined) <- c(names(datasets), names(datasets2))

# selected stocks
selected_stocks <- c("Amazon", "Apple", "Cisco", "Meta", "Microsoft", "Netflix",
                     "Starbucks", "AMEX", "BAC", "C", "GS", "MS", "WFC")

# convert list to df and filter only selected stocks
datasets_combined_df <- do.call(cbind, datasets_combined[selected_stocks])

# get dates for each stock
all_dates <- lapply(datasets_combined[selected_stocks], function(stock) as.Date(names(stock)))

# find the common trading dates across all stocks
common_dates <- Reduce(intersect, all_dates)

# filter stocks to only use the common dates
aligned_prices <- lapply(datasets_combined[selected_stocks], function(stock) {
  stock[as.Date(names(stock)) %in% as.Date(common_dates)]
})

# convert to df and ensure the same date index
daily_prices <- as.data.frame(aligned_prices)
rownames(daily_prices) <- as.character(as.Date(common_dates))

#head(daily_prices)
```

```
# risk-free rate data
risk_free_path <- "C:/Users/cupca/Desktop/UCLA/Winter2025/STAT417/DGS6MO.csv"
risk_free_data <- read.csv(risk_free_path, sep=",", skip=1, stringsAsFactors=FALSE)
colnames(risk_free_data) <- c("Date", "Yield")
risk_free_data$Date <- as.Date(risk_free_data$Date, format="%Y-%m-%d") # convert to Date
risk_free_data$Rate <- as.numeric(risk_free_data$Yield) / 100 # convert percentage to numeric
risk_free_data <- na.omit(risk_free_data) # remove NAs

#head(risk_free_data)
```

Problem 1

Part a

i

```
# daily log returns
returns <- lapply(datasets_combined[selected_stocks], function(prices) {
  log(prices[-1] / head(prices, -1))
})

# convert list to df
returns_df <- do.call(cbind, returns)
colnames(returns_df) <- names(datasets_combined[selected_stocks]) # each column is a stock name
returns_df <- na.omit(returns_df) # remove NAs
returns_df <- as.data.frame(returns_df)
#head(returns_df)
```

```
# scale to semester covariance matrix
cov_matrix_daily <- cov(returns_df, use = "complete.obs")
cov_matrix_scaled <- cov_matrix_daily * 126

# turn rownames into Date column
returns_df <- returns_df %>%
  tibble::rownames_to_column(var = "Date") %>%
  mutate(Date = as.Date(Date))

# add semester column
semester_returns <- returns_df %>%
  mutate(Semester = paste0(year(Date), "-", ifelse(month(Date) <= 6, "S1", "S2")))

# average daily return per semester
semester_returns <- semester_returns %>%
  group_by(Semester) %>%
  summarise(across(-Date, mean, na.rm = TRUE))

# scale to get semester returns
semester_returns_scaled <- semester_returns %>%
  mutate(across(-Semester, ~ .x * 126))

#head(semester_returns_scaled)
```

```
risk_free_data$Date <- as.Date(risk_free_data$Date, format="%Y-%m-%d")

# add semester column
risk_free_data <- risk_free_data %>%
  mutate(Semester = paste0(year(Date), "-S", ifelse(month(Date) <= 6, 1, 2)))

# avg risk free rate for each semester, scale to get semester risk free rate
risk_free_semester <- risk_free_data %>%
  group_by(Semester) %>%
  summarise(Avg_Rate = mean(Rate, na.rm = TRUE) / 2)
```

```r
#head(risk_free_semester)
```

ii

```r
sharpe_table <- data.frame(
  Semester = integer(),
  Sharpe_MVP = numeric(),
  Sharpe_TP = numeric(),
  Sharpe_TRP = numeric(),
  stringsAsFactors = FALSE
)

weights_table <- data.frame(
  Semester = integer(),
  stringsAsFactors = FALSE
)

nassets=13
stock_names <- colnames(semester_returns_scaled[,-1]) # exclude semester

# loop over semesters 2-20 which correspond to (2015 S2 - 2024 S2)
for (semester in 2:20) {
    # use historical data of previous semester
    mean_vect = colMeans(semester_returns_scaled[semester - 1, -1])
    cov_mat = cov_matrix_scaled
    cov_regularized <- cov_mat + diag(1e-3, ncol(cov_mat))
    sd_vect = sqrt(diag(cov_mat))

    Amat = cbind(rep(1, nassets), mean_vect, diag(1, nrow = nassets))

    min_return <- min(semester_returns_scaled[semester - 1, -1], na.rm = TRUE)
    max_return <- max(semester_returns_scaled[semester - 1, -1], na.rm = TRUE)
    muP <- seq(min_return + 1e-4, max(0.09, max_return), by = 0.0025)

    mufree <- mufree <- risk_free_semester[semester - 1,]$Avg_Rate #risk_free_semester[semester,]$Avg_R

    weights_mvp = matrix(0, nrow = length(muP) , ncol = nassets)
    weights_tp = matrix(0, nrow = length(muP) , ncol = nassets)
    weights_trp = matrix(0, nrow = 1, ncol = nassets)

    # MVP
    for (i in 1:length(muP)) {
      bvec_mvp <- c(1, rep(0, nassets))
      result_mvp <- solve.QP(
        Dmat = 2 * cov_regularized,
        dvec = rep(0, nassets),
        Amat = Amat[, 1:14],
        bvec = bvec_mvp,
        meq = 1)
    }
```

```r
# TP
for (i in 1:length(muP)) {
  bvec_tp <- c(1, muP[i], rep(0, nassets))
  #bvec_tp <- c(1, mufree, rep(0, nassets))
  if (all(mean_vect < mufree)) {
      result_tp <- NA
      weights_tp <- rep(NA, nassets)
      portfolio_sd_tp <- NA
  } else {
      result_tp <- tryCatch(
          solve.QP(
              Dmat = 2 * cov_regularized,
              dvec = rep(0, nassets),
              Amat = Amat,
              bvec = bvec_tp,
              meq = 2
          ),
          error = function(e) NA)
  }
}

# TRP
bvec_trp <- c(1, 0.0425, rep(0, nassets))
result_trp <- tryCatch({
  solve.QP(
    Dmat = 2 * cov_regularized,
    dvec = rep(0, nassets),
    Amat = Amat,
    bvec = bvec_trp,
    meq = 2
  )
}, error = function(e) NA)

weights_mvp <- result_mvp$solution
weights_trp <- if (!is.null(result_trp) && "solution" %in% names(result_trp))
  result_trp$solution else rep(NA, nassets)
weights_tp <- if (!is.null(result_tp) && "solution" %in% names(result_tp))
  result_tp$solution else rep(NA, nassets)

weights_mvp <- setNames(weights_mvp, stock_names)
weights_tp <- setNames(weights_tp, stock_names)
weights_trp <- setNames(weights_trp, stock_names)

weights_mvp_flat <- as.vector(weights_mvp)
weights_tp_flat <- as.vector(weights_tp)
weights_trp_flat <- as.vector(weights_trp)

weight_columns <- setNames(as.list(c(weights_mvp_flat, weights_tp_flat, weights_trp_flat)),
                           c(paste0("Weight_", stock_names, "_MVP"),
                             paste0("Weight_", stock_names, "_TP"),
                             paste0("Weight_", stock_names, "_TRP")))

portfolio_return_mvp <- sum(weights_mvp * mean_vect)
```

```
    portfolio_return_tp <- if (!is.null(result_tp)) sum(weights_tp * mean_vect) else NA
    portfolio_return_trp <- if (!is.null(result_trp)) sum(weights_trp * mean_vect) else NA

    portfolio_sd_mvp <- sqrt(result_mvp$value)
    portfolio_sd_tp <- if (!is.null(result_tp) && !is.na(result_tp[1])) sqrt(result_tp$value) else NA
    portfolio_sd_trp <- if (!is.null(result_trp) && !is.na(result_trp[1])) sqrt(result_trp$value) else

    sharpe_mvp <- (portfolio_return_mvp - mufree) / portfolio_sd_mvp
    sharpe_tp <- (portfolio_return_tp - mufree) / portfolio_sd_tp
    sharpe_trp <- if (!is.null(result_trp)) (portfolio_return_trp - mufree) / portfolio_sd_trp else NA

    sharpe_table <- rbind(sharpe_table, data.frame(
    Semester = semester,
    Sharpe_MVP = sharpe_mvp,
    Sharpe_TP = sharpe_tp,
    Sharpe_TRP = sharpe_trp,
    stringsAsFactors = FALSE
  ))

    weights_table <- rbind(weights_table, data.frame(
    Semester = semester,
    weight_columns,
    stringsAsFactors = FALSE
  ))
}

#print(sharpe_table)
#print(weights_table)
```

```
sharpe_table <- sharpe_table %>%
  mutate(Semester = recode(Semester,
                           "2" = "2015-S2", "3" = "2016-S1", "4" = "2016-S2",
                           "5" = "2017-S1", "6" = "2017-S2", "7" = "2018-S1",
                           "8" = "2018-S2", "9" = "2019-S1", "10" = "2019-S2",
                           "11" = "2020-S1", "12" = "2020-S2", "13" = "2021-S1",
                           "14" = "2021-S2", "15" = "2022-S1", "16" = "2022-S2",
                           "17" = "2023-S1", "18" = "2023-S2", "19" = "2024-S1",
                           "20" = "2024-S2"))

weights_table <- weights_table %>%
  mutate(Semester = recode(Semester,
                           "2" = "2015-S2", "3" = "2016-S1", "4" = "2016-S2",
                           "5" = "2017-S1", "6" = "2017-S2", "7" = "2018-S1",
                           "8" = "2018-S2", "9" = "2019-S1", "10" = "2019-S2",
                           "11" = "2020-S1", "12" = "2020-S2", "13" = "2021-S1",
                           "14" = "2021-S2", "15" = "2022-S1", "16" = "2022-S2",
                           "17" = "2023-S1", "18" = "2023-S2", "19" = "2024-S1",
                           "20" = "2024-S2"))
```

iii

```
ggplot(sharpe_table, aes(x = Semester, group=1)) +
  geom_line(aes(y = Sharpe_MVP, color = "MVP"), linewidth = 1) +
  geom_line(aes(y = Sharpe_TP, color = "TP"), linewidth = 1) +
  geom_line(aes(y = Sharpe_TRP, color = "TRP"), linewidth = 1) +
  labs(title = "Sharpe Ratios Over Semesters", y = "Sharpe Ratio", x = "Semester") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
# MVP weights
mvp_columns <- c("Semester", grep("MVP", colnames(weights_table), value = TRUE))
weights_mvp <- weights_table[, mvp_columns]

# TP weights
tp_columns <- c("Semester", grep("TP", colnames(weights_table), value = TRUE))
weights_tp <- weights_table[, tp_columns]

# TRP weights
trp_columns <- c("Semester", grep("TRP", colnames(weights_table), value = TRUE))
weights_trp <- weights_table[, trp_columns]

# rename columns, remove the suffix
rename_weights <- function(df) {
  colnames(df) <- gsub("(_MVP|_TP|_TRP)$", "", colnames(df))  # Remove suffix
  return(df)
}

weights_mvp <- rename_weights(weights_mvp)
weights_tp  <- rename_weights(weights_tp)
weights_trp <- rename_weights(weights_trp)
# print(weights_mvp)
# print(weights_tp)
# print(weights_trp)
```

```
tech_stocks <- c("Weight_Amazon", "Weight_Apple", "Weight_Cisco",
                 "Weight_Meta", "Weight_Microsoft", "Weight_Netflix")

bank_stocks <- c("Weight_AMEX", "Weight_BAC", "Weight_C",
                 "Weight_GS", "Weight_MS", "Weight_WFC")

# function to aggregate sector weights
aggregate_sector_weights <- function(weights_df, portfolio_type) {
  weights_df %>%
    mutate(
      Tech_Sector = rowSums(dplyr::select(., all_of(tech_stocks)), na.rm = TRUE),
      Banking_Sector = rowSums(dplyr::select(., all_of(bank_stocks)), na.rm = TRUE),
      Type = portfolio_type
    ) %>%
    dplyr::select(Semester, Type, Tech_Sector, Banking_Sector)
}
mvp_sectors <- aggregate_sector_weights(weights_mvp, "MVP")
```

```r
tp_sectors  <- aggregate_sector_weights(weights_tp, "TP")
trp_sectors <- aggregate_sector_weights(weights_trp, "TRP")

# combine portfolios
sector_weights <- bind_rows(mvp_sectors, tp_sectors, trp_sectors)
#sector_weights

# convert semester to a factor
sector_weights <- sector_weights %>%
  mutate(Semester = factor(Semester, levels = unique(Semester), ordered = TRUE))

# plot weights
ggplot(sector_weights, aes(x = Semester, y = Tech_Sector, color = Type, group = Type)) +
  geom_line(linewidth = 1) +
  labs(title = "Tech Sector Weights Over Time", y = "Weight", x = "Semester") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

ggplot(sector_weights, aes(x = Semester, y = Banking_Sector, color = Type, group = Type)) +
  geom_line(linewidth = 1) +
  labs(title = "Banking Sector Weights Over Time", y = "Weight", x = "Semester") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Part b

i

```r
# daily log returns for SPY
spy_prices <- datasets[["SPY"]]
spy_returns <- log(spy_prices[-1] / head(spy_prices, -1))
market_returns <- data.frame(Date = as.Date(names(spy_returns)), SPY = spy_returns)

market_returns <- market_returns %>% # create semester column
  mutate(Semester = paste0(year(Date), "-", ifelse(month(Date) <= 6, "S1", "S2")))

# avg daily return
market_returns <- market_returns %>%
  group_by(Semester) %>%
  summarise(Avg_Daily_Return = mean(SPY, na.rm = TRUE))  # average daily return

# scale to get semester log return
market_returns <- market_returns %>%
  mutate(Semester_Return = Avg_Daily_Return * 126)
#head(market_returns)

# get daily prices from part a, create Date column
daily_prices$Date <- rownames(daily_prices)

# add semester column
daily_prices <- daily_prices %>%
```

```
 mutate(Semester = paste0(year(Date), "-", ifelse(month(Date) <= 6, "S1", "S2")))
#head(daily_prices)
```

ii

```
# convert to long format
daily_prices_long <- daily_prices %>%
    pivot_longer(cols = -c(Date, Semester), names_to = "Stock", values_to = "Price") %>%
    filter(Semester >= "2015-S2" & Semester <= "2024-S2")

# convert weights to long format, remove weight prefix
weights_mvp_long <- weights_mvp %>%
  pivot_longer(cols = -Semester, names_to = "Stock", values_to = "Weight") %>%
  mutate(Stock = gsub("Weight_", "", Stock)) %>%
  mutate(Stock = str_replace(Stock, "_MVP$", ""))
weights_tp_long <- weights_tp %>%
  pivot_longer(cols = -Semester, names_to = "Stock", values_to = "Weight") %>%
  mutate(Stock = gsub("Weight_", "", Stock)) %>%
  mutate(Stock = str_replace(Stock, "_TP$", ""))
weights_trp_long <- weights_trp %>%
  pivot_longer(cols = -Semester, names_to = "Stock", values_to = "Weight") %>%
  mutate(Stock = gsub("Weight_", "", Stock)) %>%
  mutate(Stock = str_replace(Stock, "_TRP$", ""))

# join with daily prices to apply semester weights
daily_weight_mvp <- daily_prices_long %>%
  left_join(weights_mvp_long, by = c("Semester", "Stock")) %>%
  group_by(Semester, Stock) %>%
  fill(Weight, .direction = "downup")
#print(daily_weight_mvp)
daily_weight_tp <- daily_prices_long %>%
  left_join(weights_tp_long, by = c("Semester", "Stock")) %>%
  group_by(Semester, Stock) %>%
  fill(Weight, .direction = "downup")
#print(daily_weight_tp)
daily_weight_trp <- daily_prices_long %>%
  left_join(weights_trp_long, by = c("Semester", "Stock")) %>%
  group_by(Semester, Stock) %>%
  fill(Weight, .direction = "downup")
#print(daily_weight_trp)

# merge daily prices with weights for each portfolio type
daily_portfolio_mvp <- daily_weight_mvp %>%
  mutate(Weighted_Price = Price * Weight) %>%
  group_by(Date, Semester) %>%
  summarise(Portfolio_Value = sum(Weighted_Price, na.rm = TRUE), .groups = "drop") %>%
  mutate(Type = "MVP")
daily_portfolio_tp <- daily_weight_tp %>%
  mutate(Weighted_Price = Price * Weight) %>%
  group_by(Date, Semester) %>%
  summarise(Portfolio_Value = sum(Weighted_Price, na.rm = TRUE), .groups = "drop") %>%
  mutate(Type = "TP")
```

```r
daily_portfolio_trp <- daily_weight_trp %>%
  mutate(Weighted_Price = Price * Weight) %>%
  group_by(Date, Semester) %>%
  summarise(Portfolio_Value = sum(Weighted_Price, na.rm = TRUE)) %>%
  mutate(Type = "TRP")

# combine values of 3 portfolios together
daily_portfolio_values <- bind_rows(daily_portfolio_mvp, daily_portfolio_tp, daily_portfolio_trp)
#head(daily_portfolio_values)

daily_portfolio_returns <- daily_portfolio_values %>%
  group_by(Type) %>%
  arrange(Date) %>%
  mutate(Return = log(Portfolio_Value / lag(Portfolio_Value))) %>%
  ungroup()

# convert to semester scale
semester_portfolio_returns <- daily_portfolio_returns %>%
  group_by(Semester, Type) %>%
  summarise(Semester_Return_Portfolio = sum(Return, na.rm = TRUE)) %>%
  ungroup()

# join with market returns and risk-free rate
excess_returns <- semester_portfolio_returns %>%
  left_join(market_returns, by = "Semester") %>%
  mutate(Excess_Return = Semester_Return_Portfolio - Semester_Return)  # Subtract market return
#head(excess_returns)

# remove NAs, NaN, Inf, -Inf
excess_returns_clean <- excess_returns %>%
  filter(
    !is.na(Excess_Return) &
    !is.nan(Excess_Return) &
    is.finite(Excess_Return)
  )
```

iii

```r
# run CAPM model
capm_results <- excess_returns_clean %>%
  group_by(Type) %>%
  summarise(
    alpha = coef(lm(Excess_Return ~ Semester_Return, data = cur_data()))[1],
    beta  = coef(lm(Excess_Return ~ Semester_Return, data = cur_data()))[2],
    p_value_alpha = summary(lm(Excess_Return ~ Semester_Return, data = cur_data()))$coefficients[1, 4]
  )

print(capm_results)
```

Problem 2

Part a

i

```
returns_df <- as.data.frame(returns_df)

# monthly log returns
monthly_returns <- returns_df %>%
  mutate(YearMonth = floor_date(Date, "month")) %>%
  group_by(YearMonth) %>%
  summarise(across(where(is.numeric), \(x) sum(x, na.rm = TRUE))) %>%
  rename(Date = YearMonth) %>%
  ungroup()
monthly_returns$Date <- as.Date(monthly_returns$Date, format="%Y-%m-%d")

# historical monthly log returns (Jan 2015 - Dec 2023)
start_date <- as.Date("2015-01-01")
end_date <- as.Date("2023-12-31")
historical_monthly_returns <- monthly_returns[monthly_returns$Date >= start_date & monthly_returns$Date
#head(historical_monthly_returns)

historical_monthly_returns$Date <- as.Date(historical_monthly_returns$Date)
rownames(historical_monthly_returns) <- historical_monthly_returns$Date # set the date as the index
historical_monthly_returns$Date <- NULL # remove Date column for analysis

# monthly risk free rate
risk_free_data$Date <- as.Date(risk_free_data$Date)

# aggregate daily to monthly average risk free rate
monthly_risk_free <- risk_free_data %>%
  mutate(Date = floor_date(Date, "month")) %>%
  group_by(Date) %>%
  summarise(Avg_Rate = mean(Rate, na.rm = TRUE) / 12) %>%
  ungroup()

# historical period rates (Jan 2015 - Dec 2023)
historical_monthly_risk_free <- monthly_risk_free %>%
  filter(Date >= as.Date("2015-01-01") & Date <= as.Date("2023-12-01"))
#head(historical_monthly_risk_free)
```

ii

```
# fit ARIMA model to forecast the 12 months of returns
forecast_returns <- function(stock_returns) {
  fit <- arima(stock_returns, order = c(1, 0, 0))
  forecast_result <- predict(fit, n.ahead = 12)
  return(forecast_result$pred)
}
```

```r
expected_returns <- sapply(historical_monthly_returns, forecast_returns)

# expected returns for Jan 2024 to Dec 2024
expected_returns <- as.data.frame(expected_returns)
expected_returns <- expected_returns %>%
  mutate(Forecast_Month = seq(as.Date("2024-01-01"), by = "month", length.out = 12)) #%>% dplyr::select
head(expected_returns)

# monthly covariance matrix
cov_matrix_monthly <- cov(dplyr::select(expected_returns, -Forecast_Month), use = "complete.obs")

historical_monthly_risk_free <- historical_monthly_risk_free %>%
  arrange(Date)

# convert to time series object
ts_risk_free <- ts(historical_monthly_risk_free$Avg_Rate, start = c(2015, 1), frequency = 12)

# fit ARIMA model to forecast for 12 months of risk free rates
arima_model <- auto.arima(ts_risk_free)

forecast_2024 <- as.data.frame(forecast(arima_model, h = 12))
forecast_months <- rownames(forecast_2024)
forecast_dates <- as.Date(paste("01", forecast_months), format = "%d %b %Y")

expected_risk_free <- forecast_2024 %>%
  mutate(Date = forecast_dates) %>%
  dplyr::select(Date, `Point Forecast`)

#print(expected_risk_free)
#plot(forecast_2024)
```

```r
sharpe_table_2a <- data.frame(
  Month = integer(),
  Sharpe_MVP = numeric(),
  Sharpe_TP = numeric(),
  Sharpe_TRP = numeric(),
  stringsAsFactors = FALSE
)

weights_table_2a <- data.frame(
  Month = integer(),
  stringsAsFactors = FALSE
)

nassets = 13
stock_names <- colnames(dplyr::select(expected_returns, -Forecast_Month)) # exclude Forecast_Month

# loop over 12 months in 2024
for (month in 1:12) {
    mean_vect = colMeans(dplyr::select(expected_returns, -Forecast_Month))
    cov_mat = cov_matrix_monthly
    cov_regularized <- cov_mat + diag(1e-3, ncol(cov_mat))
    sd_vect = sqrt(diag(cov_mat))
```

```r
Amat = cbind(rep(1, nassets), mean_vect, diag(1, nrow = nassets))

min_return <- min(colMeans(dplyr::select(expected_returns, -Forecast_Month)), na.rm = TRUE)
max_return <- max(colMeans(dplyr::select(expected_returns, -Forecast_Month)), na.rm = TRUE)
muP <- seq(min_return + 1e-4, max_return, by = 0.0005)

mufree <- expected_risk_free[month,]$`Point Forecast`  # risk free rate

weights_mvp_2a = matrix(0, nrow = length(muP) , ncol = nassets)
weights_tp_2a = matrix(0, nrow = length(muP) , ncol = nassets)
weights_trp_2a = matrix(0, nrow = 1 , ncol = nassets)

# TP
for (i in 1:length(muP)) {
  bvec_tp <- c(1, muP[i], rep(0, nassets))
  result_tp <- solve.QP(
    Dmat = 2 * cov_regularized,
    dvec = rep(0, nassets),
    Amat = Amat,
    bvec = bvec_tp,
    meq = 2)
  }
# MVP
for (i in 1:length(muP)) {
  bvec_mvp <- c(1, rep(0, nassets + 1))
  result_mvp <- solve.QP(
    Dmat = 2 * cov_regularized,
    dvec = rep(0, nassets),
    Amat = Amat,
    bvec = bvec_mvp,
    meq = 2)
  }
# TRP
bvec_trp <- c(1, 0.085/12, rep(0, nassets))
result_trp <- tryCatch(
  {solve.QP(
    Dmat = 2 * cov_regularized,
    dvec = rep(0, nassets),
    Amat = Amat,
    bvec = bvec_trp,
    meq = 2)
  }, error = function(e) NA)

weights_mvp_2a <- result_mvp$solution
weights_trp_2a <- if (!is.null(result_trp) && "solution" %in% names(result_trp))
  result_trp$solution else rep(NA, nassets)
weights_tp_2a <- if (!is.null(result_tp) && "solution" %in% names(result_tp))
  result_tp$solution else rep(NA, nassets)

weights_mvp_2a <- setNames(weights_mvp_2a, stock_names)
weights_tp_2a <- setNames(weights_tp_2a, stock_names)
weights_trp_2a <- setNames(weights_trp_2a, stock_names)
```

```
    weights_mvp_flat_2a <- as.vector(weights_mvp_2a)
    weights_tp_flat_2a <- as.vector(weights_tp_2a)
    weights_trp_flat_2a <- as.vector(weights_trp_2a)

    weight_columns_2a <- setNames(as.list(c(weights_mvp_flat_2a, weights_tp_flat_2a, weights_trp_flat_2a
                            c(paste0("Weight_", stock_names, "_MVP"),
                              paste0("Weight_", stock_names, "_TP"),
                              paste0("Weight_", stock_names, "_TRP")))

    portfolio_return_mvp <- sum(weights_mvp_2a * mean_vect)
    portfolio_return_tp <- if (!is.null(result_tp)) sum(weights_tp_2a * mean_vect) else NA
    portfolio_return_trp <- if (!is.null(result_trp)) sum(weights_trp_2a * mean_vect) else NA

    portfolio_sd_mvp <- sqrt(result_mvp$value)
    portfolio_sd_tp <- if (!is.null(result_tp) && !is.na(result_tp[1])) sqrt(result_tp$value) else NA
    portfolio_sd_trp <- if (!is.null(result_trp) && !is.na(result_trp[1])) sqrt(result_trp$value) else

    sharpe_mvp_2a <- (portfolio_return_mvp - mufree) / portfolio_sd_mvp
    sharpe_tp_2a <- (portfolio_return_tp - mufree) / portfolio_sd_tp
    sharpe_trp_2a <- if (!is.null(result_trp)) (portfolio_return_trp - mufree) / portfolio_sd_trp else

    sharpe_table_2a <- rbind(sharpe_table_2a, data.frame(
    Month = month,
    Sharpe_MVP = sharpe_mvp_2a,
    Sharpe_TP = sharpe_tp_2a,
    Sharpe_TRP = sharpe_trp_2a,
    stringsAsFactors = FALSE
  ))

    weights_table_2a <- rbind(weights_table_2a, data.frame(
    Month = month,
    weight_columns,
    stringsAsFactors = FALSE
  ))
}

#print(sharpe_table_2a)
#print(weights_table_2a)
```

```
sharpe_table_2a <- sharpe_table_2a %>%
  mutate(Month = recode(Month,
                        "1" = "2024-01-01", "2" = "2024-02-01", "3" = "2024-03-01",
                        "4" = "2024-04-01", "5" = "2024-05-01", "6" = "2024-06-01",
                        "7" = "2024-07-01", "8" = "2024-08-01", "9" = "2024-09-01",
                        "10" = "2024-10-01", "11" = "2024-11-01", "12" = "2024-12-01"))

weights_table_2a <- weights_table_2a %>%
  mutate(Month = recode(Month,
                        "1" = "2024-01-01", "2" = "2024-02-01", "3" = "2024-03-01",
                        "4" = "2024-04-01", "5" = "2024-05-01", "6" = "2024-06-01",
                        "7" = "2024-07-01", "8" = "2024-08-01", "9" = "2024-09-01",
                        "10" = "2024-10-01", "11" = "2024-11-01", "12" = "2024-12-01"))
```

iii

```
# plot sharpe ratios
ggplot(sharpe_table_2a, aes(x = Month, group=1)) +
  geom_line(aes(y = Sharpe_MVP, color = "MVP"), linewidth = 1) +
  geom_line(aes(y = Sharpe_TP, color = "TP"), linewidth = 1) +
  geom_line(aes(y = Sharpe_TRP, color = "TRP"), linewidth = 1) +
  labs(title = "Sharpe Ratios Over 2024 Months", y = "Sharpe Ratio", x = "Month") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
# MVP weights
mvp_columns_2a <- c("Month", grep("MVP", colnames(weights_table_2a), value = TRUE))
weights_mvp_2a <- weights_table_2a[, mvp_columns_2a]

# TP weights
tp_columns_2a <- c("Month", grep("TP", colnames(weights_table_2a), value = TRUE))
weights_tp_2a <- weights_table_2a[, tp_columns_2a]

# TRP weights
trp_columns_2a <- c("Month", grep("TRP", colnames(weights_table_2a), value = TRUE))
weights_trp_2a <- weights_table_2a[, trp_columns_2a]

# rename columns, remove the suffix
rename_weights <- function(df) {
  colnames(df) <- gsub("(_MVP|_TP|_TRP)$", "", colnames(df))  # Remove suffix
  return(df)
}

weights_mvp_2a <- rename_weights(weights_mvp_2a)
weights_tp_2a  <- rename_weights(weights_tp_2a)
weights_trp_2a <- rename_weights(weights_trp_2a)
# print(weights_mvp)
# print(weights_tp)
# print(weights_trp)
```

```
tech_stocks <- c("Weight_Amazon", "Weight_Apple", "Weight_Cisco",
                 "Weight_Meta", "Weight_Microsoft", "Weight_Netflix")

bank_stocks <- c("Weight_AMEX", "Weight_BAC", "Weight_C",
                 "Weight_GS", "Weight_MS", "Weight_WFC")

# function to aggregate sector weights
aggregate_sector_weights <- function(weights_df, portfolio_type) {
  weights_df %>%
    mutate(
      Tech_Sector = rowSums(dplyr::select(., all_of(tech_stocks)), na.rm = TRUE),
      Banking_Sector = rowSums(dplyr::select(., all_of(bank_stocks)), na.rm = TRUE),
      Type = portfolio_type
    ) %>%
    dplyr::select(Month, Type, Tech_Sector, Banking_Sector)
}
```

```r
mvp_sectors_2a <- aggregate_sector_weights(weights_mvp_2a, "MVP")
tp_sectors_2a  <- aggregate_sector_weights(weights_tp_2a, "TP")
trp_sectors_2a <- aggregate_sector_weights(weights_trp_2a, "TRP")

# combine portfolios
sector_weights_2a <- bind_rows(mvp_sectors_2a, tp_sectors_2a, trp_sectors_2a)
#sector_weights

# convert Month to a factor
sector_weights_2a <- sector_weights_2a %>%
  mutate(Month = factor(Month, levels = unique(Month), ordered = TRUE))

# plot weights
ggplot(sector_weights_2a, aes(x = Month, y = Tech_Sector, color = Type, group = Type)) +
  geom_line(linewidth = 1) +
  labs(title = "Tech Sector Weights Over 2024 Months", y = "Weight", x = "Month") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

ggplot(sector_weights_2a, aes(x = Month, y = Banking_Sector, color = Type, group = Type)) +
  geom_line(linewidth = 1) +
  labs(title = "Banking Sector Weights Over 2024 Months", y = "Weight", x = "Month") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Part b

i

```r
monthly_returns <- returns_df %>%
  mutate(YearMonth = floor_date(Date, "month")) %>%
  group_by(YearMonth) %>%
  summarise(across(where(is.numeric), \(x) sum(x, na.rm = TRUE))) %>%
  rename(Date = YearMonth) %>%
  ungroup()
monthly_returns$Date <- as.Date(monthly_returns$Date, format="%Y-%m-%d")

# filter for historical monthly log returns, risk free rates we need (Jan 2015 - Dec 2023)
start_date <- as.Date("2023-12-01")
end_date <- as.Date("2024-11-30")

historical_monthly_returns <- monthly_returns[monthly_returns$Date >= start_date & monthly_returns$Date
#head(historical_monthly_returns)

historical_monthly_risk_free <- monthly_risk_free[monthly_risk_free$Date >= start_date & monthly_risk_fi
#head(historical_monthly_risk_free)

# reshape historical_monthly_returns to long format
historical_monthly_returns_long <- historical_monthly_returns %>%
  pivot_longer(cols = -Date, names_to = "Stock", values_to = "Monthly_Return") %>%
  arrange(Stock, Date) %>%
```

```
  group_by(Stock) %>%
  mutate(Lagged_Return = lag(Monthly_Return)) %>% # previous month's return
  ungroup()
#historical_monthly_returns_long
```

ii

```
# construct portfolios for 12 months of 2024
start_date <- as.Date("2024-01-01")
end_date <- as.Date("2024-12-01")

sharpe_table_2b <- data.frame(
  Month = integer(),
  Sharpe_MVP = numeric(),
  Sharpe_TP = numeric(),
  Sharpe_TRP = numeric(),
  stringsAsFactors = FALSE
)

weights_table_2b <- data.frame(
  Month = integer(),
  stringsAsFactors = FALSE
)

nassets = 13
stock_names <- colnames(dplyr::select(historical_monthly_returns, -Date)) # exclude Date

for (month in seq(start_date, end_date, by = "1 month")) {

  # use historical data of previous month
  previous_month <- as.Date(month) - months(1)
  mean_vect = colMeans(historical_monthly_returns %>% filter(Date == as.Date(previous_month)) %>% dplyr

  cov_mat = cov_matrix_monthly # monthly cov matrix
  cov_regularized <- cov_mat + diag(1e-3, ncol(cov_mat))
  sd_vect = sqrt(diag(cov_mat))

  Amat = cbind(rep(1, nassets), mean_vect, diag(1, nrow = nassets))

  min_return <- min(mean_vect, na.rm = TRUE)
  max_return <- max(mean_vect, na.rm = TRUE)
  muP <- seq(min_return + 1e-4, max_return, by = 0.0005)

  mufree <- as.numeric(historical_monthly_risk_free %>% filter(Date == previous_month) %>% dplyr::selec

  weights_mvp_2b = matrix(0, nrow = length(muP) , ncol = nassets)
  weights_tp_2b = matrix(0, nrow = length(muP) , ncol = nassets)
  weights_trp_2b = matrix(0, nrow = 1 , ncol = nassets)

  # TP
  for (i in 1:length(muP)) {
    bvec_tp <- c(1, muP[i], rep(0, nassets))
```

```
  result_tp <- solve.QP(
    Dmat = 2 * cov_regularized,
    dvec = rep(0, nassets),
    Amat = Amat,
    bvec = bvec_tp,
    meq = 2)
  }

# MVP
for (i in 1:length(muP)) {
   bvec_mvp <- c(1, rep(0, nassets + 1))
   result_mvp <- solve.QP(
     Dmat = 2 * cov_regularized,
     dvec = rep(0, nassets),
     Amat = Amat,
     bvec = bvec_mvp,
     meq = 2)
   }

# TRP
bvec_trp <- c(1, 0.085/12, rep(0, nassets))
result_trp <- tryCatch(
  {
  solve.QP(
    Dmat = 2 * cov_regularized,
    dvec = rep(0, nassets),
    Amat = Amat,
    bvec = bvec_trp,
    meq = 2)
    }, error = function(e) NA)

  weights_mvp_2b <- result_mvp$solution
  weights_trp_2b <- if (!is.null(result_trp) && "solution" %in% names(result_trp))
    result_trp$solution else rep(NA, nassets)
  weights_tp_2b <- if (!is.null(result_tp) && "solution" %in% names(result_tp))
    result_tp$solution else rep(NA, nassets)

  weights_mvp_2b <- setNames(weights_mvp_2b, stock_names)
  weights_tp_2b <- setNames(weights_tp_2b, stock_names)
  weights_trp_2b <- setNames(weights_trp_2b, stock_names)

  weights_mvp_flat_2b <- as.vector(weights_mvp_2b)
  weights_tp_flat_2b <- as.vector(weights_tp_2b)
  weights_trp_flat_2b <- as.vector(weights_trp_2b)

  weight_columns_2b <- setNames(as.list(c(weights_mvp_flat_2b, weights_tp_flat_2b, weights_trp_flat_2
                          c(paste0("Weight_", stock_names, "_MVP"),
                            paste0("Weight_", stock_names, "_TP"),
                            paste0("Weight_", stock_names, "_TRP")))

  portfolio_return_mvp <- sum(weights_mvp_2b * mean_vect)
  portfolio_return_tp <- if (!is.null(result_tp)) sum(weights_tp_2b * mean_vect) else NA
  portfolio_return_trp <- if (!is.null(result_trp)) sum(weights_trp_2b * mean_vect) else NA
```

```
    portfolio_sd_mvp <- sqrt(result_mvp$value)
    portfolio_sd_tp <- if (!is.null(result_tp) && !is.na(result_tp[1])) sqrt(result_tp$value) else NA
    portfolio_sd_trp <- if (!is.null(result_trp) && !is.na(result_trp[1])) sqrt(result_trp$value) else

    sharpe_mvp_2b <- (portfolio_return_mvp - mufree) / portfolio_sd_mvp
    sharpe_tp_2b <- (portfolio_return_tp - mufree) / portfolio_sd_tp
    sharpe_trp_2b <- if (!is.null(result_trp)) (portfolio_return_trp - mufree) / portfolio_sd_trp else

    sharpe_table_2b <- rbind(sharpe_table_2b, data.frame(
    Month = as.Date(month),
    Sharpe_MVP = sharpe_mvp_2b,
    Sharpe_TP = sharpe_tp_2b,
    Sharpe_TRP = sharpe_trp_2b,
    stringsAsFactors = FALSE
  ))

    weights_table_2b <- rbind(weights_table_2b, data.frame(
    Month = as.Date(month),
    weight_columns_2b,
    stringsAsFactors = FALSE
  ))
}

#print(sharpe_table_2b)
#print(weights_table_2b)
```

iii

```
# plot sharpe ratios
ggplot(sharpe_table_2b, aes(x = Month, group=1)) +
  geom_line(aes(y = Sharpe_MVP, color = "MVP"), linewidth = 1) +
  geom_line(aes(y = Sharpe_TP, color = "TP"), linewidth = 1) +
  geom_line(aes(y = Sharpe_TRP, color = "TRP"), linewidth = 1) +
  labs(title = "Sharpe Ratios Over 2024 Months", y = "Sharpe Ratio", x = "Month") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
# MVP weights
mvp_columns_2b <- c("Month", grep("MVP", colnames(weights_table_2b), value = TRUE))
weights_mvp_2b <- weights_table_2b[, mvp_columns_2b]

# TP weights
tp_columns_2b <- c("Month", grep("TP", colnames(weights_table_2b), value = TRUE))
weights_tp_2b <- weights_table_2b[, tp_columns_2b]

# TRP weights
trp_columns_2b <- c("Month", grep("TRP", colnames(weights_table_2b), value = TRUE))
weights_trp_2b <- weights_table_2b[, trp_columns_2b]

# rename columns, remove the suffix
rename_weights <- function(df) {
```

```r
  colnames(df) <- gsub("(_MVP|_TP|_TRP)$", "", colnames(df))
  return(df)
}

weights_mvp_2b <- rename_weights(weights_mvp_2b)
weights_tp_2b  <- rename_weights(weights_tp_2b)
weights_trp_2b <- rename_weights(weights_trp_2b)
# print(weights_mvp_2b)
# print(weights_tp_2b)
# print(weights_trp_2b)
```

```r
tech_stocks <- c("Weight_Amazon", "Weight_Apple", "Weight_Cisco",
                 "Weight_Meta", "Weight_Microsoft", "Weight_Netflix")

bank_stocks <- c("Weight_AMEX", "Weight_BAC", "Weight_C",
                 "Weight_GS", "Weight_MS", "Weight_WFC")

# function to aggregate sector weights
aggregate_sector_weights <- function(weights_df, portfolio_type) {
  weights_df %>%
    mutate(
      Tech_Sector = rowSums(dplyr::select(., all_of(tech_stocks)), na.rm = TRUE),
      Banking_Sector = rowSums(dplyr::select(., all_of(bank_stocks)), na.rm = TRUE),
      Type = portfolio_type
    ) %>%
    dplyr::select(Month, Type, Tech_Sector, Banking_Sector)
}
mvp_sectors_2b <- aggregate_sector_weights(weights_mvp_2b, "MVP")
tp_sectors_2b  <- aggregate_sector_weights(weights_tp_2b, "TP")
trp_sectors_2b <- aggregate_sector_weights(weights_trp_2b, "TRP")

# combine portfolios
sector_weights_2b <- bind_rows(mvp_sectors_2b, tp_sectors_2b, trp_sectors_2b)
#sector_weights

# convert Month to a factor
sector_weights_2b <- sector_weights_2b %>%
  mutate(Month = factor(Month, levels = unique(Month), ordered = TRUE))

# plot weights
ggplot(sector_weights_2b, aes(x = Month, y = Tech_Sector, color = Type, group = Type)) +
  geom_line(linewidth = 1) +
  labs(title = "Tech Sector Weights Over 2024 Months", y = "Weight", x = "Month") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

ggplot(sector_weights_2b, aes(x = Month, y = Banking_Sector, color = Type, group = Type)) +
  geom_line(linewidth = 1) +
  labs(title = "Banking Sector Weights Over 2024 Months", y = "Weight", x = "Month") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```r
# get dates for each stock
all_dates <- lapply(datasets_combined[selected_stocks], function(stock) as.Date(names(stock)))

# find the common trading dates across all stocks
common_dates <- Reduce(intersect, all_dates)
aligned_prices <- lapply(datasets_combined[selected_stocks], function(stock) {
  stock[as.Date(names(stock)) %in% as.Date(common_dates)]
})

# convert to df and ensure the same date index
daily_prices <- as.data.frame(aligned_prices)
rownames(daily_prices) <- as.character(as.Date(common_dates))

daily_prices$Date <- as.Date(rownames(daily_prices))

# calculate monthly closing prices
monthly_prices <- daily_prices %>%
  group_by(month = format(Date, "%Y-%m")) %>%
  summarise(across(-Date, last)) %>%
  ungroup()

# convert month to Date format
monthly_prices$Date <- as.Date(paste0(monthly_prices$month, "-01"))
monthly_prices <- monthly_prices %>% dplyr::select(-month)

#head(monthly_prices)
```

```r
# end of month closing prices
monthly_closing_prices <- daily_prices %>%
  group_by(year_month = floor_date(Date, "month")) %>%
  filter(Date == max(Date)) %>%
  ungroup() %>%
  filter(Date >= "2024-01-01" & Date <= "2025-01-01") %>% # filter for months we need
  dplyr::select(-Date) #%>%
#monthly_closing_prices
```

```r
# remove prefix from column names
rename_weights <- function(df) {
  colnames(df) <- gsub("^Weight_", "", colnames(df))
  return(df)
}
weights_mvp_2a <- rename_weights(weights_mvp_2a)
weights_mvp_2b <- rename_weights(weights_mvp_2b)
weights_tp_2a <- rename_weights(weights_tp_2a)
weights_tp_2a <- rename_weights(weights_tp_2a)
weights_trp_2a <- rename_weights(weights_trp_2a)
weights_trp_2a <- rename_weights(weights_trp_2a)


# calculate portfolio values
compute_portfolio_values <- function(prices, weights) {
  prices_long <- prices %>%
    pivot_longer(-year_month, names_to = "Stock", values_to = "Price")
```

```
  weights_long <- weights %>%
    pivot_longer(-Month, names_to = "Stock", values_to = "Weight") %>%
    rename(year_month = Month) %>%
    mutate(year_month = as.Date(year_month))

  # join prices with weights
  portfolio_values <- prices_long %>%
    left_join(weights_long, by = c("year_month", "Stock")) %>%
    mutate(Value = Price * Weight) %>%
    group_by(year_month) %>%
    summarise(Portfolio_Value = sum(Value, na.rm = TRUE))
  return(portfolio_values)
}

# compute portfolio values under both methods
portfolio_values_mvp_2a <- compute_portfolio_values(monthly_closing_prices, weights_mvp_2a)
portfolio_values_mvp_2b <- compute_portfolio_values(monthly_closing_prices, weights_mvp_2b)

portfolio_values_tp_2a <- compute_portfolio_values(monthly_closing_prices, weights_tp_2a)
portfolio_values_tp_2b <- compute_portfolio_values(monthly_closing_prices, weights_tp_2b)

portfolio_values_trp_2a <- compute_portfolio_values(monthly_closing_prices, weights_trp_2a)
portfolio_values_trp_2b <- compute_portfolio_values(monthly_closing_prices, weights_trp_2b)
```

```
# plot comparison of Method A vs Method B
portfolio_values_mvp <- portfolio_values_mvp_2a %>%
  rename(Portfolio_A = Portfolio_Value) %>%
  left_join(portfolio_values_mvp_2b %>% rename(Portfolio_B = Portfolio_Value), by = "year_month")

ggplot(portfolio_values_mvp, aes(x = year_month)) +
  geom_line(aes(y = Portfolio_A, color = "Method A"), linewidth = 1) +
  geom_line(aes(y = Portfolio_B, color = "Method B"), linewidth = 1) +
  labs(title = "MVP Portfolio Value Comparison",
       x = "Month",
       y = "Portfolio Value") +
  theme_minimal()
```

```
portfolio_values_tp <- portfolio_values_tp_2a %>%
  rename(Portfolio_A = Portfolio_Value) %>%
  left_join(portfolio_values_tp_2b %>% rename(Portfolio_B = Portfolio_Value), by = "year_month")

ggplot(portfolio_values_tp, aes(x = year_month)) +
  geom_line(aes(y = Portfolio_A, color = "Method A"), linewidth = 1) +
  geom_line(aes(y = Portfolio_B, color = "Method B"), linewidth = 1) +
  labs(title = "TP Portfolio Value Comparison",
       x = "Month",
       y = "Portfolio Value") +
  theme_minimal()
```

```
portfolio_values_trp <- portfolio_values_trp_2a %>%
  rename(Portfolio_A = Portfolio_Value) %>%
  left_join(portfolio_values_trp_2b %>% rename(Portfolio_B = Portfolio_Value), by = "year_month")
```

```
ggplot(portfolio_values_trp, aes(x = year_month)) +
  geom_line(aes(y = Portfolio_A, color = "Method A"), linewidth = 1) +
  geom_line(aes(y = Portfolio_B, color = "Method B"), linewidth = 1) +
  labs(title = "TRP Portfolio Value Comparison",
       x = "Month",
       y = "Portfolio Value") +
  theme_minimal()
```

Problem 3

i

```
# historical monthly returns (2015-2023)
historical_returns <- monthly_returns %>%
  filter(Date >= "2015-02-01" & Date <= "2023-12-31")
```

ii & iii

```
set.seed("1010")

returns.all = historical_returns[-1]  # exclude Date column
S = 1e8 # 100 million
alpha = 0.01
params.multt = fit_mvt(returns.all)

# months in 2024 that we will calculate VaR and confidence intervals
months_2024 = seq.Date(from = as.Date("2024-01-01"), to = as.Date("2024-12-01"), by = "month")

# weights for the 3 portfolios: MVP, TP, and TRP
weights_list = list(weights_mvp_2a, weights_tp_2a, weights_trp_2a)
weights_names = c("MVP", "TP", "TRP")

VaR_over_time = data.frame(
  Month = rep(months_2024, times = length(weights_list)),
  Portfolio = rep(weights_names, each = length(months_2024)),
  VaR = numeric(length(months_2024) * length(weights_list)),
  CI_95_lower = numeric(length(months_2024) * length(weights_list)),
  CI_95_upper = numeric(length(months_2024) * length(weights_list))
)

# loop through each portfolio and each month
row_idx = 1
for (i in seq_along(weights_list)) {
  for (month in months_2024) {
    # get weights for the given portfolio and month
    weight = as.numeric(weights_list[[i]][weights_list[[i]]$Month == as.Date(month), -1])

    # calculate VaR
```

```
    mu.p = params.multt$mu %*% weight
    sigma.p = sqrt(weight %*% params.multt$cov %*% weight)
    VaR = -S * (mu.p + sigma.p * qt(alpha, params.multt$nu))

    # calculate the 95% confidence interval
    B = 500
    VaR.boot.results = array()

    for (repl in 1:B) {
      indices = sample(1:nrow(returns.all), nrow(returns.all), replace = TRUE)
      returns.bootsamples = returns.all[indices, ]
      params.multt.boot = fit_mvt(returns.bootsamples)

      mu.p.boot = params.multt.boot$mu %*% weight
      sigma.p.boot = sqrt(weight %*% params.multt.boot$cov %*% weight)
      VaR.boot = -S * (mu.p.boot + sigma.p.boot * qt(alpha, params.multt.boot$nu))

      VaR.boot.results = cbind(VaR.boot.results, VaR.boot)
    }

    VaR.boot.results = VaR.boot.results[, 2:B]

    # store the VaR and confidence intervals
    VaR_over_time[row_idx, "VaR"] = VaR
    VaR_over_time[row_idx, "CI_95_lower"] = quantile(VaR.boot.results, 0.005, na.rm = TRUE)
    VaR_over_time[row_idx, "CI_95_upper"] = quantile(VaR.boot.results, 0.995, na.rm = TRUE)
    row_idx = row_idx + 1
  }
}

print(VaR_over_time)
```

Problem 4

i

```
# risk_free_data read in during problem 1

risk_free_data$Date <- as.Date(risk_free_data$Date, format="%Y-%m-%d")

# add semester column
risk_free_data <- risk_free_data %>%
  mutate(Semester = paste0(year(Date), "-S", ifelse(month(Date) <= 6, 1, 2)))

# avg risk free rate for each semester, scale to get semester risk free rate
risk_free_semester <- risk_free_data %>%
  group_by(Semester) %>%
  summarise(Avg_Rate = mean(Rate, na.rm = TRUE) / 2)

#head(risk_free_semester)
```

ii

```
# calculate bond prices for each semester
# P = V(1 + r/2) ^ -2T
risk_free_semester$Price <- 1000 / (1 + risk_free_semester$Avg_Rate / 2) ^ 0.5
#head(risk_free_semester)
```

iii

```
# plot interest rates and bond prices
ggplot(risk_free_semester, aes(x = Avg_Rate, y = Price)) +
  geom_point(color = "blue", size = 3) +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Zero-Coupon Bond Price vs. Interest Rate",
       x = "Interest Rate",
       y = "Bond Price") +
  theme_minimal()
```

```
# correlation between interest rates and bond prices
correlation <- cor(risk_free_semester$Avg_Rate, risk_free_semester$Price)
print(paste("Correlation between interest rates and bond prices:", correlation))
```