Financial Research project comparing Single-Index Regression Model with Fama French 3-factor model

Khizar Tahir

A. Introduction

Using Financial Models for Performance Evaluation

Analysts utilize several methods to evaluate a companies financial performance. The financial reports of a public company allow stakeholders to gain deep insights into financial performance, but there are methods outside of reports that also provide valuable information. Looking outside the financial statements allows users to use training variables to predict how a company will perform against the market in a given period.

The underlying concept of a financial model states that stock returns are typcally sensitive to a set of factors. The Single Index Model (SIM) allows us to compare stock returns with market returns and is used to explain the performance of a stock using one market index. Further, the Fama French 3 (FF3) Model utilizes not only the market index factor, but also small minus big (SMB), which is concerned with the size of the firm, and high minus low (HML), which is concerned with whether a firm is growth firm or value firm.

Financial Models can be represented by the following equations:

Traditional Linear Model:

$$y = \alpha + \beta x$$

Single Index Model:

$$R_i - R_f = \alpha_i + \beta_i(R_m - R_f)$$

Fama French 3 Factor Model:

$$r = R_f + \beta_1(R_m - R_f) + \beta_s SMB + \beta_v HML + \alpha$$

Where:

$R_f = riskfreerate,$
$R_m = marketreturn,$
$R_i = stockreturn,$
$SMB = SmallminusBig,$
$HML = HighminusLow,$
$R_i - R_f = excessreturnonstock$

B. Project Overview

In this project, we independently run the two regression models using the same dataset. We pick a training period, run regression, and evaluate the coefficients in both the equations as displayed above and use those estimates to make predictions about Returns for our prediction period. We then match our results with actual data to analyse how our models performed.

We will do the process through a user function that will take as input the file names to retrieve data from, vector for our training period, and a vector for our prediction period.

Our function will return as output, a main table containing various aspects of model analyses, a summary table, and a plot presenting an aspect of our analyses.

C. Code Explanation

Part 1. Including Required Packages for Analysis

```
library(dplyr)
library(tidyverse)
library(reshape)
library(cowplot)
```

Part 2. Reading Datasets and Merging Dataframes

In order to run single and multiple index regressions models, several factors are required. We were provided with two datasets to accomplish our tasks. The first dataset lists the returns of specific stocks over a period of time. The second data set is compiled by Professor French and his staff, a listing of market returns, SMB, HML and risk free rates over a period of time. Both of these datasets are crucial to runing a Single Index Model and a Fama French 3 model.

Our first step is to read the provided datasets into dataframes, complete any required data cleaning and then merge the dataframes in order to join the Fama French 3 data with each of the tickers in the Stock data.

In this section, we set our dates for training and prediction and for data cleaning, we ensured that the dates in both dataframes were formatted the same, which is an important step in time series analysis.

Our merged dataframe, titled df, uses dplyr to select only the columns we need for running our regressions.

```
# Preparing to read data
fil_dir <- "C:/users/chrdo/Downloads/"
stocks_file <- paste(fil_dir, as.character(stocks_filename), sep = "")
ff3facs_file <- paste(fil_dir, as.character(ff3facs_filename), sep = "")

# Setting dates for training and prediction
training_from <- as.numeric(training[1])
training_till <- as.numeric(training[length(training)])
prediction_from <- as.numeric(pred[1])
prediction_till <- as.numeric(pred[length(pred)])

# Reading data
stocks <- read.csv(stocks_file, na.strings = "C")
stocks$date1 <- as.Date(as.character(stocks$date), "%m/%d/%Y")
stocks$finaldate <- format(stocks$date1, "%Y%m%d")

ff3facs <- read.csv(ff3facs_file, na.strings = "C")
#print(colnames(ff3facs[1])) # first column is already names X.
ff3facs$X <- as.character(ff3facs$X) # renaming hidden X to X so the name is replaced and visible.

# Merging the two dataframes into one.
df_temp <- merge(stocks, ff3facs, by.x = "finaldate", by.y = "X")
df_temp$date <- as.numeric(df_temp$finaldate)

df <- df_temp %>%
  mutate(exRET = RET - (RF/100),
         FacMkt = Mkt.RF/100,
         FacSMB = SMB/100,
         FacHML = HML/100,
         RF_dec = RF/100) %>%
  select(date, date1, TICKER, exRET, FacMkt, FacSMB, FacHML, RF_dec)
```

Part 3. Creating Training and Prediction Dataframes for SIM and FF3

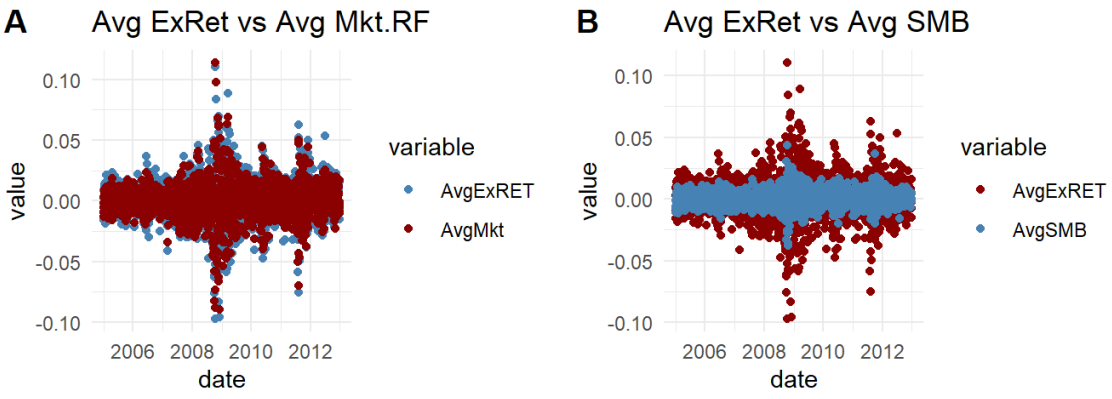
Once we have read our datasets and merged into one dataframe, we can use the merged dataframe to create training and prediction dataframes that will cater to specific dates we would like to analyze. We create two dataframes for the training period - one for SIM and one for FF3. We then run respective regressions for each model and store the results as a list containing regressions results for each ticker.

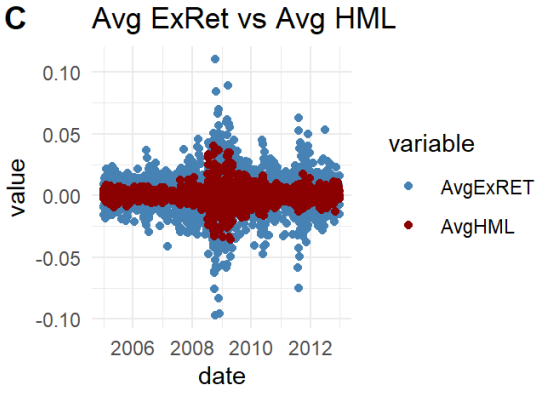
We then created a prediction dataframe that contains the actual data for the prediction period.

```
  # Final dataframe to run SIM regression on. This dataframe will have TICKERS and for each ticker, a regressi
on result.
  df_training_sim <- df %>%
    filter((date >= training_from) & (date <= training_till)) %>%
    group_by(TICKER) %>%
    summarise(reg_result = list(lm(exRET ~ FacMkt)))

  # Final dataframe to run 3FF regression on. This dataframe will have TICKERS and for each ticker, a regressi
on result.
  df_training <- df %>%
    filter((date >= training_from) & (date <= training_till)) %>%
    group_by(TICKER) %>%
    summarise(reg_result = list(lm(exRET ~ FacMkt + FacSMB + FacHML)))

  # Creating prediction dataframe - ACTUAL data.
  df_prediction_temp <- df %>%
    filter((date >= prediction_from) & (date <= prediction_till)) %>%
    select(date, date1, TICKER, exRET, FacMkt, FacSMB, FacHML) %>%
    arrange(TICKER)
```

Visualizing The Data Prior to Analyzing Final Results

Mapping average returns for all stocks by date relative to averages of the three different paramters, we can get a glimpse into the behavior the model might predict. Below we observe that, for the training data set, the Mkt.Rf parameter closely impacts the average return varaible for all stocks by date. The other two factors show quite less impact than Mkt.Rf.





Part 4. Merging Dataframes to Retain Firms with Data in Both Training and Prediction Periods

It is important to note that some companies had data in the training period but did not have data in the prediction period and vice versa. For the purpose of our analysis, we decided to only proceed analyzing companies with data in both periods. We completed that as follows.

We started by creating the pred_firms dataframe that stored all of the unique tickers from the df_prediction_temp dataframe in Part 3. We then used merge functions to combine the list of unique tickers (pred_firms) and the training dataframes in Part 3 for both SIM and FF3.

Then, we created the pred_firms_merge_sim (for SIM) and pred_firms_merge (for FF3) dataframes to output only those companies with data in both periods for each type of regression.

The result of the above shows us the names of the firms with complete data. df_prediction_sim (for SIM) and df_prediction (for FF3) filter the above results to only list firms with complete data.

Finally, We arrange these final dataframes and use split to convert to a list of dataframes split based on ticker and then convert those lists to matrices for subsequent analysis.

The ultimate prediction matrix that has to be multiplied needs to have columns adjusted to the rows of our to-be-created coefficient matrix to allow matrix multiplication and a successful resulting prediction based on our equations. So we create matrices accordingly.

```
# Creating a df to store unique tickers from prediction dataset to know which firms do we need to predict.
pred_firms <- data.frame("TICKER" = unique(df_prediction_temp$TICKER))

# Creating a df, which combines/merges the dataframes for prediction df AND the df_training dataframes.
pred_firms_merge_sim <- merge(pred_firms, df_training_sim, by.x = 'TICKER', by.y = 'TICKER')
pred_firms_merge <- merge(pred_firms, df_training, by.x = 'TICKER', by.y = 'TICKER')

#Filter for only firms with complete data and store in df_prediction_sim (SIM) and df_prediction (FF3).
df_prediction_sim <- df_prediction_temp %>%
  filter(TICKER %in% pred_firms_merge_sim$TICKER) %>%
  arrange(TICKER)
df_prediction <- df_prediction_temp %>%
  filter(TICKER %in% pred_firms_merge$TICKER) %>%
  arrange(TICKER)

# List of prediction data by firms. The two variables below contain respective data for SIM and 3FF.
df_prediction_list_sim <-split(df_prediction_sim, df_prediction_sim$TICKER)
df_prediction_list <-split(df_prediction, df_prediction$TICKER)

# Converting our list of to-predict dataframes into matrix. Only featching the FacMkt for SIM.
df_prediction_matrix_sim <- lapply(df_prediction_list_sim, function(x){
  cbind(1, x$FacMkt)
})
# Converting our list of to-predict dataframes into matrix. Only featching the FacMkt for 3FF.
df_prediction_matrix <- lapply(df_prediction_list, function(x){
  cbind(1, x$FacMkt, x$FacSMB, x$FacHML)
})
```

Part 5. Obtaining Regression Results for prediction multiplication and Summary Output

Now that we have pared down our data to just those firms with data in both the training and prediction periods, using our training datasets from Part 3 to create several outputs for both SIM and FF3. All of these results use lapply to pull the coefficients, rsquared values and alphas from the regression outputs calculated in Part 3 with the training dataframes for SIM and FF3. All outputs for this section will be as matrices for work in subsequent parts.

- Matrices for coefficients

- List R-squared values - each corresponding to one firm.

- List of alphas - each corresponding to one firm.

```
  # Calculating coefficients matrices for SIM and 3FF using the merged dataframes from Part 4.
  coeffs_matrix_sim <- lapply(pred_firms_merge_sim$reg_result, function(x){as.matrix(x$coefficients)})
  coeffs_matrix <- lapply(pred_firms_merge$reg_result, function(x){as.matrix(x$coefficients)})

  #Calculating and storing R-squared values for SIM and 3FF.Lists containing these values corresponding to eac
h firm.
  rsqrd_list_sim <- lapply(pred_firms_merge_sim$reg_result, function(x){
    temp <- summary(x)
    temp$r.squared
  })
  rsqrd_list <- lapply(pred_firms_merge$reg_result, function(x){
    temp <- summary(x)
    temp$r.squared
  })

  # Fetching and storing alphas for SIM and 3FF like above.
  alpha_list_sim <- lapply(pred_firms_merge_sim$reg_result, function(x){
    alpha_temp <- as.matrix(x$coefficients)
    alpha_temp[1,1]
  })
  alpha_list <- lapply(pred_firms_merge$reg_result, function(x){
    alpha_temp <- as.matrix(x$coefficients)
    alpha_temp[1,1]
  })
```

Part 6. Matrix Multiplication

Next, we matrix multiply the coefficient matrix from Part 5 and the prediction matrix from Part 6 resuting in a list of lists. Each sub-list contains all predictions for the predicted duration. We carefully handle matrix dimensions to ensure multiplication is possible and the resulting value is a single number for each day.

```
  # Multiplying coeff matrices for SIM with prediction matrices for SIM.
  predictions_sim <- mapply(function(a,b) a %*% b, df_prediction_matrix_sim, coeffs_matrix_sim)
  # Multiplying coeff matrices for 3FF with prediction matrices for 3FF.
  predictions <- mapply(function(a,b) a %*% b, df_prediction_matrix, coeffs_matrix)
```

Part 7. Adjusting the Length of Lists for Rsquared and Alpha to Align with our Final Dataframe.

In this section, we apply the r squared and alpha values to the entire dataframe since only one rsquared and alpha value is calculated per firm, but our current dataframe outputs daily values.

```
  # For each firm, there are many rows corresponding to daily data but only 1 R-squared value. So just scaling
the size of Rsquared to match our dataframe.
  rsqrd_list_adjusted_sim <- mapply(function(a,b) rep(b, length(a)), predictions_sim, rsqrd_list_sim)
  rsqrd_list_adjusted <- mapply(function(a,b) rep(b, length(a)), predictions, rsqrd_list)
  #print(rsqrd_list_adjusted[[1]])

  alpha_list_adjusted_sim <- mapply(function(a,b) rep(b, length(a)), predictions_sim, alpha_list_sim)
  alpha_list_adjusted <- mapply(function(a,b) rep(b, length(a)), predictions_sim, alpha_list)
```

Part 8. Results Table (Table 1)

Finally, we put together our results from our previous calculations using a provided training and prediction period, and calculations using our resultant predictions and actual data. Please see output table below code explanations.

```
  # Creating final dataframe with all computations.
  df_final_calc <- df_prediction_temp %>%

    filter(TICKER %in% pred_firms_merge$TICKER) %>%
    arrange(TICKER) %>%

    mutate(Rsqrd_3FF = unlist(rsqrd_list_adjusted)) %>%
    mutate(Rsqrd_SIM = unlist(rsqrd_list_adjusted_sim)) %>%

    mutate(Prediction_SIM = unlist(predictions_sim)) %>%
    mutate(Prediction_3FF = unlist(predictions)) %>%

    mutate(alpha_SIM = unlist(alpha_list_adjusted_sim)) %>%
    mutate(alpha_3FF = unlist(alpha_list_adjusted)) %>%

    mutate(pred_err_SIM = exRET - Prediction_SIM) %>%
    mutate(pred_err_3FF = exRET - Prediction_3FF) %>%

    mutate(Abs_pred_err_SIM = abs(exRET - Prediction_SIM)) %>%
    mutate(Abs_pred_err_3FF = abs(exRET - Prediction_3FF)) %>%

    group_by(TICKER) %>%

    mutate(mean_abs_PredErr_SIM = mean(Abs_pred_err_SIM)) %>%
    mutate(mean_abs_PredErr_3FF = mean(Abs_pred_err_3FF)) %>%

    mutate(stdev_PredErr_SIM = sd(alpha_SIM)) %>%
    mutate(stdev_PredErr_3FF = sd(alpha_3FF)) %>%

    slice(1) %>%
    na.omit() %>%
    ungroup()

  df_final <- df_final_calc %>%
    select(TICKER, Rsqrd_SIM, Rsqrd_3FF, alpha_SIM, alpha_3FF, mean_abs_PredErr_SIM, mean_abs_PredErr_3FF, std
ev_PredErr_SIM, stdev_PredErr_3FF)
```

Part 9. High level Summary Table (Table 2)

This final table shows a very brief comparison of the two models. It shows what each of the metrics 'generally, were with each model. Most importantly, the Average mean prediction error with the Single Index Model and the Fama French 3 Factor Model can be seen. Please see output table in the next section.

```
 # Summary data to show how each model performed.

  df_summary <- df_final %>%
    mutate(Overall_mean_Rsqrd_SIM = mean(Rsqrd_SIM)) %>%
    mutate(Overall_mean_Rsqrd_3FF = mean(Rsqrd_3FF)) %>%

    mutate(Overall_mean_alpha_SIM = mean(alpha_SIM)) %>%
    mutate(Overall_mean_alpha_3FF = mean(alpha_3FF)) %>%

    mutate(Overall_mean_PredErr_SIM = mean(mean_abs_PredErr_SIM)) %>%
    mutate(Overall_mean_PredErr_3FF = mean(mean_abs_PredErr_3FF)) %>%

    mutate(Overall_mean_stdev_SIM = mean(stdev_PredErr_SIM)) %>%
    mutate(Overall_mean_stdev_3FF = mean(stdev_PredErr_3FF)) %>%

    select(Overall_mean_Rsqrd_SIM, Overall_mean_Rsqrd_3FF, Overall_mean_alpha_SIM, Overall_mean_alpha_3FF, Ove
rall_mean_PredErr_SIM, Overall_mean_PredErr_3FF, Overall_mean_stdev_SIM, Overall_mean_stdev_3FF) %>%
    slice(1)
  print(df_summary)
```

Part 10: Plot

At last, we plot a graph of Rsquared values with mean absolute prediction error for each model to see how each model performed in summary.

```
  ggplot(data = df_final) +
      xlab(label = "R Squared Values") +
      ylab(label = "Mean Absolute Prediction Error") +
      geom_smooth(mapping = aes(x = Rsqrd_SIM, y = mean_abs_PredErr_SIM, color = 'SIM'), se = FALSE) +
      geom_smooth(mapping = aes(x = Rsqrd_3FF, y = mean_abs_PredErr_3FF, color = '3FF'), se = FALSE) +
      scale_colour_manual(name="legend", values=c("red", "blue")) +
      ggtitle("R Squared - MAE plots for SIM and 3FF") +
      theme(plot.title = element_text(hjust = 0.5, size = 15))
```

D. Running the function for training period: 2005-2012, and prediction period: 2013-2014.

```
train <- c("20050101", "20121231")
predict <- c("20130101", "20141231")

reg_comparator("780 B2 Stocks daily.csv", "780 B2 FF3factors daily.csv", train, predict)
```

E. Output Tables and graph

Main Regressions Results Table:

```
##    TICKER Rsqrd_SIM Rsqrd_3FF     alpha_SIM     alpha_3FF mean_abs_PredErr_SIM
## 1    AGCO 0.5606347 0.5636243 -5.480215e-05 -6.737033e-05          0.009712666
## 2      BA 0.5057559 0.5093052  1.852264e-04  1.950099e-04          0.007586094
## 3    CBRL 0.3706981 0.4056002  3.090814e-04  2.620780e-04          0.007535199
## 4     CSX 0.5316105 0.5343011  5.497210e-04  5.378682e-04          0.006855816
## 5     CVS 0.3121715 0.3157975  3.585611e-04  3.573518e-04          0.006038173
## 6      DE 0.5422627 0.5438634  4.544212e-04  4.526723e-04          0.007228234
## 7       F 0.3148497 0.3257325  1.683712e-04  1.395135e-04          0.007665967
## 8     GBX 0.3753329 0.4347150  2.099290e-04  1.017922e-04          0.015920890
## 9    GILD 0.2688722 0.3057069  6.888383e-04  7.067553e-04          0.012156849
## 10   JNPR 0.3715988 0.3917548 -8.871996e-05 -8.908835e-05          0.012081730
## 11   LRCX 0.4117217 0.4415570  1.420710e-04  1.230525e-04          0.008941110
## 12   MSFT 0.5074037 0.5456865 -2.572053e-05  4.495850e-06          0.008201464
## 13   PDLI 0.1622776 0.1719799  2.821916e-04  2.627553e-04          0.011495730
## 14   SONC 0.3748614 0.4252885 -3.005848e-04 -3.589204e-04          0.010861239
## 15    STZ 0.2408307 0.2414387  2.072981e-04  2.071168e-04          0.009341593
## 16   SWHC 0.1580207 0.2000888  6.286751e-04  5.228387e-04          0.014459011
## 17   TSLA 0.1593696 0.1948052  4.105132e-04  4.334072e-04          0.022767321
## 18    TTM 0.3042547 0.3240517  6.818047e-04  6.385980e-04          0.014625870
## 19    XOM 0.5929569 0.6407324  2.251518e-04  2.625857e-04          0.005402276
##    mean_abs_PredErr_3FF stdev_PredErr_SIM stdev_PredErr_3FF
## 1           0.010056384       0.013457739       0.013816726
## 2           0.007588642       0.010612071       0.010656981
## 3           0.007496409       0.010944513       0.010933016
## 4           0.006802187       0.009895962       0.009766668
## 5           0.006131640       0.007815933       0.007960422
## 6           0.007425343       0.010224850       0.010452431
## 7           0.007698460       0.011258320       0.011153882
## 8           0.015759414       0.023429589       0.023545342
## 9           0.011547851       0.017064914       0.016379720
## 10          0.012485851       0.018082862       0.018327692
## 11          0.009389506       0.013439509       0.013707297
## 12          0.008217891       0.012386891       0.012357619
## 13          0.011375786       0.017467233       0.017356792
## 14          0.010918592       0.015078912       0.015293122
## 15          0.009341595       0.021579331       0.021570894
## 16          0.014448899       0.021080837       0.021308116
## 17          0.021822055       0.035251991       0.034121313
## 18          0.014488585       0.019713715       0.019703758
## 19          0.005520572       0.007203488       0.007513323
```

After cleansing all the data, performing regressions, and calculating all metrics, we arrive at these firms and their data which encapsulates all the results.

High Level Summary Table

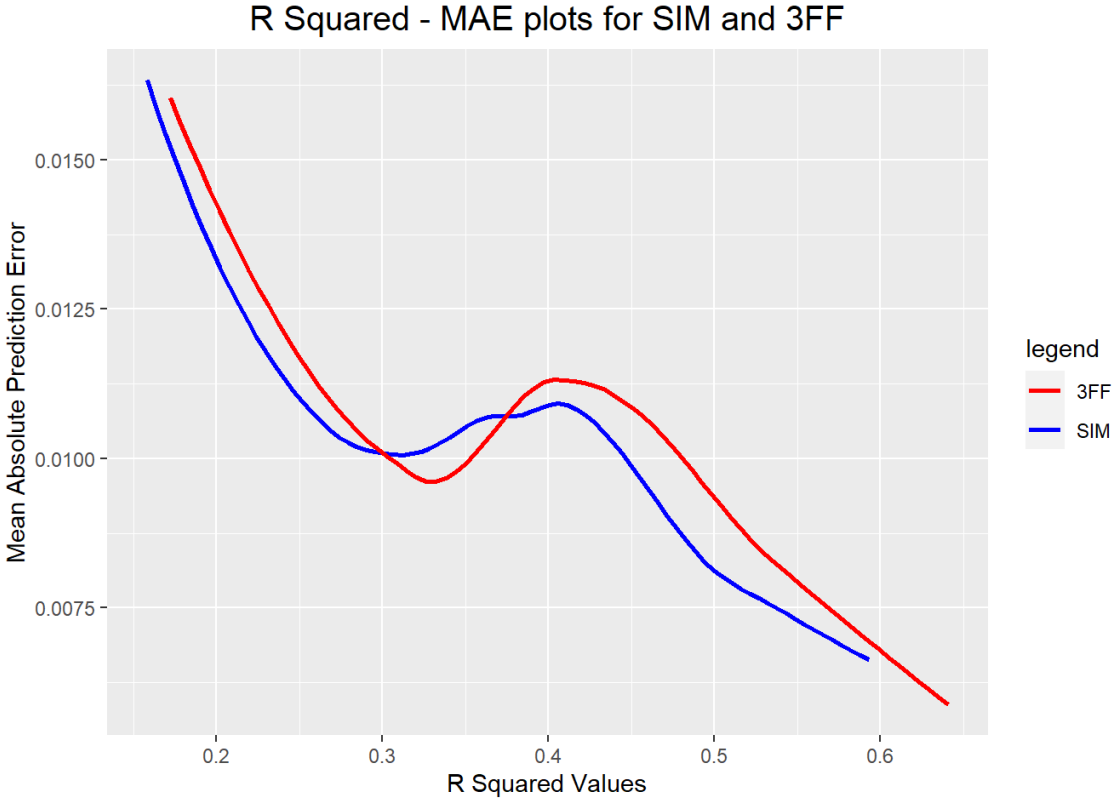
```
##   MeanRsqrdSIM MeanRsqrd3FF MeanAlphaSIM MeanAlpha3FF MeanPredErrSIM
## 1    0.3718676    0.3955805 0.0002648436 0.0002469743     0.01046722
##   MeanPredErr3FF MeanStDevSIM MeanStDev3FF
## 1     0.01044819   0.01557835   0.01557501
```

This table calculates means for All columns of table 1 to show a brief overview of what the two models generally produced. We can see that the mean prediction error for 3FF is slightly less than SIM across all firms generally.

Also, we noticed that the overall mean Rsquared value of 3FF is higher than SIM meaning it fitted the data better than SIM hence lesser prediction errors.

Graph Plot

From our main table, we plotted the Rsqrd values against mean prediction error for each firm. The firms are not shown because they are not important to our objective of this graph.



We see from this plot that for each model, the greater the Rsqrd value, lesser the mean prediction error as expected.

We notice that the Rsquared values for 3FF plot are generally larger leading to a plot displaced slightly to the right.

The mean prediction error for 3FF is also slightly less in “most” parts leading to a plot slightly displaced towards the bottom.

F. Conclusion

We can conclude that both models provide good predictions generally.

3FF does slightly better than SIM.

But as we saw from our initial graphs, the ExRET seemed to depend largely on Mkt.Rf (most significant) which both models incorporate, hence they have similar results.

FacSMB and FacHML dictate ExRET far less than Mkt.Rf so their inclusion in 3FF does improve the results but only slightly.

It is also possible that these two models would differ more for cases where FacSMB and FacHML have bigger roles to play. This process can be repeated for more cases to test further.