Econ 104 Project 2

Phoebe Green, Shannan Liu, Divyanshi Singh, Allison Young

5/24/2021

Table of Contents

Introduction 2

Load and Organise Data 2

- Align Dates, Create IDs, and Compute Daily Returns 2
- Create Panel Data Frame 5

Data Exploration 5

Pooled Model 9

Fixed Effects Model 13

- Least Squares Dummy Variable Estimator 13
- Fixed Effects (Within) Estimator 16

Random Effects Model 17

Sets of Regression Equations 18

Training and Testing Our Model 23

Forecasting 23

Future Work 26

References 27

Introduction

This project explores the Fama-French 3-factor asset pricing model, which is an extension of the capital asset pricing model (CAPM). The Fama-French model builds upon the CAPM's idea that the risk premium of a security is proportional to the risk premium of the market portfolio by proposing 2 more variables to consider: the size premium and value premium of companies.

The following equation represents the Fama-French model:

$$R_i - R_f = \alpha + \beta(R_m - R_f) + b_s SMB + b_v HML$$

- $R_m - R_f$ (Market Risk Premium): the return of the market portfolio minus the risk-free rate
- SMB (Size Premium): measures the historic excess returns of small cap companies over large cap companies
- HML (Value Premium): measures the historic excess returns of high book-to-market value ratio companies and companies with a low book-to-market value ratio

To explore the asset pricing model developed by Fama and French, the daily returns of four different tech companies' stocks were computed and regressed on the 3 asset pricing factors outlined above. In this project, we used 35 years of daily returns data from 1986 - 2021, and we chose to explore the following companies' stocks: IBM, Apple, Microsoft, and Intel.

The daily return data for IBM, Apple, Microsoft, and Intel was collected from Yahoo Finance (https://finance.yahoo.com/). The daily data on the 3 factors of the Fama-French model was sourced from Kenneth R. French's Data Library (https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html).

Load and Organise Data

```
rm(list=ls())
ff = read.csv("/Users/phoebegreen/Desktop/F-F_daily_clean.csv")
ibm = read.csv("/Users/phoebegreen/Desktop/IBM.csv")
apple = read.csv("/Users/phoebegreen/Desktop/AAPL.csv")
msft = read.csv("/Users/phoebegreen/Desktop/MSFT.csv")
intel = read.csv("/Users/phoebegreen/Desktop/INTC.csv")
```

Align Dates, Create IDs, and Compute Daily Returns

Each data frame that was loaded had a different number of rows, meaning that the individual data files contained misaligned dates and represented different time periods. Therefore, the first task was to organise the respective data frames so that they would all reflect data from the same time interval.

```
# Reformat Fama-French dates to something legible in R
# Set format the 4 stocks' Date columns as "Date" type
ff$Date <- as.Date(as.character(ff[["Date"]]), format = '%Y%m%d')
ibm$Date <- as.Date(ibm$Date)
apple$Date <- as.Date(apple$Date)
msft$Date <- as.Date(msft$Date)
intel$Date <- as.Date(intel$Date)

# The latest starting date is: "1986-03-13"
start_date = max(msft$Date[1],apple$Date[1],intel$Date[1],
                 ibm$Date[1],ff$Date[1])
```

```
# The earliest end date is: 2021-03-30
end_date = min(msft$Date[length(msft$Date)],
          apple$Date[length(apple$Date)],
          intel$Date[length(intel$Date)],
          ibm$Date[length(ibm$Date)],
          ff$Date[length(ff$Date)])

# Filter data frames to only have rows from mar-13-1986 -> mar-30-2021
ff_86 = filter(ff,ff$Date>=start_date,ff$Date<=end_date)
ibm_86 = filter(ibm,ibm$Date>=start_date,ibm$Date<=end_date)
apple_86 = filter(apple,apple$Date>=start_date,apple$Date<=end_date)
msft_86 = filter(msft,msft$Date>=start_date,msft$Date<=end_date)
intel_86 = filter(intel,intel$Date>=start_date,intel$Date<=end_date)
```

After the aligning the dates, we create the IDs for each company and compute the daily returns.

```
# create id column
ibm_86["id"] <- "IBM"
apple_86["id"] <- "APPLE"
msft_86["id"] <- "MSFT"
intel_86["id"] <- "INTEL"

ibm_86["daily_returns"] <- NA
apple_86["daily_returns"] <- NA
msft_86["daily_returns"] <- NA
intel_86["daily_returns"] <- NA

for (i in 1:(length(ibm_86$Adj.Close)-1)){
    ibm_86$daily_returns[i+1] =
      (ibm_86$Adj.Close[i+1] - ibm_86$Adj.Close[i])/(ibm_86$Adj.Close[i])
}

for (i in 1:(length(apple_86$Adj.Close)-1)){
    apple_86$daily_returns[i+1] =
      (apple_86$Adj.Close[i+1] - apple_86$Adj.Close[i])/(apple_86$Adj.Close[i])
}

for (i in 1:(length(msft_86$Adj.Close)-1)){
    msft_86$daily_returns[i+1] =
      (msft_86$Adj.Close[i+1] - msft_86$Adj.Close[i])/(msft_86$Adj.Close[i])
}

for (i in 1:(length(intel_86$Adj.Close)-1)){
    intel_86$daily_returns[i+1] =
      (intel_86$Adj.Close[i+1] - intel_86$Adj.Close[i])/(intel_86$Adj.Close[i])
}
```

Now we have IDs and daily returns for each company, we can prepare to organise the stocks into a panel data frame. To do so, we remove all columns from the stock data frames except for "id", "date", and "daily_returns". Then, for each stock, we will add new columns called "Mkt.RF", "SMB", "HML", and "RF", which are directly obtained from the Fama French data frame "ff_86".

```
# Only keep id, date, daily_returns columns for each stock
# ff columns except for "Date" are cbinded onto each stock's data frame
ibm_86_final = ibm_86[,(names(ibm_86) %in% c("id","Date","daily_returns"))]
apple_86_final = apple_86[,(names(apple_86) %in%
                              c("id","Date","daily_returns"))]
msft_86_final = msft_86[,(names(msft_86) %in% c("id","Date","daily_returns"))]
intel_86_final = intel_86[,(names(intel_86) %in%
                              c("id","Date","daily_returns"))]

# FF columns needed: "Mkt.RF" "SMB" "HML" "RF"
# IBM
ibm_86_final$Mkt_Minus_RF = ff_86$Mkt.RF
ibm_86_final$SMB = ff_86$SMB
ibm_86_final$HML = ff_86$HML
ibm_86_final$RF = ff_86$RF
ibm_86_final$Ri_minus_Rf = ibm_86_final$daily_returns - ff_86$RF
ibm_86_final <- na.omit(ibm_86_final)
# Check that all the right columns are present
kable(head(ibm_86_final))
```

	Date	id	daily_returns	Mkt_Minus_RF	SMB	HML	RF	Ri_minus_Rf
2	1986-03-14	IBM	-0.0008307	1.03	-0.84	-0.22	0.03	-0.0308307
3	1986-03-17	IBM	0.0033255	-0.75	0.02	-0.32	0.03	-0.0266745
4	1986-03-18	IBM	0.0099420	0.47	0.04	-0.17	0.03	-0.0200580
5	1986-03-19	IBM	-0.0049218	-0.17	0.13	-0.09	0.03	-0.0349218
6	1986-03-20	IBM	-0.0098933	0.39	-0.16	-0.02	0.03	-0.0398933
7	1986-03-21	IBM	-0.0074939	-0.83	1.69	0.68	0.03	-0.0374939

```
#APPLE
apple_86_final$Mkt_Minus_RF = ff_86$Mkt.RF
apple_86_final$SMB = ff_86$SMB
apple_86_final$HML = ff_86$HML
apple_86_final$RF = ff_86$RF
apple_86_final$Ri_minus_Rf = apple_86_final$daily_returns - ff_86$RF
apple_86_final<-na.omit(apple_86_final)
# Check that all the right columns are present
kable(head(apple_86_final))
```

	Date	id	daily_returns	Mkt_Minus_RF	SMB	HML	RF	Ri_minus_Rf
2	1986-03-14	APPLE	0.0555613	1.03	-0.84	-0.22	0.03	0.0255613
3	1986-03-17	APPLE	-0.0047842	-0.75	0.02	-0.32	0.03	-0.0347842
4	1986-03-18	APPLE	0.0336611	0.47	0.04	-0.17	0.03	0.0036611
5	1986-03-19	APPLE	-0.0139625	-0.17	0.13	-0.09	0.03	-0.0439625
6	1986-03-20	APPLE	0.0660414	0.39	-0.16	-0.02	0.03	0.0360414
7	1986-03-21	APPLE	-0.0221214	-0.83	1.69	0.68	0.03	-0.0521214

```
#MSFT
msft_86_final$Mkt_Minus_RF = ff_86$Mkt.RF
msft_86_final$SMB = ff_86$SMB
```

```
msft_86_final$HML = ff_86$HML
msft_86_final$RF = ff_86$RF
msft_86_final$Ri_minus_Rf = msft_86_final$daily_returns - ff_86$RF
msft_86_final<-na.omit(msft_86_final)
# Check that all the right columns are present
kable(head(msft_86_final))
```

	Date	id	daily_returns	Mkt_Minus_RF	SMB	HML	RF	Ri_minus_Rf
2	1986-03-14	MSFT	0.0357259	1.03	-0.84	-0.22	0.03	0.0057259
3	1986-03-17	MSFT	0.0172389	-0.75	0.02	-0.32	0.03	-0.0127611
4	1986-03-18	MSFT	-0.0254356	0.47	0.04	-0.17	0.03	-0.0554356
5	1986-03-19	MSFT	-0.0173891	-0.17	0.13	-0.09	0.03	-0.0473891
6	1986-03-20	MSFT	-0.0265453	0.39	-0.16	-0.02	0.03	-0.0565453
7	1986-03-21	MSFT	-0.0272691	-0.83	1.69	0.68	0.03	-0.0572691

```
# INTEL
intel_86_final$Mkt_Minus_RF = ff_86$Mkt.RF
intel_86_final$SMB = ff_86$SMB
intel_86_final$HML = ff_86$HML
intel_86_final$RF = ff_86$RF
intel_86_final$Ri_minus_Rf = intel_86_final$daily_returns - ff_86$RF
intel_86_final<-na.omit(intel_86_final)
# Check that all the right columns are present
kable(head(intel_86_final))
```

	Date	id	daily_returns	Mkt_Minus_RF	SMB	HML	RF	Ri_minus_Rf
2	1986-03-14	INTEL	-0.0353997	1.03	-0.84	-0.22	0.03	-0.0653997
3	1986-03-17	INTEL	0.0000000	-0.75	0.02	-0.32	0.03	-0.0300000
4	1986-03-18	INTEL	0.0091761	0.47	0.04	-0.17	0.03	-0.0208239
5	1986-03-19	INTEL	-0.0181825	-0.17	0.13	-0.09	0.03	-0.0481825
6	1986-03-20	INTEL	0.0185193	0.39	-0.16	-0.02	0.03	-0.0114807
7	1986-03-21	INTEL	-0.0090927	-0.83	1.69	0.68	0.03	-0.0390927

Create Panel Data Frame

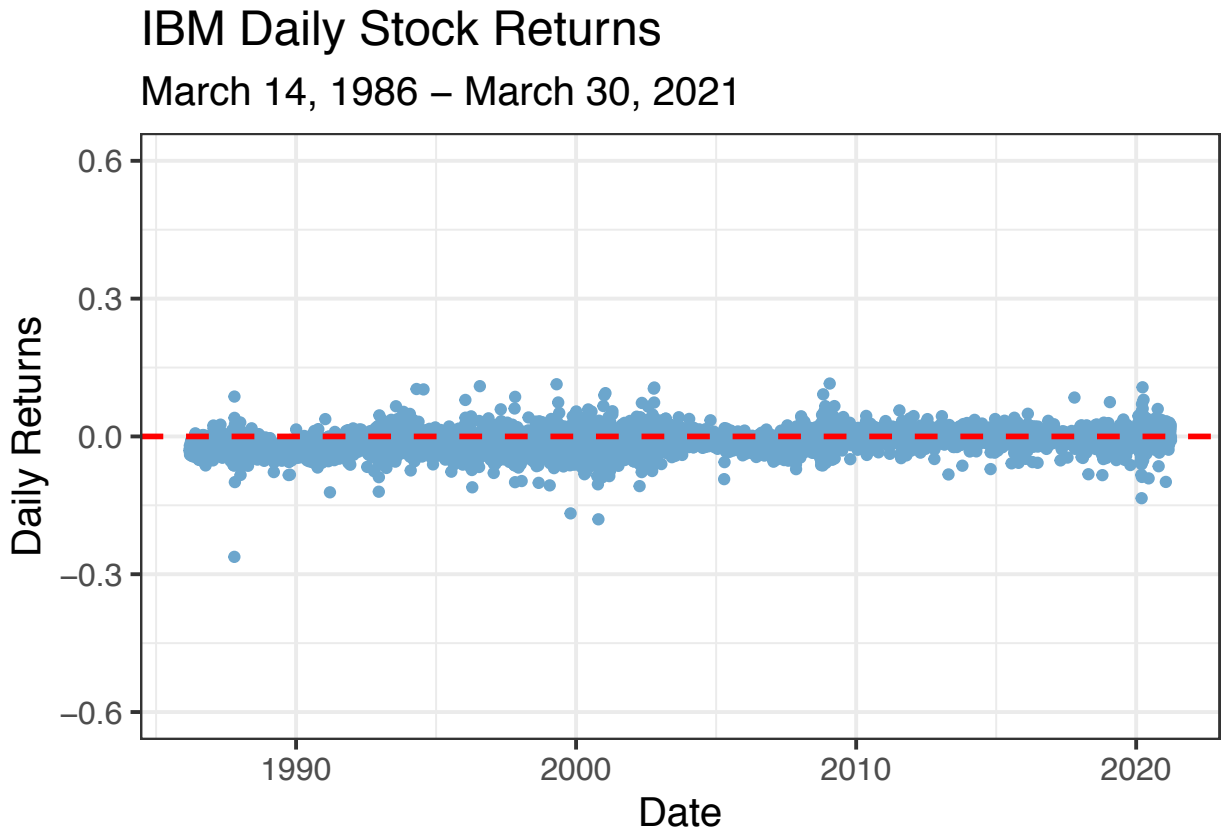
Finally, we can create our panel data frame by binding all the cleaned up data frames together.

```
panel_df <- bind_rows(ibm_86_final,apple_86_final,msft_86_final,intel_86_final)
# create a panel data structure
df.pd <- pdata.frame(panel_df, index=c("id", "Date"))
```

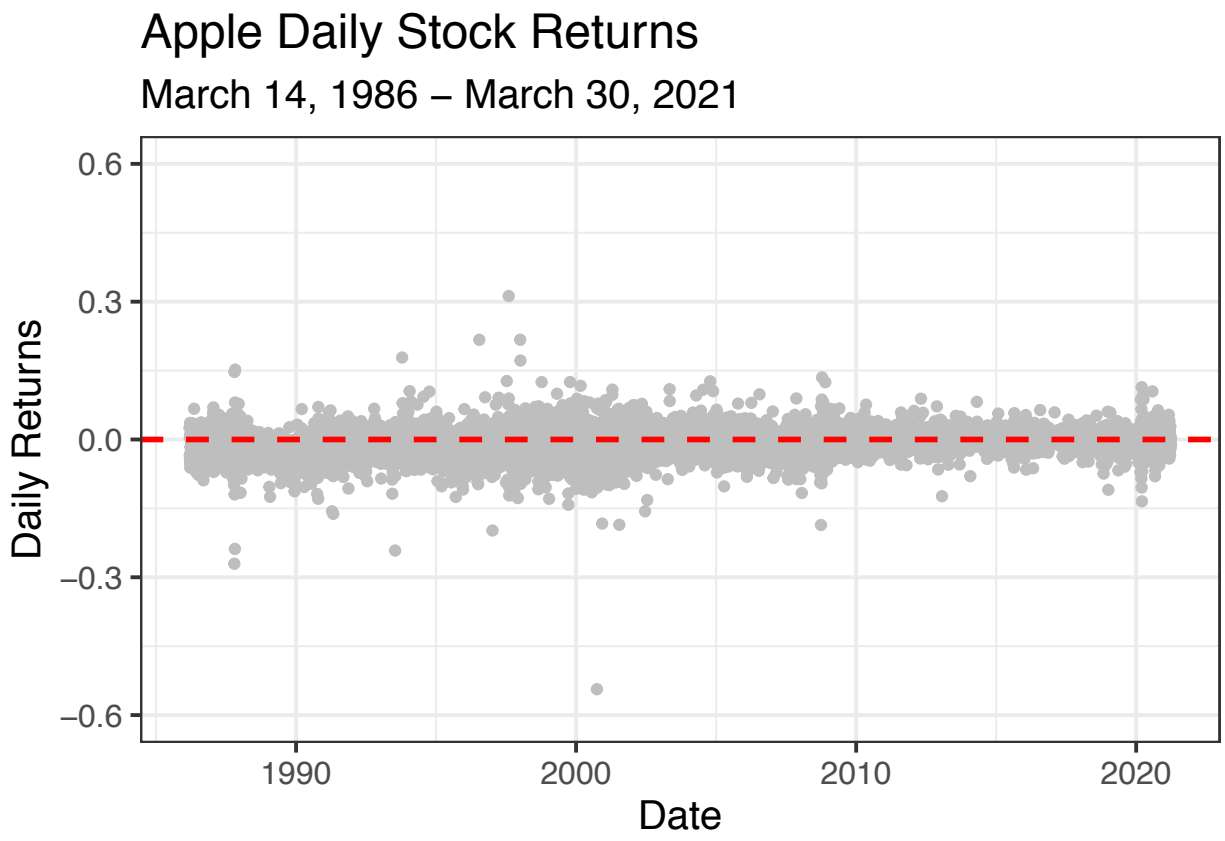
Data Exploration

The daily returns appear to exhibit stationarity. Meanwhile, the daily return on Apple's, Microsoft's, and Intel's stocks appear to exhibit similar amounts of volatility, whereas IBM's stock looks a little less volatile.

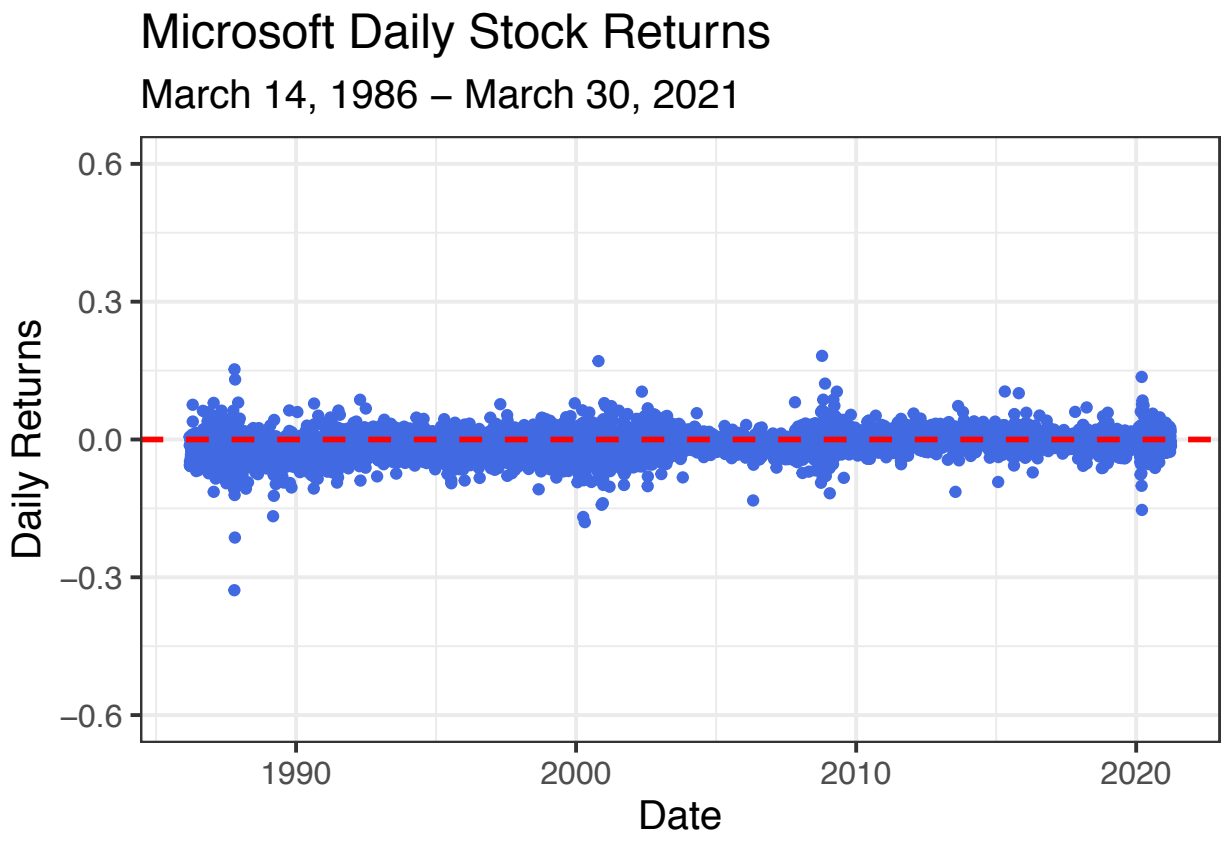
```r
# plot the data using ggplot2 and pipes
ibm_86_final %>%ggplot(aes(x = Date, y = Ri_minus_Rf)) +
  geom_point(color = "skyblue3") +
  ylim(-0.6,0.6) +
  labs(title = "IBM Daily Stock Returns",
       subtitle = "March 14, 1986 - March 30, 2021",
       y = "Daily Returns",
       x = "Date") + theme_bw(base_size = 15) +
  geom_hline(yintercept=0, linetype="dashed",
             color = "red", size=1)
```



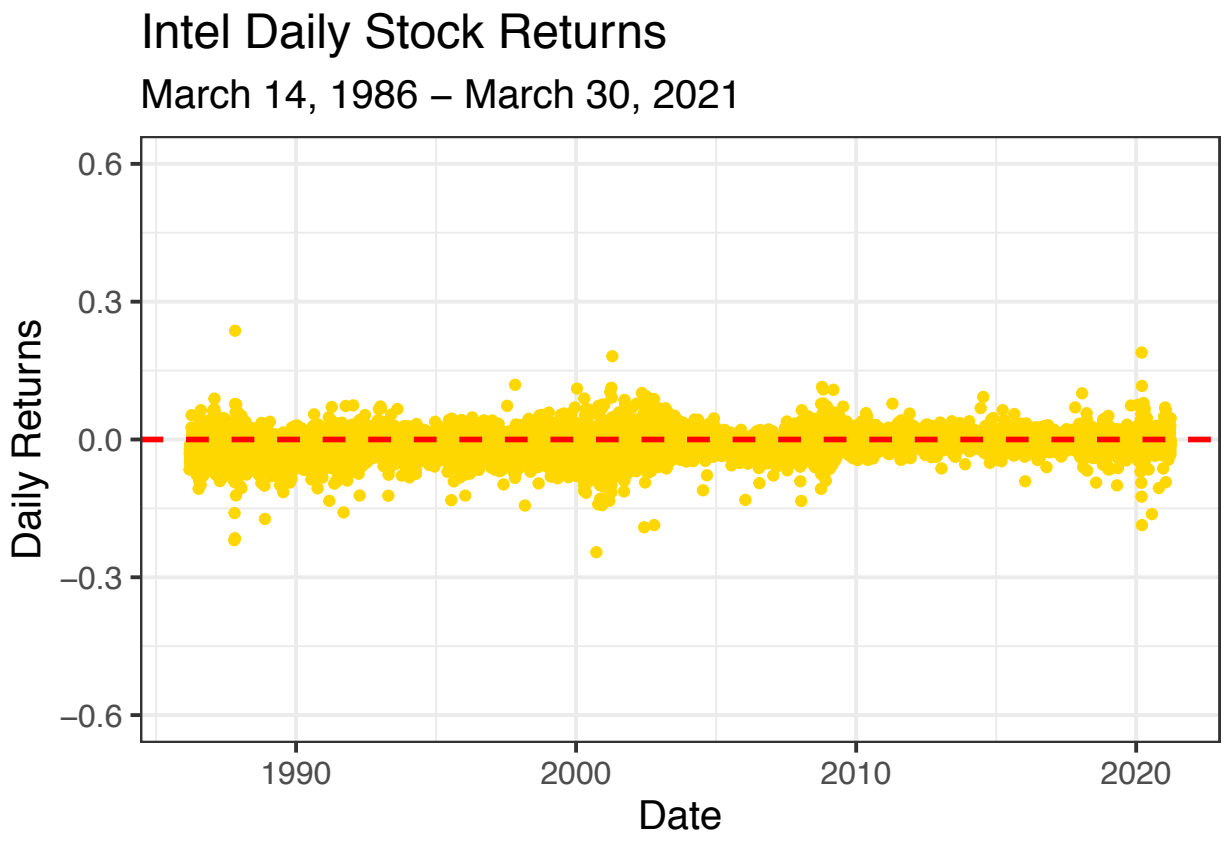

```r
apple_86_final %>%ggplot(aes(x = Date, y = Ri_minus_Rf)) +
  geom_point(color = "grey") +
  ylim(-0.6,0.6) +
  labs(title = "Apple Daily Stock Returns",
       subtitle = "March 14, 1986 - March 30, 2021",
       y = "Daily Returns",
       x = "Date") + theme_bw(base_size = 15) +
  geom_hline(yintercept=0, linetype="dashed",
             color = "red", size=1)
```




```
msft_86_final %>%ggplot(aes(x = Date, y = Ri_minus_Rf)) +
  geom_point(color = "royalblue") +
  ylim(-0.6,0.6) +
  labs(title = "Microsoft Daily Stock Returns",
       subtitle = "March 14, 1986 - March 30, 2021",
       y = "Daily Returns",
       x = "Date") + theme_bw(base_size = 15) +
  geom_hline(yintercept=0, linetype="dashed",
             color = "red", size=1)
```




```
intel_86_final %>%ggplot(aes(x = Date, y = Ri_minus_Rf)) +
  geom_point(color = "gold") +
  ylim(-0.6,0.6) +
  labs(title = "Intel Daily Stock Returns",
       subtitle = "March 14, 1986 - March 30, 2021",
       y = "Daily Returns",
       x = "Date") + theme_bw(base_size = 15) +
  geom_hline(yintercept=0, linetype="dashed",
             color = "red", size=1)
```



Next, we move on to building our models and analysing the data.

Pooled Model

```
#pooled model
df.pd$ri_minus_rf <- df.pd$daily_returns - df.pd$RF
pooledmodel <- plm(ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, type = "pooling",
                   data = df.pd)
summary(pooledmodel)
```

```
## Oneway (individual) effect Within Model
##
## Call:
## plm(formula = ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, data = df.pd,
##     type = "pooling")
##
## Balanced Panel: n = 4, T = 8833, N = 35332
##
## Residuals:
##       Min.    1st Qu.     Median    3rd Qu.       Max.
## -0.5096635 -0.0113160  0.0018115  0.0117039  0.3146335
##
## Coefficients:
##                 Estimate  Std. Error  t-value  Pr(>|t|)
## Mkt_Minus_RF  0.01160493  0.00009752 119.0010 < 2.2e-16 ***
```

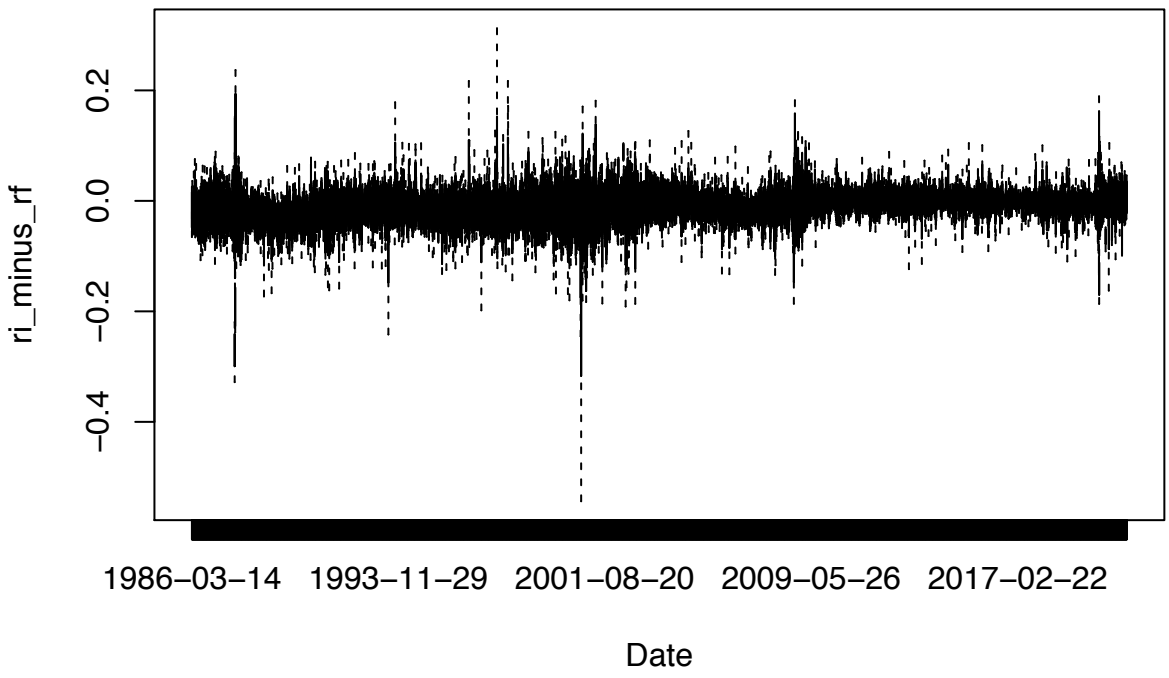
```
## SMB          -0.00111016  0.00018416  -6.0281 1.675e-09 ***
## HML          -0.00544110  0.00017429 -31.2192 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    22.356
## Residual Sum of Squares: 15.469
## R-Squared:      0.30804
## Adj. R-Squared: 0.30793
## F-statistic: 5241.95 on 3 and 35325 DF, p-value: < 2.22e-16
```

```
#with cluster-robust standard errors
coeftest(pooledmodel, vcov=vcovHC(pooledmodel, type="HC0",cluster="group"))
```

```
##
## t test of coefficients:
##
##                 Estimate  Std. Error t value  Pr(>|t|)
## Mkt_Minus_RF  0.01160493  0.00073457 15.7982 < 2.2e-16 ***
## SMB          -0.00111016  0.00053620 -2.0704   0.03842 *
## HML          -0.00544110  0.00111243 -4.8912 1.007e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now, we examine heterogeneity across time and subgroup.

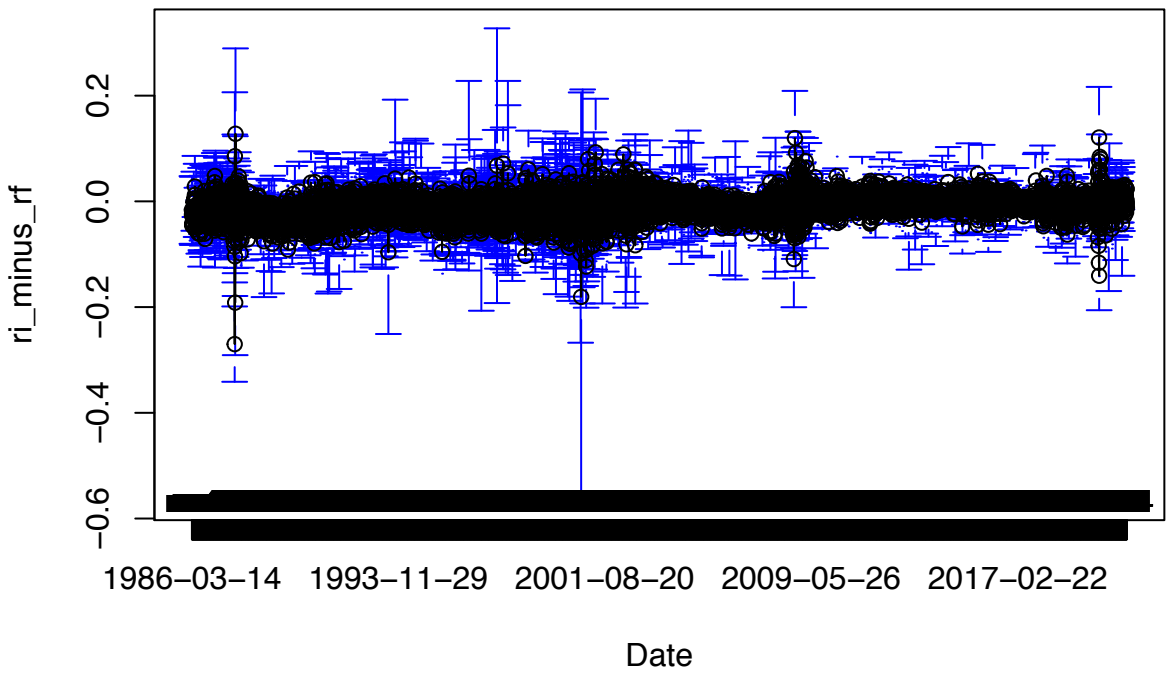
```
scatterplot(ri_minus_rf ~ Date|id, data = df.pd)
```



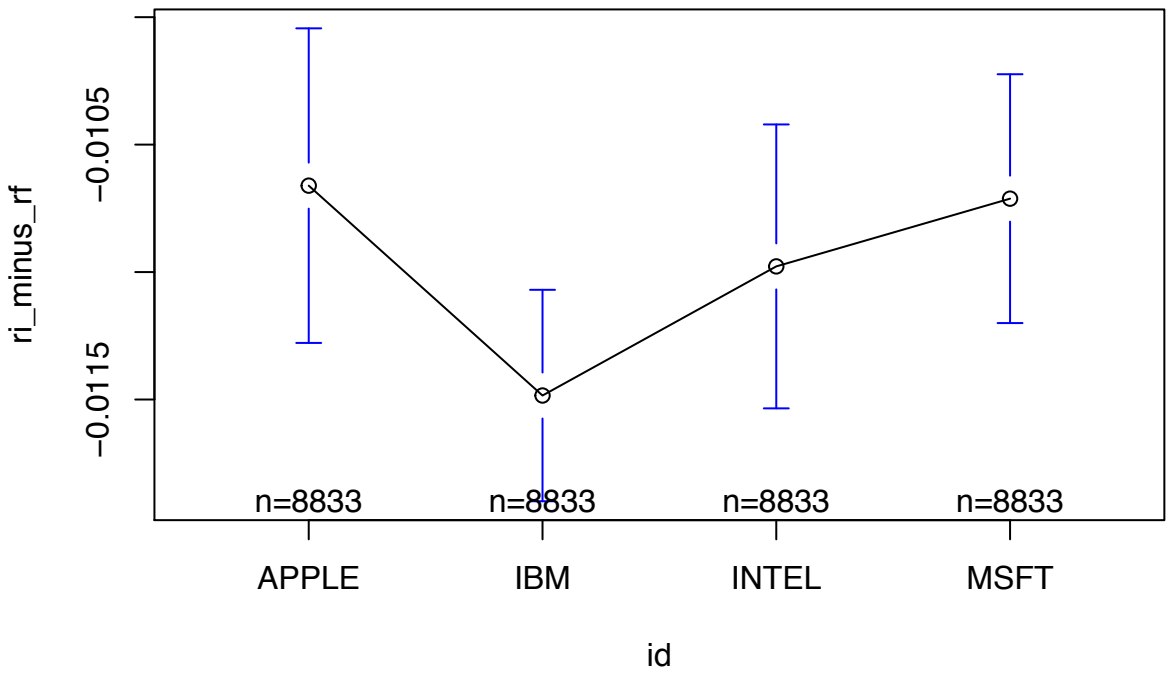

```
library(gplots)
```

```
##
## Attaching package: 'gplots'

## The following object is masked from 'package:tis':
##
##     barplot2

## The following object is masked from 'package:stats':
##
##     lowess
```

```
plotmeans(ri_minus_rf ~ Date, data = df.pd)
```




```
#across companies
plotmeans(ri_minus_rf ~ id, data = df.pd)
```



There is evidence of heterogeneity across date and across companies (id).

So, we can try a fixed effects model to compare the results.

Fixed effects Model

We will attempt both the least squares dummy variable estimator as well as the fixed effects (within) estimator.

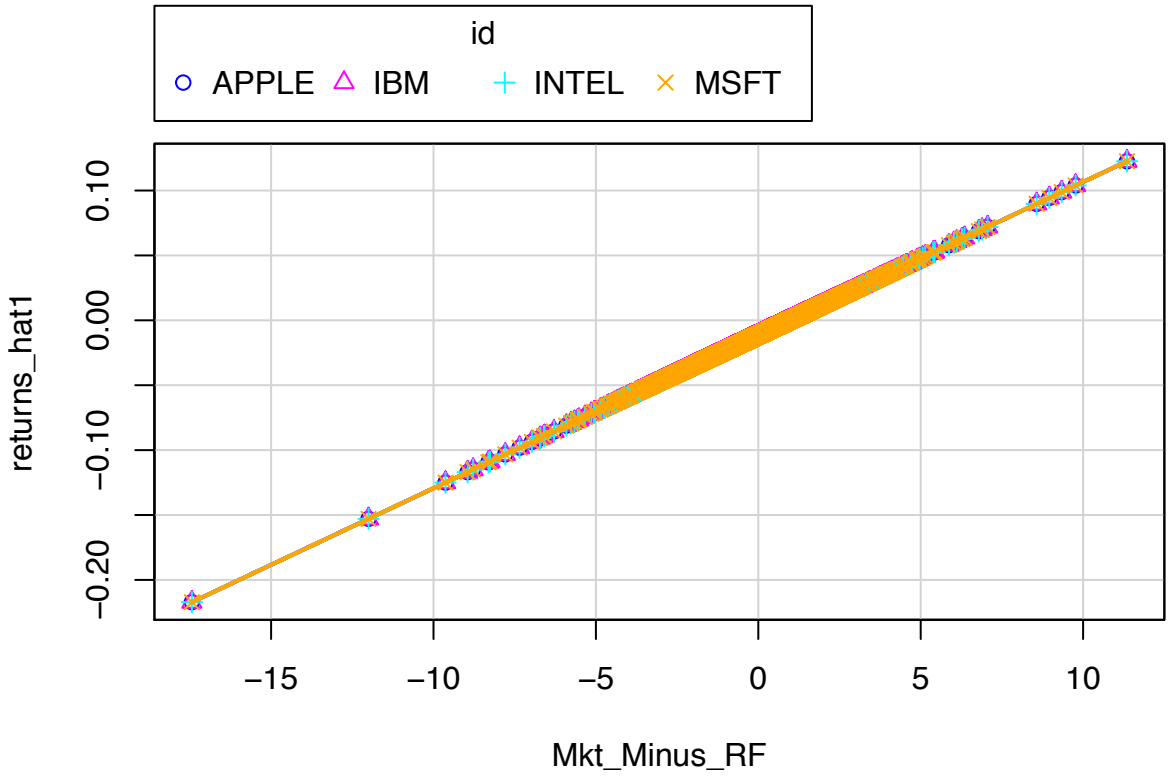
Least Squares Dummy Variable Estimator

```
fixed.dummy <- lm(ri_minus_rf ~ Mkt_Minus_RF + SMB + HML + factor(id)-1,
                  data = df.pd)
summary(fixed.dummy)
```

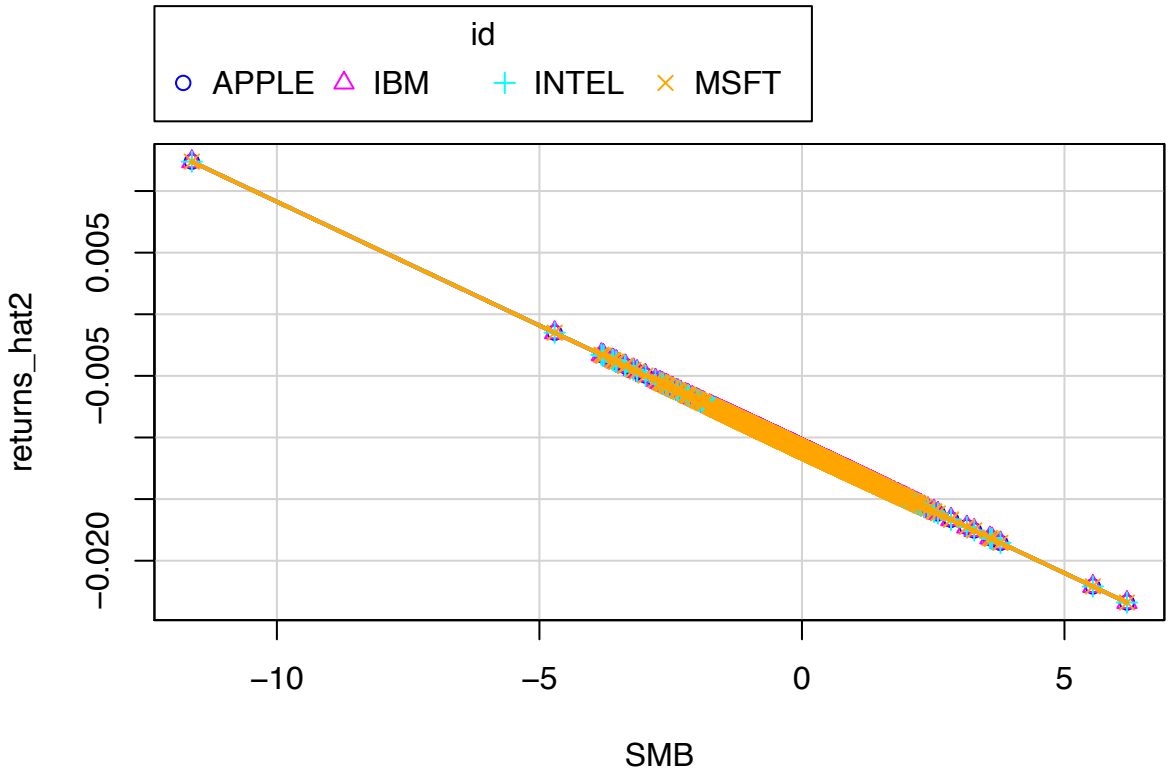
```
##
## Call:
## lm(formula = ri_minus_rf ~ Mkt_Minus_RF + SMB + HML + factor(id) -
##     1, data = df.pd)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.50966 -0.01132  0.00181  0.01170  0.31463
##
```

```
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## Mkt_Minus_RF     1.160e-02  9.752e-05 119.001  < 2e-16 ***
## SMB             -1.110e-03  1.842e-04  -6.028 1.68e-09 ***
## HML             -5.441e-03  1.743e-04 -31.219  < 2e-16 ***
## factor(id)APPLE -1.104e-02  2.227e-04 -49.559  < 2e-16 ***
## factor(id)IBM   -1.186e-02  2.227e-04 -53.257  < 2e-16 ***
## factor(id)INTEL -1.135e-02  2.227e-04 -50.982  < 2e-16 ***
## factor(id)MSFT  -1.109e-02  2.227e-04 -49.788  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02093 on 35325 degrees of freedom
## Multiple R-squared:  0.4185, Adjusted R-squared:  0.4184
## F-statistic:  3632 on 7 and 35325 DF,  p-value: < 2.2e-16
```

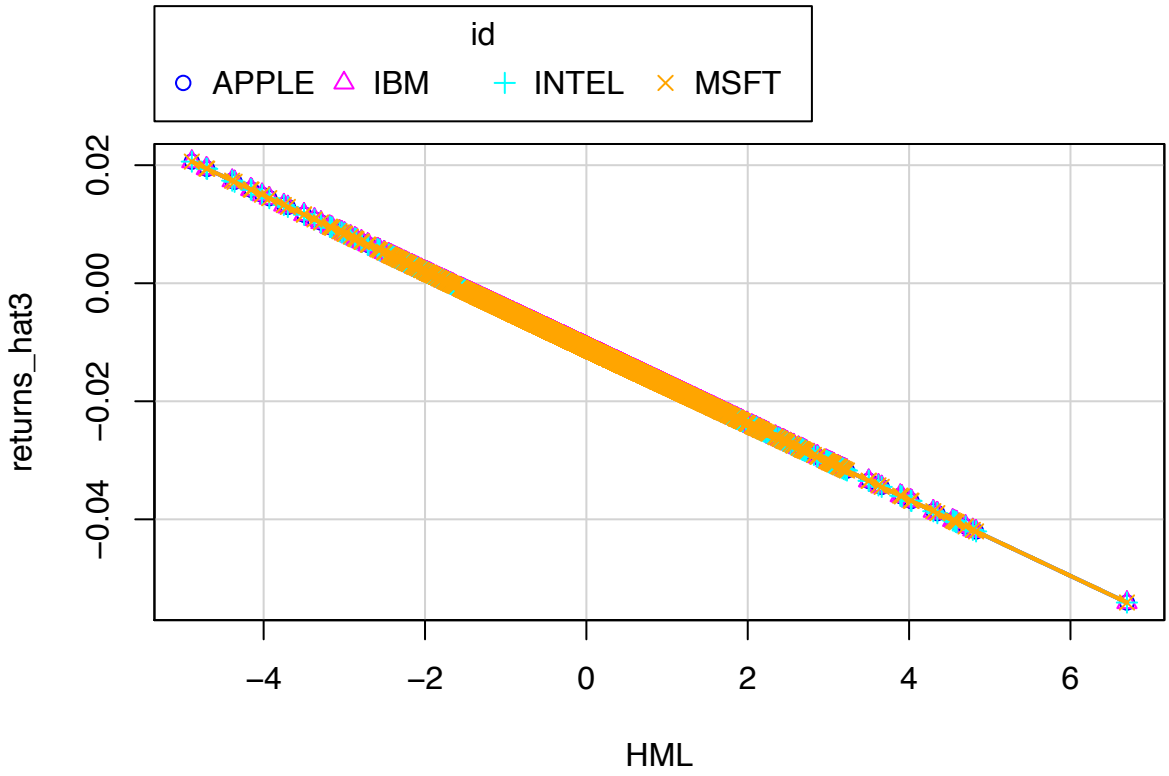
```
#plots to see dummy variable effect across companies
dummy.mkt <- lm(ri_minus_rf ~ Mkt_Minus_RF, data = df.pd)
returns_hat1 <- fitted.values(dummy.mkt)
scatterplot(returns_hat1 ~ Mkt_Minus_RF|id, data = df.pd)
```




```
dummy.smb <- lm(ri_minus_rf ~ SMB, data = df.pd)
returns_hat2 <- fitted.values(dummy.smb)
scatterplot(returns_hat2 ~ SMB|id, data = df.pd)
```




```
dummy.hml <- lm(ri_minus_rf ~ HML, data = df.pd)
returns_hat3 <- fitted.values(dummy.hml)
scatterplot(returns_hat3 ~ HML|id, data = df.pd)
```



In the plots above, there are not many visible dummy variable effects but this might be because finer differences are not as visible in a plot that spans such a large dataset.

Fixed Effects (Within) Estimator

```
fixed.within <- plm(ri_minus_rf ~ Mkt_Minus_RF + SMB + HML,
                    data = df.pd, model = "within")
summary(fixed.within)
```

```
## Oneway (individual) effect Within Model
##
## Call:
## plm(formula = ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, data = df.pd,
##     model = "within")
##
## Balanced Panel: n = 4, T = 8833, N = 35332
##
## Residuals:
##       Min.    1st Qu.     Median    3rd Qu.       Max.
## -0.5096635 -0.0113160  0.0018115  0.0117039  0.3146335
##
## Coefficients:
##                 Estimate  Std. Error  t-value  Pr(>|t|)
## Mkt_Minus_RF  0.01160493  0.00009752 119.0010 < 2.2e-16 ***
## SMB          -0.00111016  0.00018416  -6.0281 1.675e-09 ***
```

```
## HML          -0.00544110  0.00017429 -31.2192 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    22.356
## Residual Sum of Squares: 15.469
## R-Squared:      0.30804
## Adj. R-Squared: 0.30793
## F-statistic: 5241.95 on 3 and 35325 DF, p-value: < 2.22e-16
```

```
#extract fixed effects
fixef(fixed.within)
```

```
##     APPLE       IBM     INTEL      MSFT
## -0.011036 -0.011860 -0.011353 -0.011087
```

From the fixed effect estimation, it seems there might be significant individual (id) effects.

To confirm, we carry out the following test to compare the pooled and fixed effects models.

```
pooled.ols <-lm(ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, data=df.pd)
#using lm version of pooled model as plm version didn't work with pFtest
pFtest(fixed.within, pooled.ols)
```

```
##
##  F test for individual effects
##
## data:  ri_minus_rf ~ Mkt_Minus_RF + SMB + HML
## F = 2.8653, df1 = 3, df2 = 35325, p-value = 0.03519
## alternative hypothesis: significant effects
```

The test returns a p-value = 0.035, so we conclude there are significant fixed effects i.e. fixed effects model is better than the pooled model for this data.

Random Effects Model

```
randommodel <- plm(ri_minus_rf ~ Mkt_Minus_RF + SMB + HML,
                   data = df.pd, model = "within")
summary(randommodel)
```

```
## Oneway (individual) effect Within Model
##
## Call:
## plm(formula = ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, data = df.pd,
##     model = "within")
##
## Balanced Panel: n = 4, T = 8833, N = 35332
##
## Residuals:
##       Min.    1st Qu.     Median    3rd Qu.       Max.
## -0.5096635 -0.0113160  0.0018115  0.0117039  0.3146335
```

```
##
## Coefficients:
##                 Estimate  Std. Error  t-value  Pr(>|t|)
## Mkt_Minus_RF  0.01160493  0.00009752 119.0010 < 2.2e-16 ***
## SMB          -0.00111016  0.00018416  -6.0281 1.675e-09 ***
## HML          -0.00544110  0.00017429 -31.2192 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Total Sum of Squares:    22.356
## Residual Sum of Squares: 15.469
## R-Squared:      0.30804
## Adj. R-Squared: 0.30793
## F-statistic: 5241.95 on 3 and 35325 DF, p-value: < 2.22e-16
```

The results of the random effects model are identical to that of the fixed effects model. So, it is redundant to compare them.

Thus, we conclude that: of all the models estimated above, the fixed effects model seems best suited for our data.

However, since our panel is long and narrow, a model that utilizes sets of regression equations might be more accurate. So, we estimate that next.

Sets of Regression Equations

```
SURmod <- systemfit(ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, data = df.pd, method = "SUR")
summary(SURmod)
```

```
##
## systemfit results
## method: SUR
##
##            N    DF     SSR detRCov   OLS-R2 McElroy-R2
## system 35332 35316 15.2743       0 0.316759   0.199251
##
##          N   DF     SSR      MSE     RMSE       R2   Adj R2
## APPLE 8833 8829 5.77759 0.000654 0.025581 0.252618 0.252364
## IBM   8833 8829 2.45498 0.000278 0.016675 0.296886 0.296647
## INTEL 8833 8829 4.04179 0.000458 0.021396 0.358133 0.357915
## MSFT  8833 8829 2.99997 0.000340 0.018433 0.379756 0.379545
##
## The covariance matrix of the residuals used for estimation
##             APPLE         IBM       INTEL        MSFT
## APPLE 0.000654388 0.000123340 0.000174258 0.000130894
## IBM   0.000123340 0.000278058 0.000125001 0.000105197
## INTEL 0.000174258 0.000125001 0.000457786 0.000157922
## MSFT  0.000130894 0.000105197 0.000157922 0.000339786
##
## The covariance matrix of the residuals
##             APPLE         IBM       INTEL        MSFT
## APPLE 0.000654388 0.000123340 0.000174258 0.000130894
## IBM   0.000123340 0.000278058 0.000125001 0.000105197
```

```
## INTEL 0.000174258 0.000125001 0.000457786 0.000157922
## MSFT  0.000130894 0.000105197 0.000157922 0.000339786
##
## The correlations of the residuals
##          APPLE      IBM    INTEL     MSFT
## APPLE 1.000000 0.289146 0.318379 0.277588
## IBM   0.289146 1.000000 0.350359 0.342240
## INTEL 0.318379 0.350359 1.000000 0.400414
## MSFT  0.277588 0.342240 0.400414 1.000000
##
##
## SUR estimates for 'APPLE' (equation 1)
## Model Formula: APPLE_ri_minus_rf ~ APPLE_Mkt_Minus_RF + APPLE_SMB + APPLE_HML
## <environment: 0x7fa03f46b190>
##
##                  Estimate   Std. Error   t value Pr(>|t|)
## (Intercept)  -0.011042289  0.000272340 -40.54593  < 2e-16 ***
## Mkt_Minus_RF  0.012116354  0.000238422  50.81886  < 2e-16 ***
## SMB           0.000521475  0.000450254   1.15818  0.24682
## HML          -0.007297708  0.000426109 -17.12639  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.025581 on 8829 degrees of freedom
## Number of observations: 8833 Degrees of Freedom: 8829
## SSR: 5.777594 MSE: 0.000654 Root MSE: 0.025581
## Multiple R-Squared: 0.252618 Adjusted R-Squared: 0.252364
##
##
## SUR estimates for 'IBM' (equation 2)
## Model Formula: IBM_ri_minus_rf ~ IBM_Mkt_Minus_RF + IBM_SMB + IBM_HML
## <environment: 0x7fa03f46b190>
##
##                  Estimate   Std. Error   t value   Pr(>|t|)
## (Intercept)  -0.011802533  0.000177526 -66.48339 < 2.22e-16 ***
## Mkt_Minus_RF  0.009263845  0.000155417  59.60654 < 2.22e-16 ***
## SMB          -0.001827122  0.000293500  -6.22528 5.0259e-10 ***
## HML          -0.001645634  0.000277761  -5.92465 3.2469e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.016675 on 8829 degrees of freedom
## Number of observations: 8833 Degrees of Freedom: 8829
## SSR: 2.454977 MSE: 0.000278 Root MSE: 0.016675
## Multiple R-Squared: 0.296886 Adjusted R-Squared: 0.296647
##
##
## SUR estimates for 'INTEL' (equation 3)
## Model Formula: INTEL_ri_minus_rf ~ INTEL_Mkt_Minus_RF + INTEL_SMB + INTEL_HML
## <environment: 0x7fa03f46b190>
##
##                  Estimate   Std. Error   t value Pr(>|t|)
## (Intercept)  -0.011408238  0.000227785 -50.08335  < 2e-16 ***
## Mkt_Minus_RF  0.013299976  0.000199416  66.69456  < 2e-16 ***
```

```
## SMB          -0.000858381  0.000376592  -2.27934 0.022671 *
## HML          -0.006226551  0.000356397 -17.47083  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.021396 on 8829 degrees of freedom
## Number of observations: 8833 Degrees of Freedom: 8829
## SSR: 4.041789 MSE: 0.000458 Root MSE: 0.021396
## Multiple R-Squared: 0.358133 Adjusted R-Squared: 0.357915
##
##
## SUR estimates for 'MSFT' (equation 4)
## Model Formula: MSFT_ri_minus_rf ~ MSFT_Mkt_Minus_RF + MSFT_SMB + MSFT_HML
## <environment: 0x7fa03f46b190>
##
##                  Estimate   Std. Error   t value   Pr(>|t|)
## (Intercept)  -0.011083470  0.000196244 -56.47799 < 2.22e-16 ***
## Mkt_Minus_RF  0.011739557  0.000171803  68.33134 < 2.22e-16 ***
## SMB          -0.002276595  0.000324446  -7.01686 2.4363e-12 ***
## HML          -0.006594524  0.000307047 -21.47723 < 2.22e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.018433 on 8829 degrees of freedom
## Number of observations: 8833 Degrees of Freedom: 8829
## SSR: 2.999966 MSE: 0.00034 Root MSE: 0.018433
## Multiple R-Squared: 0.379756 Adjusted R-Squared: 0.379545
```

```r
#LM Test for contemporaneous correlation
teststat <- 8834*(0.28914 * 0.318379 * 0.277588 * 0.350359 * 0.342240 * 0.400414)
dof <- 4 * 3 / 2
qchisq(0.95, dof)
```

```
## [1] 12.59159
```

There is some low contemporaneous correlation between the residuals of the different companies, with the highest correlation being 0.400414 between Microsoft and Intel. We perform an LM Test to if the correlation between the errors is significantly different from zero. The 5% critical value of the χ^2 distribution with 6 degrees of freedom is 12.59159. The test-statistic is T times the sum of squares of all the correlations, which is 10.83837. Thus, we fail to reject the null hypothesis that there is no correlation. We conclude that OLS estimations of the regression equations are just as good as the SUR technique. Upon running separate OLS regressions, we can see that the SUR model and the separate OLS regressions produce the same results (below).

All four companies have similar intercept estimates which are approximately -0.01. In all of the equations, the coefficients for *MKT_MINUS_RF* are positive, the coefficients for HML are negative, and the significant coefficients for SMB are negative. All the coefficients are significant, except for the coefficient for SMB in the equation for Apple.

The R^2 for the four equations are quite low, ranging from 0.252618 to 0.358133. The R^2 for each company, in ascending order, are: Apple with 0.252618, IBM with 0.296886, Intel with 0.358133, and Microsoft with 0.379756.

```r
dummypd <- df.pd
dummypd$apple <- 0
dummypd$ibm <- 0
dummypd$intel <- 0
dummypd$microsoft <- 0
dummypd$apple[dummypd$id == "APPLE"] <- 1
dummypd$ibm[dummypd$id == "IBM"] <- 1
dummypd$intel[dummypd$id == "INTEL"] <- 1
dummypd$microsoft[dummypd$id == "MSFT"] <- 1

applemod <- lm(ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, data=dummypd[dummypd$apple == 1,])
summary(applemod)
```

```
##
## Call:
## lm(formula = ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, data = dummypd[dummypd$apple ==
##     1, ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.50732 -0.01351  0.00187  0.01378  0.31461
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.0110423  0.0002723 -40.546   <2e-16 ***
## Mkt_Minus_RF  0.0121164  0.0002384  50.819   <2e-16 ***
## SMB           0.0005215  0.0004503   1.158    0.247
## HML          -0.0072977  0.0004261 -17.126   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02558 on 8829 degrees of freedom
## Multiple R-squared:  0.2526, Adjusted R-squared:  0.2524
## F-statistic: 994.7 on 3 and 8829 DF,  p-value: < 2.2e-16
```

```r
intelmod <- lm(ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, data=dummypd[dummypd$intel == 1,])
summary(intelmod)
```

```
##
## Call:
## lm(formula = ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, data = dummypd[dummypd$intel ==
##     1, ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.23984 -0.01169  0.00190  0.01246  0.16806
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.0114082  0.0002278 -50.083   <2e-16 ***
## Mkt_Minus_RF  0.0133000  0.0001994  66.695   <2e-16 ***
## SMB          -0.0008584  0.0003766  -2.279   0.0227 *
```

```
## HML          -0.0062266  0.0003564 -17.471   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0214 on 8829 degrees of freedom
## Multiple R-squared:  0.3581, Adjusted R-squared:  0.3579
## F-statistic:  1642 on 3 and 8829 DF,  p-value: < 2.2e-16
```

```
ibmmod <- lm(ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, data=dummypd[dummypd$ibm == 1,])
summary(ibmmod)
```

```
##
## Call:
## lm(formula = ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, data = dummypd[dummypd$ibm ==
##     1, ])
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.160307 -0.010364  0.001772  0.010541  0.112379
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.0118025  0.0001775 -66.483  < 2e-16 ***
## Mkt_Minus_RF  0.0092638  0.0001554  59.607  < 2e-16 ***
## SMB          -0.0018271  0.0002935  -6.225 5.03e-10 ***
## HML          -0.0016456  0.0002778  -5.925 3.25e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01668 on 8829 degrees of freedom
## Multiple R-squared:  0.2969, Adjusted R-squared:  0.2966
## F-statistic:  1243 on 3 and 8829 DF,  p-value: < 2.2e-16
```

```
microsoftmod <- lm(ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, data=dummypd[dummypd$microsoft == 1,])
summary(microsoftmod)
```

```
##
## Call:
## lm(formula = ri_minus_rf ~ Mkt_Minus_RF + SMB + HML, data = dummypd[dummypd$microsoft ==
##     1, ])
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.154826 -0.009789  0.001895  0.010828  0.122288
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.0110835  0.0001962 -56.478  < 2e-16 ***
## Mkt_Minus_RF  0.0117396  0.0001718  68.331  < 2e-16 ***
## SMB          -0.0022766  0.0003244  -7.017 2.44e-12 ***
## HML          -0.0065945  0.0003070 -21.477  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.01843 on 8829 degrees of freedom
## Multiple R-squared:  0.3798, Adjusted R-squared:  0.3795
## F-statistic:  1802 on 3 and 8829 DF,  p-value: < 2.2e-16
```

Training and Testing Our Model

```
# splitting data into train and test sets
trainindex <- createDataPartition(y = df.pd$Ri_minus_Rf, p = .75,list = FALSE)
training <- df.pd[ trainindex,]
testing <- df.pd[-trainindex,]

# train model
trainmod <- train(Ri_minus_Rf ~ Mkt_Minus_RF + SMB + HML, data = training, method = "lm")

# predict values using the testing data & pooled model
testing$pred <- predict(trainmod, newdata = testing)

# evaluate the performance of the model
trainmod
```

```
## Linear Regression
##
## 26500 samples
##     3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 26500, 26500, 26500, 26500, 26500, 26500, ...
## Resampling results:
##
##   RMSE        Rsquared   MAE
##   0.02102582  0.3086514  0.01504421
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Since we are using panel data, we had some errors with the training and testing of our model. The train function doesn't allow for a plm or pooled model type, so instead we used a linear model which isn't the most accurate way to model panel data, and thus our predictions are not very specific or accurate. This results in our relatively low R^2 of 0.304. However, the RMSE is fairly low (0.021) at least for the training model. Additionally, our confusion matrix would not generate because of the size of our data, so it was difficult to fully evaluate the performance of our trained model on the test data.

Forecasting

```
# forecast 10 days into the future
forecast(apple_86_final$Ri_minus_Rf, 10)
```

```
##      Point Forecast       Lo 80      Hi 80       Lo 95      Hi 95
## 8834   0.0002498795 -0.03570517 0.03620492 -0.05473862 0.05523837
```

```
## 8835   0.0002498795 -0.03570819 0.03620795 -0.05474324 0.05524300
## 8836   0.0002498795 -0.03571121 0.03621097 -0.05474786 0.05524762
## 8837   0.0002498795 -0.03571423 0.03621399 -0.05475249 0.05525225
## 8838   0.0002498795 -0.03571726 0.03621702 -0.05475711 0.05525687
## 8839   0.0002498795 -0.03572028 0.03622004 -0.05476173 0.05526149
## 8840   0.0002498795 -0.03572330 0.03622306 -0.05476635 0.05526611
## 8841   0.0002498795 -0.03572632 0.03622608 -0.05477098 0.05527073
## 8842   0.0002498795 -0.03572935 0.03622910 -0.05477560 0.05527536
## 8843   0.0002498795 -0.03573237 0.03623213 -0.05478022 0.05527998
```

```
forecast(ibm_86_final$Ri_minus_Rf, 10)
```

```
##      Point Forecast       Lo 80      Hi 80       Lo 95      Hi 95
## 8834    0.001676522 -0.02083647 0.02418952 -0.03275413 0.03610718
## 8835    0.001676522 -0.02083981 0.02419286 -0.03275924 0.03611228
## 8836    0.001676522 -0.02084315 0.02419619 -0.03276434 0.03611739
## 8837    0.001676522 -0.02084649 0.02419953 -0.03276945 0.03612249
## 8838    0.001676522 -0.02084982 0.02420287 -0.03277455 0.03612759
## 8839    0.001676522 -0.02085316 0.02420620 -0.03277965 0.03613270
## 8840    0.001676522 -0.02085649 0.02420954 -0.03278475 0.03613780
## 8841    0.001676522 -0.02085983 0.02421287 -0.03278985 0.03614290
## 8842    0.001676522 -0.02086316 0.02421621 -0.03279495 0.03614800
## 8843    0.001676522 -0.02086650 0.02421954 -0.03280005 0.03615309
```

```
forecast(intel_86_final$Ri_minus_Rf, 10)
```

```
##      Point Forecast       Lo 80      Hi 80       Lo 95      Hi 95
## 8834    0.002613453 -0.02957589 0.03480280 -0.04661590 0.05184281
## 8835    0.002598853 -0.02959064 0.03478835 -0.04663073 0.05182844
## 8836    0.002585841 -0.02960396 0.03477564 -0.04664421 0.05181590
## 8837    0.002574244 -0.02961605 0.03476454 -0.04665656 0.05180505
## 8838    0.002563908 -0.02962708 0.03475489 -0.04666795 0.05179577
## 8839    0.002554697 -0.02963719 0.03474658 -0.04667854 0.05178793
## 8840    0.002546487 -0.02964650 0.03473947 -0.04668844 0.05178141
## 8841    0.002539170 -0.02965512 0.03473346 -0.04669775 0.05177609
## 8842    0.002532648 -0.02966315 0.03472844 -0.04670657 0.05177187
## 8843    0.002526836 -0.02967064 0.03472432 -0.04671496 0.05176863
```

```
forecast(msft_86_final$Ri_minus_Rf, 10)
```

```
##      Point Forecast       Lo 80      Hi 80       Lo 95      Hi 95
## 8834   0.0009759256 -0.02678664 0.02873849 -0.04148325 0.04343510
## 8835   0.0009491158 -0.02681357 0.02871180 -0.04151024 0.04340847
## 8836   0.0009250819 -0.02683784 0.02868800 -0.04153464 0.04338481
## 8837   0.0009035365 -0.02685977 0.02866684 -0.04155677 0.04336385
## 8838   0.0008842220 -0.02687962 0.02864807 -0.04157692 0.04334536
## 8839   0.0008669072 -0.02689765 0.02863146 -0.04159531 0.04332913
## 8840   0.0008513852 -0.02691404 0.02861681 -0.04161217 0.04331494
## 8841   0.0008374704 -0.02692899 0.02860393 -0.04162767 0.04330261
## 8842   0.0008249964 -0.02694266 0.02859266 -0.04164198 0.04329197
## 8843   0.0008138138 -0.02695520 0.02858282 -0.04165522 0.04328285
```

Due to the problems with the training and testing models, for forecasting, we decided to use the normal OLS regressions for each company and forecast 10 days into the future. Our forecasts show little variation in each company's predicted returns over the 10 day period. This makes sense as stocks don't vary wildly on a typical day-to-day basis. The highest average returns are for Intel with the lowest returns for Apple. However, it is worth noting that the predicted range for each day's returns is quite large. This is likely due to stock returns generally being fairly unpredictable, despite many people's best efforts to model them.

Future Work

A natural extension to our exploration of the Fama-French 3-factor model is to study the Fama-French 5-factor model. The 5-factor model proposes two more factors: Profitability and Investment. The Profitability factor measures the excess return of firms with robust operating profitability over those with weak operating profitability. The Investment factor measures the difference in returns between firms that invest aggressively and those that invest conservatively. This would be an interesting topic for further investigation because it directly builds on the work we've conducted in analysing the 3-factor model; the data is available on Kenneth R. French's website, which we used in this project; and we can compare which model makes better predictions.

Another interesting model to explore would be the Carhart 4-factor model. This model is interesting because it adds a Monthly Momentum factor to the Fama-French 3-factor model, where Monthly Momentum measures the velocity of price changes in a security. It would be interesting to contrast the predictive results of this model with the Fama-French 3-factor model's predictions because Carhart's model includes both fundamental and technical factors.

Lastly, we can choose to analyse and compare the Fama-French 5-factor model and the Carhart 4-factor model to see which one yields better predictions. All these proposed analyses dovetail well with our current project and would be intriguing topics for future study.

References

French, Kenneth R. *Fama/French 3 Factors [Daily]*, 2021, http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html. Accessed 24 May 2021.

Hayes, Adam. *Fama and French 3-factor Model*, 21 Apr. 2021, https://www.investopedia.com/terms/f/famaandfrenchthreefactormodel.asp. Accessed 24 May 2021.

Wikipedia. *Carhart four-factor model*, 5 Mar. 2021, https://en.wikipedia.org/wiki/Carhart_four-factor_model. Accessed 24 May 2021.