SPL Fama French

Structure

1. Introduction

The Fama French model is a model for explaining stock returns. It extends the classical Capital Asset Pricing Model (CAPM) by having additional factors.

$$R_i - R_F = \beta \cdot (R_M - R_F)$$

Fama and French (1993) introduces *SMB* (Small market cap Minus Big / Size) and *HML* (High book-to-market Minus Low / Value) to capture the observation that small capitalization and high book value to market value ("value" in contrast to "growth") stocks tend to outperform the market.

$$R_i - R_F = \beta_M \cdot (R_M - R_F) + \beta_S \cdot SMB + \beta_V \cdot HML$$

Fama and French (2015) adds *RMW* (Robust operating profit Minus Weak / Profitability) and *CMA* (Conservative investment strategy Minus Aggressive / Investment).

$$R_i - R_F = \beta_M \cdot (R_M - R_F) + \beta_S \cdot SMB + \beta_V \cdot HML + \beta_P \cdot RMW + \beta_I \cdot CMA$$

Fama French factors are calculated as return spreads between two portfolios, e.g. SMB is the difference between the return of a small cap portfolio and that of a large cap portfolio.

We choose the Fama French model due to the high quality data available at Kenneth R. French's data library

Refer to Wikepedia for more information.

2. Data Preparation

2.1 Fama French Data

French's data library contains data for the factors, corresponding market returns and risk free rates, as well as the portfolios returns featured in the papers:

- **3 Factors** 1926.07.01 to 2018.03.29 as daily / weekly / monthly data
- **5 Factors** 1963.07.01 to 2018.03.29 as daily / monthly / yearly data
- **25 Portfolios (5x5)** formed on Size and Book-to-Market 1926.07 to 2018.03 corresponding to the Fama and French (1993) 3-factor setup (P24 Table 6).

The downloaded CSV data contains headers and footers that need to be removed before input to R.

2.2 S&P 500 Stock Data

The `BatchGetSymbols` library has a function `BatchGetSymbols()` for downloading S&P500 stock prices and volumes from a cached repository, thus avoiding problems when downloading large amount of data directly from Yahoo or Google (e.g. the `getSymbols` function from the `quantmod` library)

```r
library(BatchGetSymbols)
```

```
##
```

```r
# Get company information incl. tickers for SP500 stocks
Companies <- GetSP500Stocks()

# Batch download data from Yahoo Finance
Stocks<- BatchGetSymbols(tickers = Companies$tickers,
                     first.date = "2017-01-01",
                     last.date = "2017-12-31")
```

The downloaded list contains 2 dataframes:

- **df.control** contains descriptive information like whether the download for the ticker is successful.
- **df.tickers** contains the downloaded price data. Each row is the price data for one ticker at one date, hence we need to process the data into a format easier to work with.

(Use `kable()` function in `Knitr` library to format table output in PDF.)

```r
kable(head(Stocks$df.control, n=3))
```

ticker	src	download.status	total.obs	perc.benchmark.dates	threshold.decision
MMM	yahoo	OK	251	1	KEEP
ABT	yahoo	OK	251	1	KEEP
ABBV	yahoo	OK	251	1	KEEP

```r
kable(head(Stocks$df.tickers, n=3))
```

price.open	price.high	price.low	price.close	volume	price.adjusted	ref.date	ticker	ret.adjusted.prices	ret.
178.83	180.00	177.22	178.05	2509300	171.7699	2017-01-03	MMM	NA	
178.03	178.90	177.61	178.32	1542000	172.0304	2017-01-04	MMM	0.0015164	
178.26	179.14	176.89	177.71	1447800	171.4419	2017-01-05	MMM	-0.0034209	

Below code selects the downloaded tickers (marked by `df.control$threshold.decision=="KEEP"`) and use the dates from 3M as the date column for dataframe `SP500.data`.

It reads stocks ticker by ticker and matches previous price series by date. The unmatched dates will have `NA`s. The new stock price series is merged into the dataframe as a new column with the ticker symbol as the column name.

```r
good.tickers <- Stocks$df.control$
        ticker[Stocks$df.control$threshold.decision=="KEEP"]

# Fill dates as the first stock "MMM" happens to have complete dates
# (column name = "date")
SP500.data<-data.frame(date = Stocks$
                         df.tickers$
                         ref.date[1:max(Stocks$df.control$total.obs)])

for(i in 1:length(good.tickers))
{
  # X is a temp dataframe that has 2 columns,
  # 1st is date (for matching), 2nd is the actual data (e.g. closing price)

  # Choose relevant data by matching tickers
  X <- data.frame(date =
        Stocks$df.tickers$ref.date[Stocks$df.tickers$ticker==good.tickers[i]],
        Stocks$df.tickers$price.adjusted[Stocks$df.tickers$ticker==good.tickers[i]])

  # change the column name of X to be the ticker of the stock
  # colnames(X)[2] = good.tickers[i] # this one don't work
  colnames(X)[2] <- Stocks$df.tickers$
        ticker[Stocks$df.tickers$ticker==good.tickers[i]]

  # merge X as a new column into SP500.data by matching date
  # missing dates will have NA by default
  SP500.data <- merge.data.frame(SP500.data, X, by = "date", all.x = TRUE)
}
```

We write the processed data to CSVs.

3. Simple Regression

`readxl` library for reading Excel data.

The imported data would be stored as `data.frame` and must be `unlist()` into vectors for regression.

(data.frame is also a list in R)

```r
library(readxl)
FF3<- read_excel("Data/FF3_196307-199112.xlsx")
# unlist: convert the data into vector format
rmrf<-unlist(FF3[,2])
```

OLS regression can be performed with two lines of code:

```r
y <- lm(rirf ~ rmrf + smb + hml);
summary(y)
```

```
##
## Call:
## lm(formula = rirf ~ rmrf + smb + hml)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7407 -1.2491 -0.0457  1.2168  7.9848
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.38175    0.10752  -3.550 0.000439 ***
## rmrf         1.03489    0.02638  39.226  < 2e-16 ***
## smb          1.39851    0.03928  35.607  < 2e-16 ***
## hml         -0.29792    0.04402  -6.768 5.79e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.94 on 338 degrees of freedom
## Multiple R-squared:  0.9385, Adjusted R-squared:  0.9379
## F-statistic:  1718 on 3 and 338 DF,  p-value: < 2.2e-16
```

`summary(y)` contains the regression results and specific results could be obtained, e.g., via:

```r
summary(y)$coefficients
```

which returns the regression betas and their standard errors, t-values and p-values in a matrix.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.38	0.11	-3.55	0
rmrf	1.03	0.03	39.23	0
smb	1.40	0.04	35.61	0
hml	-0.30	0.04	-6.77	0

4. Replicating the 3-Factor model

To check that we have implemented the Fama French model correctly, we try to replicate the results of table 6 of Fama and French (1993) which involves monthly return data of 25 value-weighted portfolios from July 1963 to December 1991.

The data set structures the data as 1 column of months ($YYYYDD$ format) plus 25 columns of portfolio monthly returns. The first return column is SMALL (market cap) LoBM (low book-to-market / "growth"). The first 5 return columns are all small cap but with increasing book-to-market ratios. The last 5 return columns are all large cap with the last column being BIG (market cap) HiBM (high book-to-market / "value").

In reporting, results are structured in a matrix with rows representing market cap and columns for book to market ratios.

4.1 Batch regression

With the OLS regression code working, below code runs regression on each portfolio and saves the results in a list `results`.

```r
# Store summaries into a results list
results <- list()
# The first column of P25 is dates, not data
for(i in 1:(ncol(P25)-1))
{
  rirf<-unlist(P25[,i+1])-rf # Data starts from the 2nd col of P25
  y<-lm(rirf~rmrf+smb+hml)
  results[[i]]<-summary(y)
}
```

4.2 Formatting the results

We then read out the results, stack them into corresponding vectors, then reshape them into the 5×5 format as in the paper for ease of comparison. Results are highly similar and we have not yet identify why they do not match exactly, perhaps due to rounding errors.

```r
betas <- vector()
std.errors <- vector()
t.values <- vector()
R.squareds <- vector()
# save all betas
for(i in 1:(ncol(P25)-1))
{
  betas <- cbind(betas,results[[i]]$coefficients[,1])
  std.errors <- cbind(std.errors,results[[i]]$sigma)
  t.values <- cbind(t.values, results[[i]]$coefficients[,3])
  R.squareds <- cbind(R.squareds, results[[i]]$adj.r.squared)
}

# resize the output to 5x5 format like Fama French paper
resize <- function(x)
{
  df = data.frame(matrix(x, nrow=5, byrow = TRUE))
  colnames(df) = c("Low", "2", "3", "4", "High")
  rownames(df) = c("Small", "2", "3", "4", "Big")
  return(df)
}
# resize alpha
alpha <- resize(betas[1,])
kable(alpha, digits=2)
```

	Low	2	3	4	High
Small	-0.38	-0.10	-0.07	0.08	0.06
2	-0.13	-0.02	0.14	0.15	0.06
3	-0.04	0.11	-0.02	0.14	0.05
4	0.11	-0.16	0.01	0.08	0.04
Big	0.21	-0.02	-0.06	-0.06	-0.18

```
# resize beta
market.beta <- resize(betas[2,])
SMB.beta <- resize(betas[3,])
HML.beta <- resize(betas[4,])

# display beta below

kable(market.beta, digits=2)
```

	Low	2	3	4	High
Small	1.03	0.97	0.94	0.89	0.95
2	1.10	1.02	0.96	0.97	1.07
3	1.10	1.02	0.97	0.97	1.06
4	1.06	1.07	1.04	1.03	1.15
Big	0.96	1.02	0.96	1.01	1.03

```
kable(SMB.beta, digits=2)
```

	Low	2	3	4	High
Small	1.4	1.27	1.16	1.10	1.19
2	1.0	0.94	0.83	0.71	0.85
3	0.7	0.63	0.54	0.45	0.65
4	0.3	0.27	0.25	0.22	0.36
Big	-0.2	-0.19	-0.27	-0.19	-0.04

```
kable(HML.beta, digits=2)
```

	Low	2	3	4	High
Small	-0.30	0.08	0.27	0.38	0.62
2	-0.48	0.03	0.23	0.47	0.70
3	-0.43	0.04	0.31	0.50	0.71
4	-0.44	0.03	0.30	0.56	0.74
Big	-0.44	-0.02	0.20	0.56	0.76

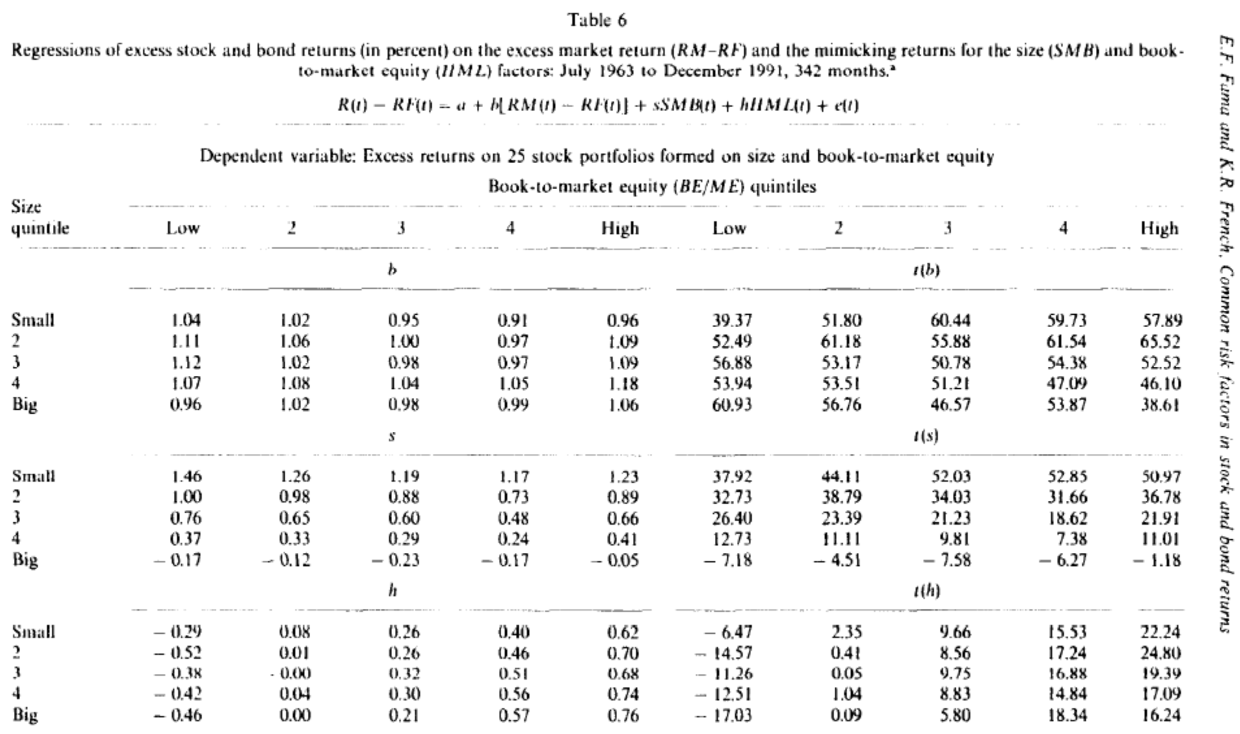


Table 6

Regressions of excess stock and bond returns (in percent) on the excess market return ($RM-RF$) and the mimicking returns for the size (SMB) and book-to-market equity (HML) factors: July 1963 to December 1991, 342 months.[a]

$$R(t) - RF(t) = a + b[RM(t) - RF(t)] + sSMB(t) + hHML(t) + e(t)$$

Dependent variable: Excess returns on 25 stock portfolios formed on size and book-to-market equity

	Book-to-market equity (BE/ME) quintiles									
Size quintile	Low	2	3	4	High	Low	2	3	4	High
			b					$t(b)$		
Small	1.04	1.02	0.95	0.91	0.96	39.37	51.80	60.44	59.73	57.89
2	1.11	1.06	1.00	0.97	1.09	52.49	61.18	55.88	61.54	65.52
3	1.12	1.02	0.98	0.97	1.09	56.88	53.17	50.78	54.38	52.52
4	1.07	1.08	1.04	1.05	1.18	53.94	53.51	51.21	47.09	46.10
Big	0.96	1.02	0.98	0.99	1.06	60.93	56.76	46.57	53.87	38.61
			s					$t(s)$		
Small	1.46	1.26	1.19	1.17	1.23	37.92	44.11	52.03	52.85	50.97
2	1.00	0.98	0.88	0.73	0.89	32.73	38.79	34.03	31.66	36.78
3	0.76	0.65	0.60	0.48	0.66	26.40	23.39	21.23	18.62	21.91
4	0.37	0.33	0.29	0.24	0.41	12.73	11.11	9.81	7.38	11.01
Big	− 0.17	− 0.12	− 0.23	− 0.17	− 0.05	− 7.18	− 4.51	− 7.58	− 6.27	− 1.18
			h					$t(h)$		
Small	− 0.29	0.08	0.26	0.40	0.62	− 6.47	2.35	9.66	15.53	22.24
2	− 0.52	0.01	0.26	0.46	0.70	− 14.57	0.41	8.56	17.24	24.80
3	− 0.38	- 0.00	0.32	0.51	0.68	− 11.26	0.05	9.75	16.88	19.39
4	− 0.42	0.04	0.30	0.56	0.74	− 12.51	1.04	8.83	14.84	17.09
Big	− 0.46	0.00	0.21	0.57	0.76	− 17.03	0.09	5.80	18.34	16.24

E.F. Fama and K.R. French, Common risk factors in stock and bond returns

Similarly for t-statistics and R^2:

```r
# resize t-stats
market.t <-resize(t.values[2,])
SMB.t <- resize(t.values[3,])
HML.t <- resize(t.values[4,])

kable(market.t, digits=2)
```

	Low	2	3	4	High
Small	39.23	50.60	58.42	57.99	57.76
2	53.20	58.56	59.98	62.77	63.25
3	59.68	56.81	53.35	58.93	51.14
4	57.16	52.61	50.34	51.30	46.30
Big	57.20	56.98	42.80	55.04	37.70

```r
kable(SMB.t, digits=2)
```

	Low	2	3	4	High
Small	35.61	44.82	48.65	48.10	48.63
2	32.62	36.36	34.80	30.78	33.82
3	25.53	23.41	20.04	18.46	21.03
4	10.92	8.75	8.06	7.49	9.64
Big	-8.10	-7.08	-7.99	-6.91	-1.05

```r
kable(HML.t, digits=2)
```

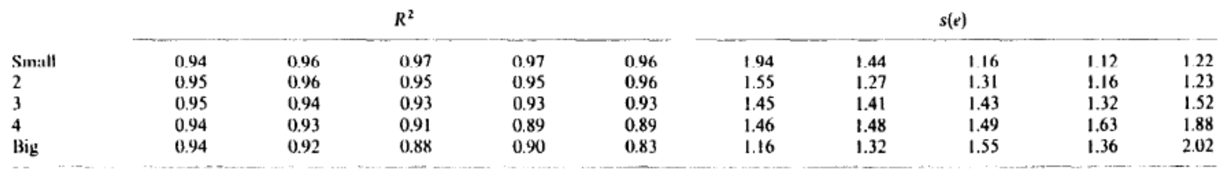
	Low	2	3	4	High
Small	-6.77	2.43	9.92	14.93	22.43
2	-13.93	0.88	8.73	18.32	24.74
3	-14.04	1.39	10.27	18.28	20.34
4	-14.24	0.79	8.77	16.68	17.79
Big	-15.96	-0.68	5.25	18.41	16.65

```r
# resize R-squareds
kable(resize(R.squareds), digits=2)
```

	Low	2	3	4	High
Small	0.94	0.96	0.96	0.96	0.96
2	0.96	0.96	0.96	0.96	0.96
3	0.96	0.95	0.93	0.94	0.93
4	0.95	0.92	0.91	0.91	0.90
Big	0.94	0.92	0.86	0.90	0.82

```r
kable(resize(std.errors), digits=2)
```

	Low	2	3	4	High
Small	1.94	1.40	1.18	1.13	1.21
2	1.52	1.28	1.18	1.13	1.24
3	1.36	1.32	1.33	1.21	1.53
4	1.37	1.50	1.52	1.48	1.83
Big	1.23	1.32	1.66	1.34	2.00



	R^2					$s(e)$				
Small	0.94	0.96	0.97	0.97	0.96	1.94	1.44	1.16	1.12	1.22
2	0.95	0.96	0.95	0.95	0.96	1.55	1.27	1.31	1.16	1.23
3	0.95	0.94	0.93	0.93	0.93	1.45	1.41	1.43	1.32	1.52
4	0.94	0.93	0.91	0.89	0.89	1.46	1.48	1.49	1.63	1.88
Big	0.94	0.92	0.88	0.90	0.83	1.16	1.32	1.55	1.36	2.02

5. S&P500 Results

We first apply the above methods on the downloaded S&P 500 stocks' price returns to see if there is any pattern with the regression results. Also to test out the code for handling hundreds of stocks.

Then we separate the data by 5-year periods and loop over both years and stocks to see if patterns change over time.

5.1 Running the model for S&P 500 stocks

Below code works as follows:

1. Read-in price data and do the necessary formatting.
2. Frame the data to the desired time period.
3. Convert the price data series into XTS series as required by 5.
4. Remove stocks with `NA`s in the series.
5. Use `quantmod` library's `monthlyReturn()` function to batch convert the whole price matrix into a monthly return matrix.

We need to remove `NA`s for using the `monthlyReturn()` function. Most `NA`s are due to data not available on the starting date of the series, e.g. the company has not IPO yet.

Here we face choices:

- Remove all columns with `NA`s, then all remaining stocks could have the regression in the same period, i.e. with the same number of observations. (This section)
- Dynamically frame the data based on the available non-`NA` data points, but then some stocks in the regression analysis will have fewer observations. (Tested in Section 5.2)

```r
library(quantmod)

# Read SP500 daily data and convert date column to date format
SP500.data <- read.csv("Data/SP500_price.adjusted_2010-2017.csv")
SP500.data$date <- as.Date(SP500.data$date)

# Select 2010 - 2017 range
Stock.Prices.Daily <- SP500.data[SP500.data$date>="2010-01-01" &
                                   SP500.data$date<="2017-12-31",-1]

# Current FF3 till 201803, monthly
FF3 <- read.csv("Data/original/FF3.csv")
FF <- FF3[FF3$X >= 201001 & FF3$X <= 201712,]

# Convert series to XTS for using quantmod's monthlyReturn function
Stock.Prices.Daily <- xts(Stock.Prices.Daily[,-1],
                          order.by = as.POSIXct(Stock.Prices.Daily$date))

# Number of stocks to start with
ncol(Stock.Prices.Daily)
```

```
## [1] 465
```

```r
# Remove stocks with NAs in the series, otherwise monthly Return will not work properly
Stock.Prices.Daily <- Stock.Prices.Daily[,colSums(is.na(Stock.Prices.Daily)) == 0]
```

```r
# Apply monthlyReturn function to each column (it seems it converts only one column at a time)
Stock.Prices.Monthly <- do.call(cbind, lapply(Stock.Prices.Daily, monthlyReturn))
# Stock.Prices.Monthly <- na.omit(Stock.Prices.Monthly)
colnames(Stock.Prices.Monthly) <- colnames(Stock.Prices.Daily)

# Number of stocks left
ncol(Stock.Prices.Monthly)
```

```
## [1] 442
```

As in this example, we start with 465 stocks and remove 23 stocks with incomplete data.

Then the regression part is similar to Section 4.1, except that we need to transpose the coefficients to get the dimensions right before stacking them together column by column, with each column representing one stock.

```r
Results <- list()
for(i in 1:ncol(Stock.Prices.Monthly))
{
  RiRF <- Stock.Prices.Monthly[,i] - FF$RF
  Regression <- lm(RiRF ~ FF$Mkt.RF + FF$SMB + FF$HML)
  Results[[i]] <- summary(Regression)
}

# Results!
betas <- vector()
std.errors <- vector()
t.values <- vector()
p.values <- vector()
r.squareds <- vector()
adj.r.squareds <- vector()

for(i in 1:ncol(Stock.Prices.Monthly))
{
  betas <- cbind(betas,Results[[i]]$coefficients[,1])
  std.errors <- cbind(std.errors,Results[[i]]$sigma)
  t.values <- cbind(t.values, Results[[i]]$coefficients[,3])
  p.values <- cbind(p.values, Results[[i]]$coefficients[,4])

  r.squareds <- cbind(r.squareds, Results[[i]]$r.squared)
  adj.r.squareds <- cbind(adj.r.squareds, Results[[i]]$adj.r.squared)

}

Regression.results <- cbind(data.frame(colnames(Stock.Prices.Monthly)),
                    t(r.squareds), t(adj.r.squareds),
                    t(betas), t(p.values))

colnames(Regression.results) = c("Ticker", "R.Squared", "Adj.R.Squared",
                         "Intercept", "Mkt.Rf", "SMB", "HML",
                         "P(Intercept)", "P(Mkt.Rf)", "P(SMB)", "P(HML)")
```

We add company information like name and sector to make the results easier to understand. The constituent data is from a downloaded CSV file, which can also be found in the downloaded data introduced in Section 2.2.

We use a left join (`merge()` function with parameter all.x = TRUE) to add company name and sector to our regression results.

```
# Read in SP500 company ticker information
Mapping <- read.csv("Data/constituents.csv")
colnames(Mapping)[1] <- "Ticker"
Regression.results <- merge(x = Regression.results, y = Mapping, by = "Ticker", all.x = TRUE)
```

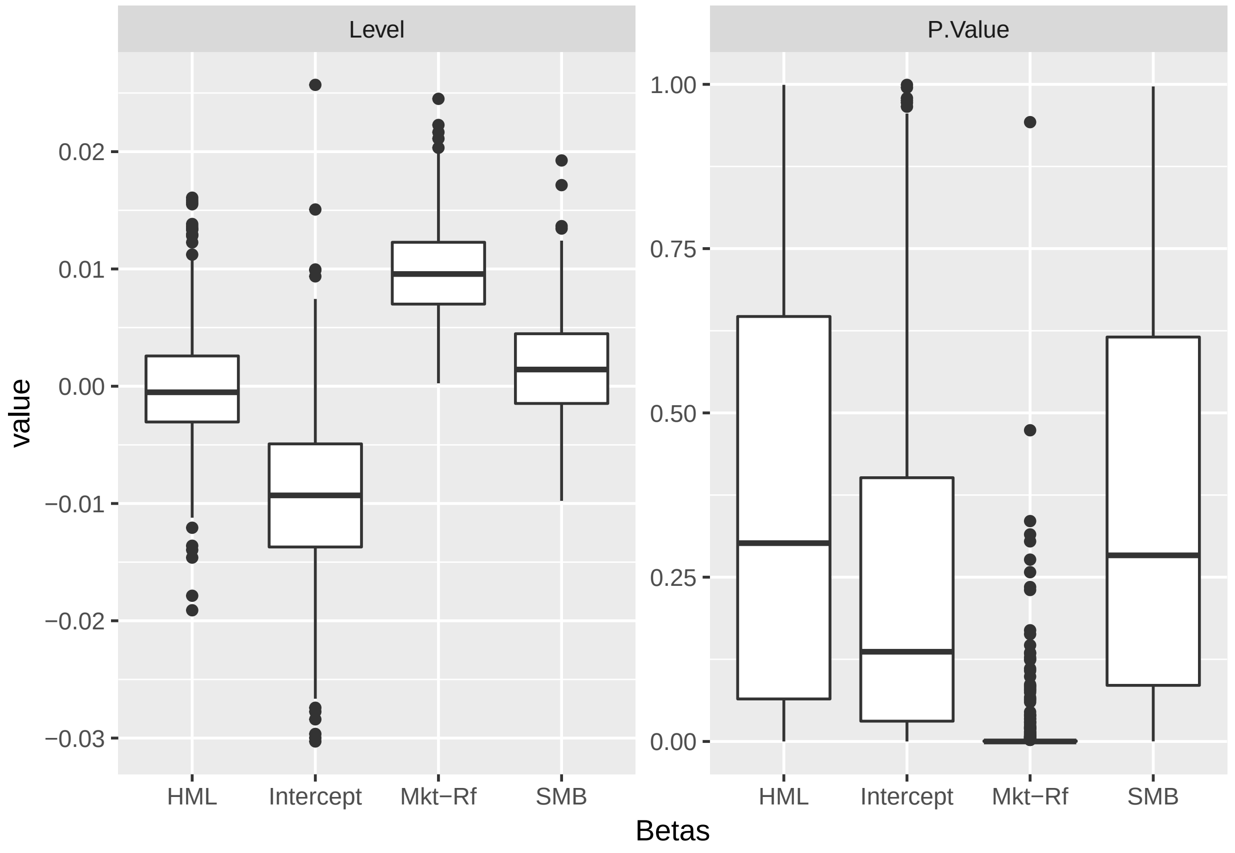
Then we can easily filter out specific companies, e.g. companies and sectors whose returns have the highest R^2 in the Fama French model. Interesting to see Financials come on top:

```
# select stocks with R2>=0.08
R2 <- Regression.results[Regression.results$R.Squared>=0.08,
                        c("Ticker","Name","Sector","R.Squared")]

# sort with R2 from largest to smallest, get top 10
kable(head(R2[order(R2$R.Squared, decreasing = T),], n=10), digits = 4)
```

	Ticker	Name	Sector	R.Squared
388	TROW	T. Rowe Price Group	Financials	0.6806
227	IVZ	Invesco Ltd.	Financials	0.6777
31	AMG	Affiliated Managers Group Inc	Financials	0.6646
283	MS	Morgan Stanley	Financials	0.6334
334	PRU	Prudential Financial	Financials	0.6308
201	HON	Honeywell Int'l Inc.	Industrials	0.6299
270	MET	MetLife Inc.	Financials	0.6279
321	PFG	Principal Financial Group	Financials	0.6262
60	BEN	Franklin Resources	Financials	0.6225
232	JPM	JPMorgan Chase & Co.	Financials	0.6027

We could also box-plot the distribution of the betas and their p-values. A new column is needed for using the `melt()` function (`reshape2` library) for the convenience of box-plot. In general, each column in the dataframe will be plotted into a separated graph, while data within each column is grouped by the value in the added column. Hence in the below code, the original data frame contains two columns: the estimated β's and their p-values. The added column in the dataframe marks which rows are the estimated coefficients for intercept, which rows are the estimated β_M, etc.

```
# boxplot of regression results
library(ggplot2)
library(reshape2)
num.stocks <- dim(Regression.results)[1]

plot.data <- data.frame(Betas = rep(c("Intercept", "Mkt-Rf", "SMB", "HML"),
                        rep(num.stocks, 4)))

plot.data$Level <- as.vector(cbind(
                    Regression.results$Intercept,
                    Regression.results$Mkt.Rf,
                    Regression.results$SMB,
                    Regression.results$HML))

plot.data$P.Value<- as.vector(cbind(
                    Regression.results$`P(Intercept)`,
                    Regression.results$`P(Mkt.Rf)`,
                    Regression.results$`P(SMB)`,
                    Regression.results$`P(HML)`))

plot.melt <- melt(plot.data, "Betas")
ggplot(plot.melt, aes(x=Betas, y=value)) + geom_boxplot() +
                    facet_wrap(~ variable, scales='free')
```



From the p-values, *SMB* and *HML* are not significant for many stocks.

5.2 Running the model for each 5-year period from 1980 to 2015

Data downloaded with `BatchGetSymbols` has an issue that the earlier the series (e.g. in the 1980s), the less stocks are available, most probably due to stocks being replaced in the S&P 500 index. To fix this issue, we could either:

1. Get the constituents for S&P 500 for each period and download those exact tickers, which may not work due to data availability. Even if it worked, we might be comparing apples to oranges, if the set of companies change over time.
2. Limit the data set to companies that survive over time. But then we have a much smaller set and miss out large names like Google or Facebook since they IPO in the 2000s.

Currently we simply take all the data available for each period for the regression, thus the results should be interpreted with a grain of salt.

Code is built based on Section 5.1, except that we stored only the results needed for plotting. Here in the document the `print()` and `cat()` functions are muted as they were merely for displaying the progress of the code in run time. Library `lubridate` provides some nice functions like `year()` for handling dates.

```r
# loop over above codes to regress data from 1980 - 2015, group every 5 yrs.
library(lubridate)
```

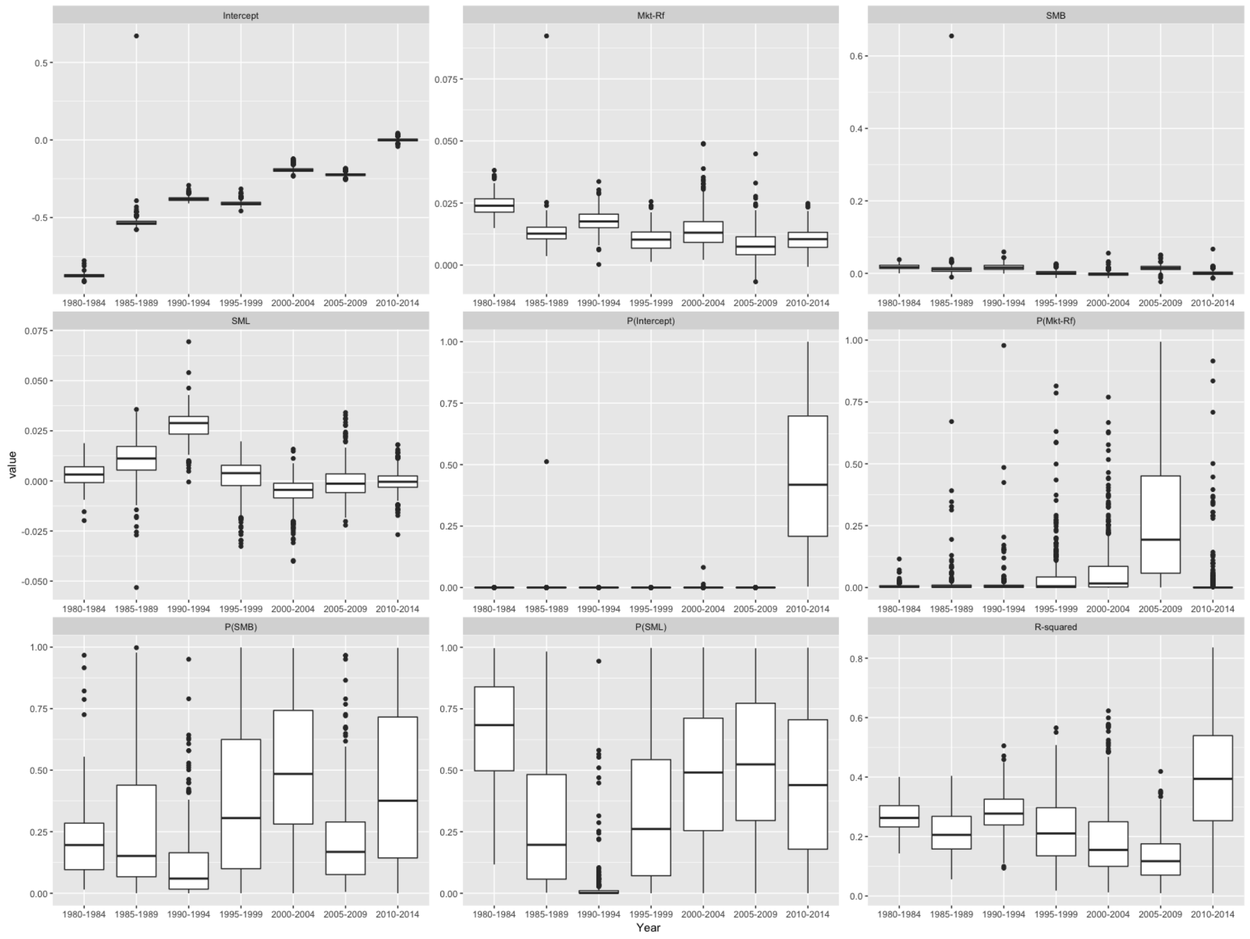
```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```r
List.of.start.date <- seq(as.Date("1980/1/1"), as.Date("2016/1/1"), "years")
List.of.start.date <- List.of.start.date[year(List.of.start.date)%%5==0]

# FF3: 192607 - 201803, monthly
FF3 <- read.csv("Data/original/FF3.csv")

# Each batch stores results for a 5yr group
Batch <- list()
Descriptions <- list()

Beta.batch <- list()

for(i in 1:(length(List.of.start.date)-1))
{
  start.date <- as.Date(List.of.start.date[i])
  end.date <- as.Date(List.of.start.date[i+1])-1
  # print(paste(start.date, end.date,sep=" - "))

  # read data
  file.name <- paste("Data/SP500_price.adjusted_",
                     paste(year(start.date), year(end.date), sep="-"), ".csv", sep="")
  SP500.data <- read.csv(file.name)
  SP500.data$date <- as.Date(SP500.data$date)

  # remove first column "X" created due to importing
  Stock.Prices.Daily <- SP500.data[SP500.data$date>= start.date &
                                     SP500.data$date<= end.date,-1]

  # Convert series to XTS for using quantmod's monthlyReturn function
  Stock.Prices.Daily <- xts(Stock.Prices.Daily[,-1],
                            order.by = as.POSIXct(Stock.Prices.Daily$date))

  # try a diff approach: loop over stocks and convert to monthly for each stock

  # initialize
  Results <- list()
  Description <- data.frame()

  betas <- data.frame()

  # loop through stocks
  for(j in 1:ncol(Stock.Prices.Daily))
  {
    # The j-th stock
    Rj <- Stock.Prices.Daily[,j]
    # cat(colnames(Stock.Prices.Daily[,j]), " ")
    # non-NA entries
    Rj <- Rj[!is.na(Rj),]
    Rj <- monthlyReturn(Rj)

    # matching FF data
    FF <- FF3[FF3$X >= format(index(head(Rj, n=1)), "%Y%m") &
              FF3$X <= format(index(tail(Rj, n=1)), "%Y%m"), ]

    # Rj is now RjRF
    Rj <- Rj-FF$RF
    Regression <- lm(Rj ~ FF$Mkt.RF + FF$SMB + FF$HML)
    Results[[j]] <- summary(Regression)
    Description <- rbind(Description,
                         data.frame(colnames(Stock.Prices.Daily[,j]),
                                    format(index(head(Rj, n=1)), "%Y%m"),
                                    format(index(tail(Rj, n=1)), "%Y%m"),
                                    length(Rj)))

    # try read-out results at regression time
    # betas, p-values, r-squareds
    betas <- rbind(betas, cbind(data.frame(t(Results[[j]]$coefficients[,1])),
                                data.frame(t(Results[[j]]$coefficients[,4])),
                                data.frame(t(Results[[j]]$r.squared))))
  }

  # Save all regression summaries
  Batch[[i]] <- Results

  # Save the ticker / dates for ease of tracking the regression summary
  colnames(Description) = c("Ticker", "Start.Month", "End.Month", "Number.of.Months")
  Descriptions[[i]] <- Description

  # Save the regression results for plotting
  colnames(betas) <- c("Intercept", "Mkt-Rf", "SMB", "SML",
                       "P(Intercept)", "P(Mkt-Rf)", "P(SMB)", "P(SML)",
                       "R-squared")

  # Try rbind here instead of list for convenience of melt.
  Beta.batch[[i]] <- betas

  # remove temp variables
  rm(Description, Results, Regression, Rj, betas)
}
```

Similar to Section 5.1, we use `melt()` function and `ggplot()` for visualizing the results:

```
df <- data.frame()
Num.Obs <- data.frame()
for(i in 1:(length(List.of.start.date)-1))
{
  start.date <- as.Date(List.of.start.date[i])
  end.date <- as.Date(List.of.start.date[i+1])-1
  label <- paste(year(start.date), year(end.date),sep="-")
  df <- rbind(df, cbind(rep(label, dim(Beta.batch[[i]])[1]), Beta.batch[[i]]))
  Num.Obs <- rbind(Num.Obs,
                   cbind(  paste(year(start.date), year(end.date),sep="-"),
                           dim(Beta.batch[[i]])[1]))
}

colnames(df) <- c("Year",
                  "Intercept", "Mkt-Rf", "SMB", "SML",
                  "P(Intercept)", "P(Mkt-Rf)", "P(SMB)", "P(SML)",
                  "R-squared")
```

```
df.melt <- melt(df, "Year")
ggplot(df.melt, aes(x=Year, y=value)) + geom_boxplot()
              + facet_wrap(~ variable, scales='free')
```

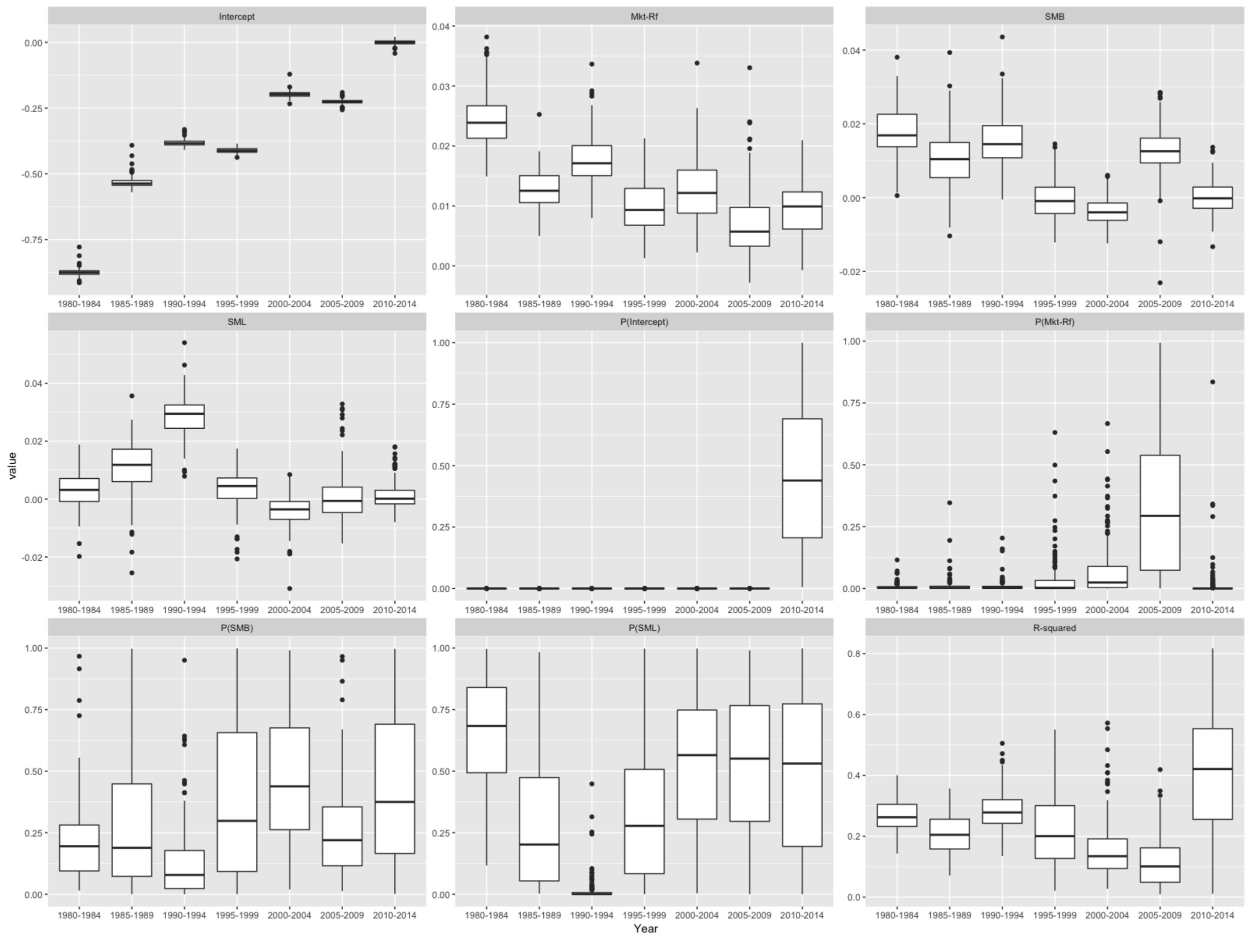


As emphasized earlier, the results are probably due to having varying stocks in each period:

Time Period	Number of Stocks
1980-1984	170
1985-1989	229
1990-1994	271
1995-1999	345
2000-2004	394

Time Period	Number of Stocks
2005-2009	432
2010-2014	459

If we filter out stocks surviving all periods, we get 168 tickers. Surprisingly, results has only minor changes



6. Going 5-Factor

References

Fama, Eugene F., and Kenneth R. French. 1993. "Common risk factors in the returns on stocks and bonds." *Journal of Financial Economics* 33 (1): 3–56. doi:10.1016/0304-405X(93)90023-5.

———. 2015. "A five-factor asset pricing model." *Journal of Financial Economics* 116 (1). Elsevier: 1–22. doi:10.1016/j.jfineco.2014.10.010.