Factor Analysis-Comparative Study of Investment Factors

May Li, Katerina Athanasia, Wenchang Zhu

Introduction

We focus on six key style factors: Beta, Size, Value, Profitability, Investment, and Momentum, in the world of investment analysis. It is structured to first replicate the renowned Fama-French model, examining the methodology behind their factor premium calculations. Subsequently, it extends this analysis through a univariate sorting approach for factors not originally treated univariately in the Fama-French framework. The overarching goal is to understand how these factors interact and contribute to stock market dynamics and how they contribute the investment returns in reality.

Load Packages

```r
library(tidyverse)
library(tidyquant)
library(scales)
library(dbplyr)
library(RSQLite)
library(DBI)
library(frenchdata)
library(quantmod)
library(slider)
library(parallelly)
library(future)
library(furrr)
library(rlang)
library(sandwich)
library(lmtest)
library(ggplot2)
library(roll)
library(corrgram)
library(broom)
```

Load data

All data needs to perform the analyses in this project has been stored in a database called tidy_finance_r.sqlite.

```r
tidy_finance <- dbConnect(
  SQLite(),
  "/Users/may/Desktop/5846/Class_12/data/tidy_finance_r.sqlite",
  extended_types = TRUE
)

#List of tables available
list <- dbListTables(tidy_finance)
list
```

```
 [1] "beta"                     "compustat"
 [3] "cpi_monthly"              "crsp_monthly"
 [5] "factors_ff3_daily"        "factors_ff3_monthly"
 [7] "factors_ff5_monthly"      "factors_mom_monthly"
 [9] "factors_q_monthly"        "ff_25_portfolios_monthly"
[11] "fisd"                     "industries_ff_monthly"
[13] "macro_predictors"         "trace_enhanced"
```

After checking the tables available in the database, we choose to load the following tables for our analysis.

```r
crsp_monthly <- tbl(tidy_finance, "crsp_monthly") |>
  select(
    permno, gvkey, month, ret_excess,
    mktcap, mktcap_lag, exchange, industry, ret
  ) |>
  collect()

compustat <- tbl(tidy_finance, "compustat") |>
    select(gvkey, datadate, be, op, inv) |>
    collect()
```

```r
factors_ff5_monthly <- tbl(tidy_finance, "factors_ff5_monthly") |>
  collect()

beta <- tbl(tidy_finance, "beta") |>
  select(permno, month, beta_monthly) |>
  collect()

beta_lag <- beta |>
  mutate(month = month %m+% months(1)) |>
  select(permno, month, beta_lag = beta_monthly) |>
  drop_na()
```

Methodology

0.1 Univariate Sort

In a univariate sort, we organize data based on a single variable.

- Selection of Variable: Choose one variable (e.g., size, book-to-market ratio, momentum, etc.) on which to sort stocks or other assets.
- Ranking and Sorting: Rank the assets based on the selected variable. For example, if the variable is firm size, you would rank all firms from the smallest to the largest.
- Portfolio Formation: Divide these ranked assets into portfolios. For instance, you might create deciles (10 portfolios), with each containing assets that fall within a specific range of the variable.
- Analysis: Analyze the performance of these portfolios to draw conclusions about the impact of the selected variable on returns or other financial metrics.

0.2 Bivariate Sort

Bivariate sorts extend the concept to two variables. There are two main types:

0.2.1 Independent Bivariate Sort

- Selection of Variables: Choose two independent variables (e.g., size and book-to-market ratio).
- Separate Ranking and Sorting: Independently rank and sort assets based on each variable. For instance, sort once by size and then separately by book-to-market ratio.

- Portfolio Formation: Form portfolios based on the intersection of these two sorts. For example, one portfolio might consist of small-size, high book-to-market stocks.
- Analysis: Examine the portfolios to understand the combined effect of the two variables on financial metrics.

0.2.2 Dependent Bivariate Sort

- Primary and Secondary Variables: Choose a primary sorting variable and a secondary sorting variable. The secondary sort is dependent on the outcome of the primary sort.
- Primary Sort and Ranking: First, sort and rank the assets based on the primary variable.
- Secondary Sort Within Each Primary Category: Within each category created by the primary sort, perform a secondary sort based on the second variable.
- Portfolio Formation: Create portfolios based on this two-step sorting process.
- Analysis: Assess how the combination of these two variables influences the behavior of the assets.

0.3 Portfolio Sort Function

We utilized the following function for grouping stocks into a chosen number of portfolios, offering us the versatility to select any variable for sorting, referred to as sorting_variable. To determine the dividing points for n_portfolios, we apply the quantile() function. Subsequently, the findInterval() function is used to allocate specific portfolios to each stock. The result of this function is a newly added column in our dataset, indicating the portfolio number assigned to each stock.

For the exchanges we try using only "NYSE" since it holds the majority of market capitalization and using all the stocks of NYSE, NASDAQ and AMEX.

A detailed explanation of choosing breakpoints will be introduced in the size factor section.

```r
assign_portfolio <- function(data,
                             sorting_variable,
                             n_portfolios,
                             exchanges) {
  breakpoints <- data |>
    filter(exchange %in% exchanges) |>
    pull({{ sorting_variable }}) |>
    quantile(
      probs = seq(0, 1, length.out = n_portfolios + 1),
      na.rm = TRUE,
      names = FALSE
    )

  assigned_portfolios <- data |>
    mutate(portfolio = findInterval(
      pick(everything()) |>
        pull({{ sorting_variable }}),
      breakpoints,
      all.inside = TRUE
    )) |>
    pull(portfolio)

  return(assigned_portfolios)
}
```

Factor 1. Market Risk

Definition

The market risk premium, especially in the context of the market factor in the Fama French model, is typically understood as the excess return of the market portfolio over the risk-free rate.

$$\text{Market Risk Premium} = R_m - R_f$$

where: R_m is the return on the market portfolio, and R_f is the risk-free rate.

This excess return isn't calculated through sorting but is rather observed in market data. Therefore, we obtain our market risk premium by the following code.

```
factors_market <- factors_ff5_monthly |>
  select(month, mkt_excess)
```

1.1 Beta Analysis

Definition

Beta refers to a measure of a stock's sensitivity to movements in the overall market, often represented by a market index like the S&P 500. Beta is a component of the Capital Asset Pricing Model (CAPM) and measures the systematic risk of an individual stock in relation to the market. For factor analysis, beta is often used as a factor reflecting a stock's exposure to market movements and considered as a component of returns attributed to market risk.

The formula for beta is given by:

$$\beta = \text{Covariance of Stock Returns with Market Returns}/\text{Variance of Market Returns}$$

where:

- Covariance of Stock Returns with Market Returns represents how the stock's returns move in relation to the market returns.
- Variance of Market Returns is the measure of the market's volatility.

A beta greater than 1 indicates that the stock tends to be more volatile than the market. A beta of less than 1 implies lower volatility compared to the market, and a negative beta suggests an inverse relationship with the market.

1.1.1 Rolling Window Estimation

The beta estimation stored in the database is based on a rolling-window estimation, which use the following functions to calculate Beta by shifting a predetermined time interval across a sequence of recorded observations through Rolling-Window method.

By moving this window of historical data point-by-point through time, the functions continually update these parameters and generate new forecasts or simulations for the future observations, which allows analysts to adapt their models dynamically.

Applying the above function using group_by() on crsp_monthly and months = 60, min_obs = 48, we will get the beta estimation stored in the database. To conserve computing power, we will not actually operate the application code but call the data stored in the database for analysis in this context.

```r
# Calculate beta (the slope coefficient) using Capital Asset Pricing Model (CAPM)
estimate_capm <- function(data, min_obs = 1) {
  if (nrow(data) < min_obs) {
    beta <- as.numeric(NA)
  } else {
    fit <- lm(ret_excess ~ mkt_excess, data = data)
    beta <- as.numeric(coefficients(fit)[2])
  }
  return(beta)
}

# Apply a rolling-window approach to estimate beta over a specified number of months
roll_capm_estimation <- function(data, months, min_obs) {
  data <- data |>
```

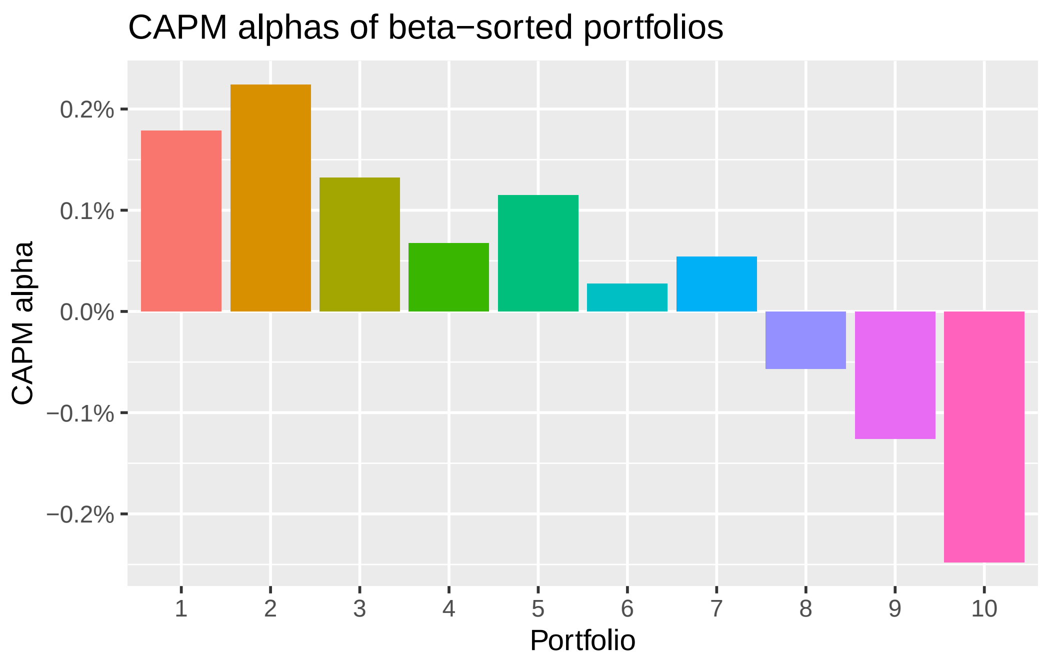
```
    arrange(month)

  betas <- slide_period_vec(
    .x = data,
    .i = data$month,
    .period = "month",
    .f = ~ estimate_capm(., min_obs),
    .before = months - 1,
    .complete = FALSE
  )

  return(tibble(
    month = unique(data$month),
    beta = betas
  ))
}
```

1.1.2 Beta-Return Relationship

We sort beta into 10 portfolios using the assign_portfolio function introduced above to examine the relationship between beta and return.

```
data_for_sorts <- crsp_monthly |>
  inner_join(beta_lag, by = c("permno", "month"))

beta_portfolios <- data_for_sorts |>
  group_by(month) |>
  mutate(
    portfolio = assign_portfolio(
      data = pick(everything()),
      sorting_variable = beta_lag,
      n_portfolios = 10,
      exchange = 'NYSE'),
    portfolio = as.factor(portfolio)) |>
  group_by(portfolio, month) |>
  summarize(
    ret_excess = weighted.mean(ret_excess, mktcap_lag),
    .groups = "drop")|>
  left_join(factors_ff5_monthly, by = "month")

beta_portfolios_summary <- beta_portfolios |>
```

```r
  select(-smb,-hml,-rf,-rmw,-cma) |>
  nest(data = c(month, ret_excess, mkt_excess)) |>
  mutate(estimates = map(
    data, ~ tidy(lm(ret_excess ~ 1 + mkt_excess, data = .x))
  )) |>
  unnest(estimates) |>
  select(portfolio, term, estimate) |>
  pivot_wider(names_from = term, values_from = estimate) |>
  rename(alpha = `(Intercept)`, beta = mkt_excess) |>
  left_join(
    beta_portfolios |>
      group_by(portfolio) |>
      summarize(ret_excess = mean(ret_excess),
                .groups = "drop"), by = "portfolio"
  )

beta_portfolios_summary |>
  ggplot(aes(x = portfolio, y = alpha, fill = portfolio)) +
  geom_bar(stat = "identity") +
  labs(
    title = "CAPM alphas of beta-sorted portfolios",
    x = "Portfolio",
    y = "CAPM alpha",
    fill = "Portfolio"
  ) +
  scale_y_continuous(labels = percent) +
  theme(legend.position = "None")
```



These results suggest a negative relation between beta and future stock returns, which contradicts the predictions of the CAPM. According to CAPM, there should be a positive correlation between returns and beta across portfolios, with risk-adjusted returns being statistically close to zero.

This discrepancy prompts the need for a more nuanced approach to understanding stock returns. Hence, we propose the introduction of additional factors as outlined in the Fama-French Five-Factor model and momentum factor as follows.

Factor 2. Size

Definition

By categorizing stocks based on their corresponding firm size, as defined by their market capitalization, we can divide them into two groups:

- **Small-Cap Stocks**: Smaller market capitalization, often seen as having higher growth potential but also higher volatility.
- **Large-Cap Stocks**: "Giants" of the market, typically characterized by stability but with potentially slower growth rates.

The average returns that are created by following the strategy of going long on small stocks and going short on large ones, are known as the size premium (**size factor**).

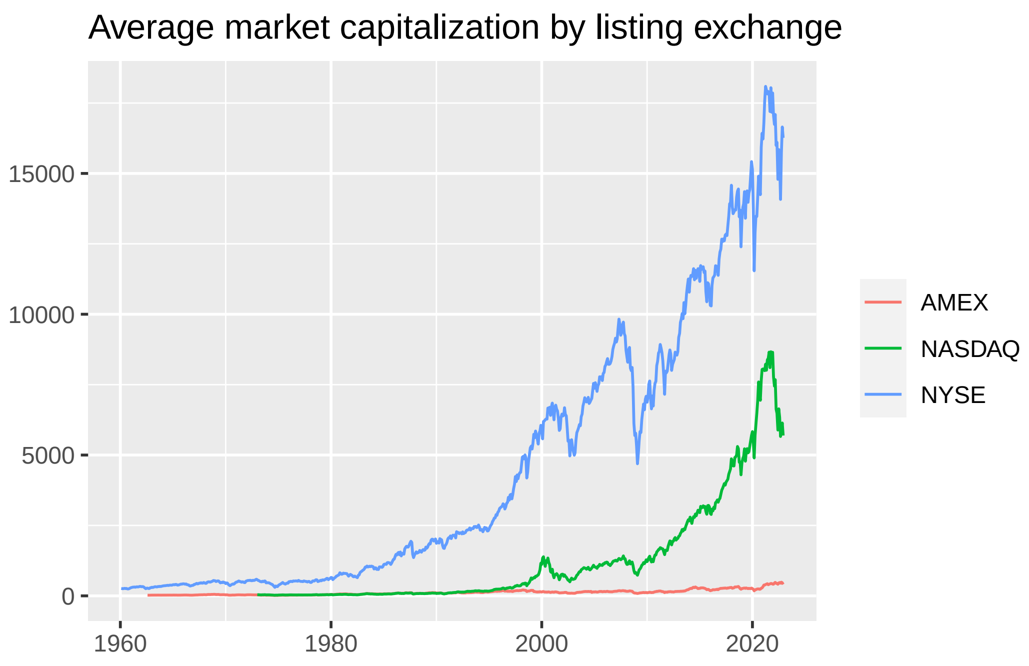
$$\text{Size Premium} = \text{SMB} = \text{Small minus Big}$$

Motivation

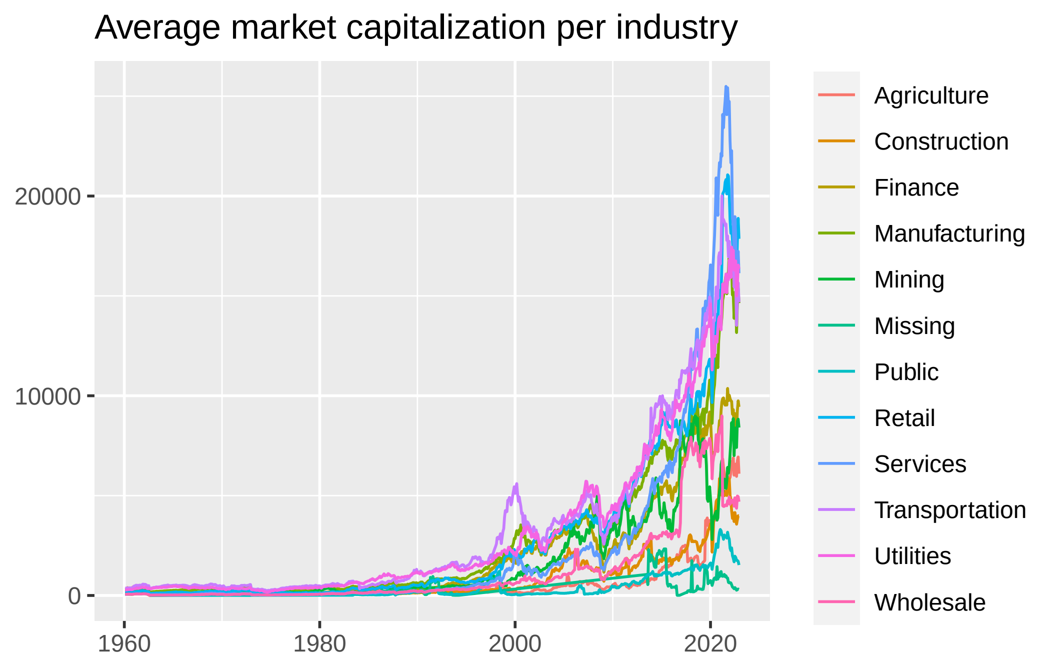
Like all the other variables/factors presented, size was at first an empirically determined average-return variable, that is they were determined based on observed patterns in returns, but Fama and French proved that combined with the other factors, they have explanatory power. When combined with the value variable presented later, the variables efficiently explain the cross-section of average returns on NYSE, Amex and NASDAQ stocks from 1963-1990. (source)

2.1 Prepare size as a dependent variable for other factors

```
# Plot average mktcap per listing exchange
crsp_monthly |>
  filter(exchange != "Other") |>
  group_by(month, exchange) |>
  summarize(mktcap = mean(mktcap),
            .groups = "drop_last") |>
  ggplot(aes(
    x = month,
    y = mktcap,
    color = exchange)) +
  geom_line() +
  labs(
    x = NULL, y = NULL, color = NULL, linetype = NULL,
    title = "Average market capitalization by listing exchange"
  )
```



```
# Plot average mktcap per industry
crsp_monthly |>
  group_by(month, industry) |>
  summarize(mktcap = mean(mktcap),
            .groups = "drop_last") |>
  ggplot(aes(
    x = month,
    y = mktcap,
    color = industry)) +
  geom_line() +
  labs(
    x = NULL, y = NULL, color = NULL, linetype = NULL,
    title = "Average market capitalization per industry"
  )
```



To compute the SMB, we split our stocks into portfolios in which we can apply the strategy resulting in the size premium.

The **breakpoints** are defined by finding the **quantiles** dividing the set of stocks, either from a specific exchange or from all exchanges, into n equal partitions.

Using the following function we can calculate the portfolio returns if we follow the strategy resulting in the size premium.

```
compute_portfolio_returns <- function(n_portfolios = 10,
                                      exchanges = c("NYSE", "NASDAQ", "AMEX"),
                                      value_weighted = TRUE,
                                      data = crsp_monthly) {

  data |>
    group_by(month) |>
    mutate(portfolio = assign_portfolio(
      sorting_variable = mktcap_lag,
      n_portfolios = n_portfolios,
      exchanges = exchanges,
      data = pick(everything())
    )) |>
    group_by(month, portfolio) |>
    summarize(
      ret = if_else(value_weighted,
        weighted.mean(ret_excess, mktcap_lag),
        mean(ret_excess)
      ),
      .groups = "drop_last"
    ) |>
    summarize(size_premium = ret[portfolio == min(portfolio)] -
      ret[portfolio == max(portfolio)]) |>
    summarize(size_premium = mean(size_premium))
}
```

We can observe that different choices in the number of portfolios, the exchange on which the breakpoints are calculated and whether the returns are value-weighted or not, it results in different size premium estimations.

Below we form different combinations in order to observe the distribution of SMB across those combinations.

Namely:

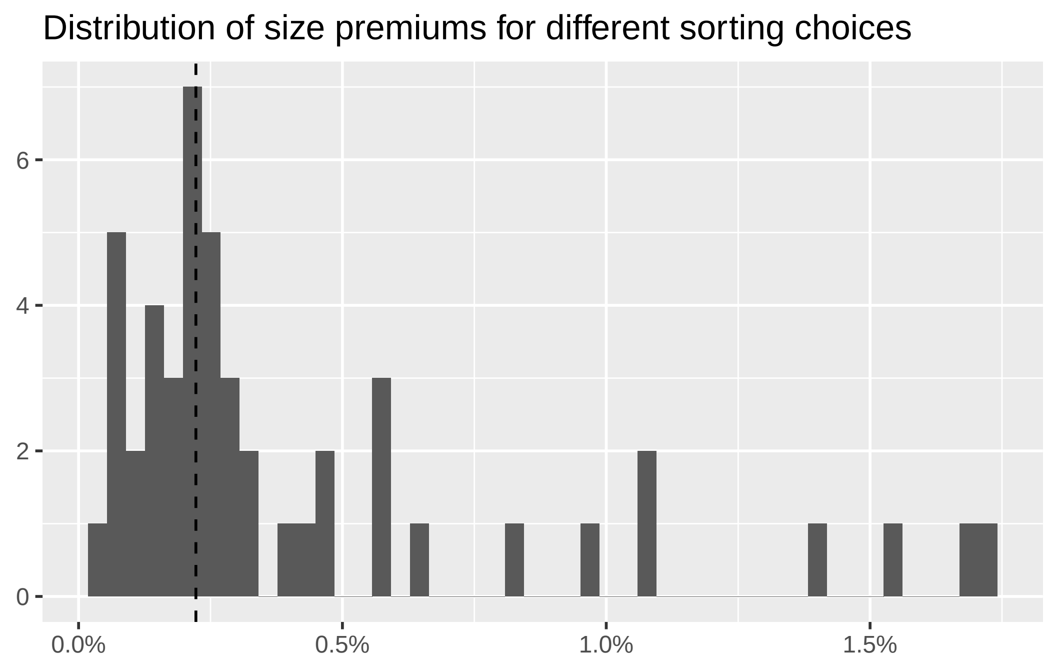
- For the split, we chose number of portfolios that corresponds to median, quartile and decile breakpoints.
- For the exchanges we try using only “NYSE” since it holds the majority of market capitalization and using all the stocks of NYSE, NASDAQ and AMEX.
- And we tried value-weighted and not returns.
- We also tried excluding firms in the Finance industry as suggested by several articles
- And tried breaking the dataset available in half in terms of time since from around that time a difference in the total market capitalization per listing exchange is observed.

```
#|cache: true
# Create the choice grid
choice_grid <- expand_grid(
  n_portfolios = c(2, 4, 10),
  exchanges = list("NYSE", c("NYSE", "NASDAQ", "AMEX")),
  value_weighted = c(TRUE, FALSE),
  data = parse_exprs(
    'crsp_monthly;
     crsp_monthly |> filter(industry != "Finance");
     crsp_monthly |> filter(month < "1990-06-01");
     crsp_monthly |> filter(month >= "1990-06-01")'
  )
)
```

```r
#|cache: true
# Parallel computation
n_cores = availableCores() - 1
plan(multisession, workers = n_cores)

choice_grid <- choice_grid |>
  mutate(size_premium = future_pmap(
    .l = list(
      n_portfolios,
      exchanges,
      value_weighted,
      data
    ),
    .f = ~ compute_portfolio_returns(
      n_portfolios = ..1,
      exchanges = ..2,
      value_weighted = ..3,
      data = eval_tidy(..4)
    )
  ))

smb_dist <- choice_grid |>
  mutate(data = map_chr(data, deparse)) |>
  unnest(size_premium) |>
  arrange(desc(size_premium))
smb_dist
```

```
# A tibble: 48 x 5
   n_portfolios exchanges value_weighted data                          size_premium
          <dbl> <list>    <lgl>          <chr>                                <dbl>
 1           10 <chr [3]> FALSE          "filter(crsp_monthly, ind~      0.0174
 2           10 <chr [3]> FALSE          "filter(crsp_monthly, mon~      0.0170
 3           10 <chr [3]> FALSE          "crsp_monthly"                  0.0155
 4           10 <chr [3]> FALSE          "filter(crsp_monthly, mon~      0.0139
 5           10 <chr [3]> TRUE           "filter(crsp_monthly, mon~      0.0109
 6           10 <chr [3]> TRUE           "filter(crsp_monthly, ind~      0.0108
 7           10 <chr [3]> TRUE           "crsp_monthly"                  0.00955
 8           10 <chr [3]> TRUE           "filter(crsp_monthly, mon~      0.00829
 9            4 <chr [3]> FALSE          "filter(crsp_monthly, ind~      0.00646
10            4 <chr [3]> FALSE          "filter(crsp_monthly, mon~      0.00574
# i 38 more rows
```

```r
# Distribution Plot
smb_dist |>
  ggplot(aes(x = size_premium)) +
  geom_histogram(bins = nrow(smb_dist)) +
  labs(
    x = NULL, y = NULL,
    title = "Distribution of size premiums for different sorting choices"
  ) +
  geom_vline(aes(xintercept = mean(factors_ff5_monthly$smb)),
             linetype = "dashed"
  ) +
  scale_x_continuous(labels = percent)
```



Below we can observe that the combination closest to the estimation of Fama and French, is for 2 portfolios, breakpoints calculated on “NYSE”, weighted value and excluding the Finance industry.

```r
# Combination closest to Fama and French
ind = which.min(abs(smb_dist$size_premium-mean(factors_ff5_monthly$smb)))
smb_dist %>% slice(ind)
```

```
# A tibble: 1 x 5
```

```
  n_portfolios exchanges value_weighted data                             size_premium
         <dbl> <list>    <lgl>          <chr>                                   <dbl>
1            2 <chr [1]> FALSE          "filter(crsp_monthly, mont~      0.00213
```

We would also like to examine the average size premium based on the listing exchange on which the breakpoints are calculated.

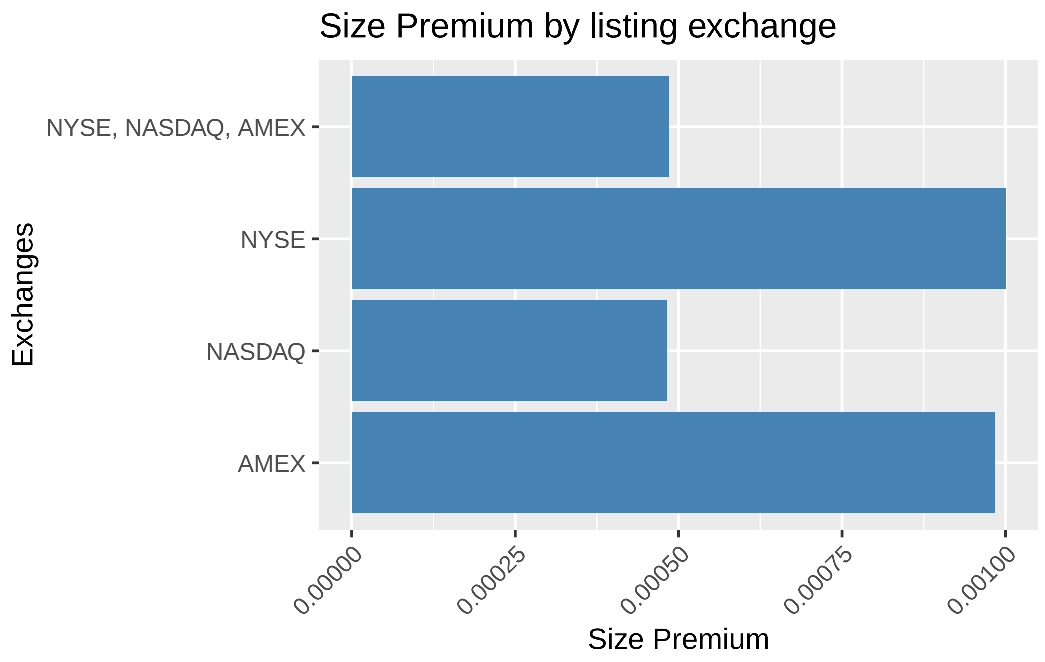
```
choice_grid_ex <- expand_grid(
  n_portfolios = c(2),
  exchanges = list("NYSE", "NASDAQ","AMEX",c("NYSE", "NASDAQ", "AMEX")),
  value_weighted = c(TRUE),
  data = parse_exprs('crsp_monthly |> filter(month >= "1990-06-01")')
  )
```

```
choice_grid_ex <- choice_grid_ex |>
  mutate(size_premium = future_pmap(
    .l = list(
      n_portfolios,
      exchanges,
      value_weighted,
      data
    ),
    .f = ~ compute_portfolio_returns(
      n_portfolios = ..1,
      exchanges = ..2,
      value_weighted = ..3,
      data = eval_tidy(..4)
    )
  ))

smb_dist_ex <- choice_grid_ex |>
  mutate(data = map_chr(data, deparse)) |>
  unnest(size_premium) |>
  arrange(desc(size_premium))
```

```
smb_dist_ex$exchanges[3]=paste(smb_dist_ex$exchanges[[3]], collapse=', ' )
```

```
smb_dist_ex %>%
  mutate(exchanges = unlist(exchanges)) %>%
  ggplot(aes(x=exchanges, y=size_premium)) +
```

```r
  geom_bar(stat="identity",fill="steelblue") +
  labs(title = "Size Premium by listing exchange", x = "Exchanges", y = "Size Premium") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  coord_flip()
```



2.2 Replicating `Fama-French` method

- Splitting the stocks in the median of the firm size distribution as portrayed by their corresponding market equity.
- Portfolio formation in June for every year t and therefore use the market capitalization of June the firm size which remains unchanged until June of the following year.
- Lastly, the breakpoints are based only of the stocks of NYSE exchange.

```r
size <- crsp_monthly |>
  filter(month(month) == 6) |>
  mutate(sorting_date = month %m+% months(1)) |>
  select(permno, exchange, sorting_date, size = mktcap)

portfolios_size <- size |>
  group_by(sorting_date) |>
  mutate(
    portfolio_size = assign_portfolio(
      data = pick(everything()),
      sorting_variable = size,
      n_portfolios = 2,
      exchanges = 'NYSE'
    )
  ) |>
  ungroup() |>
  select(permno, sorting_date,
         portfolio_size)
```

```
portfolios_size <- crsp_monthly |>
  mutate(sorting_date = case_when(
    month(month) <= 6 ~ ymd(str_c(year(month) - 1, "0701")),
    month(month) >= 7 ~ ymd(str_c(year(month), "0701"))
  )) |>
  inner_join(portfolios_size, by = c("permno", "sorting_date"))
```

2.2.1 Summarize Size as a Premium

```
factors_size <- portfolios_size |>
  group_by(portfolio_size, month) |>
  summarize(
    ret = weighted.mean(ret_excess, mktcap_lag), .groups = "drop"
  ) |>
  group_by(month) |>
  summarize(
    smb_replicated = mean(ret[portfolio_size == 1]) -
      mean(ret[portfolio_size == 2])
  )
```

2.2.2 Replication Performance Evaluation

In the previous section, we replicated size premiums following the procedure outlined by Fama and French. To test our replication performance, we look at the two time-series estimates in a regression analysis using `lm()`. If we did a good job, then we should see a non-significant intercept (rejecting the notion of systematic error), a coefficient close to 1 (indicating a high correlation), and an adjusted R-squared close to 1 (indicating a high proportion of explained variance).

```r
test <- factors_size |>
  left_join(factors_ff5_monthly,by="month")

model_size <- lm(smb_replicated ~ smb, data = test)
summary(model_size)
```

```

Call:
lm(formula = smb_replicated ~ smb, data = test)

Residuals:
       Min         1Q     Median         3Q        Max 
-0.0176504 -0.0022510  0.0000575  0.0022781  0.0149165 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.0004528  0.0001557  -2.908  0.00375 ** 
smb          1.0532705  0.0051491 204.555  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.00415 on 712 degrees of freedom
  (36 observations deleted due to missingness)
Multiple R-squared:  0.9833,	Adjusted R-squared:  0.9832 
F-statistic: 4.184e+04 on 1 and 712 DF,  p-value: < 2.2e-16
```

The results for the SMB factor are really convincing as all three criteria outlined above are met and the R-squared is at 99%.

Factor 3. Value

Definition

Using firm value, book-to-market ratio, defined as ratio of book equity to market equity (BE/ME), as a sorting variable, the returns that are created by following the strategy of buying/going long on high value stocks and selling/going short on low value ones, are known as the value premium. Therefore, the value premium is defined as:

$$\text{Value Premium} = \text{HML} = \text{High minus Low}$$

The reason this formula is used will be demonstrated through following replication of Fama-French value factor.

3.1 Load and Prepare Value Data

Fama and French form their portfolios in June of year t, whereby the returns of July are the first monthly return for the respective portfolio.

As mentioned earlier, Value is the ratio of book equity to market equity.

But the book equity value used in the calculation is taken from the company's financial statements, which are based on the fiscal year-end data. However, these financial statements are usually published a few months after the fiscal year ends. For example, the book equity from the financial statements of December 2022 may only be available in mid-2023.

To implement all these time lags, we employ the temporary sorting_date-column.

```r
market_equity <- crsp_monthly |>
  filter(month(month) == 12) |>
  mutate(sorting_date = ymd(str_c(year(month) + 1, "0701)"))) |>
  select(permno, gvkey, sorting_date, me = mktcap)
```

Notice that the investment data and profitability data, which will be introduced in the following sections, are taken from accounting data as well. To address this timing discrepancy, we will employ the same methodology in the computation of investment premium and profitability premium.

```r
book_to_market <- compustat |>
  mutate(sorting_date = ymd(str_c(year(datadate) + 1, "0701"))) |>
  select(gvkey, sorting_date, be) |>
  inner_join(market_equity, by = c("gvkey", "sorting_date")) |>
  mutate(bm = be / me) |>
  select(permno, sorting_date, me, bm)

value_sorting_variables <- size |>
  inner_join(
    book_to_market, by = c("permno", "sorting_date")
    ) |>
  drop_na() |>
  distinct(permno, sorting_date, .keep_all = TRUE)
```

3.2 Replicating `Fama-French` method

Fama-French use size-dependent bivariate sort for value. We form two portfolios in the size dimension at the median and three portfolios in the dimension of each other sorting variable at the 33%- and 66%-percentiles.

```r
value_sorting_variables <- size |>
  inner_join(book_to_market, by = c("permno", "sorting_date")) |>
  drop_na() |>
  distinct(permno, sorting_date, .keep_all = TRUE)

portfolios_value <- value_sorting_variables |>
  group_by(sorting_date) |>
  mutate(
    portfolio_size = assign_portfolio(
      data = pick(everything()),
      sorting_variable = size,
      n_portfolios = 2,
      exchanges = 'NYSE'
    ),
    portfolio_bm = assign_portfolio(
      data = pick(everything()),
      sorting_variable = bm,
      n_portfolios = 3,
      exchanges = 'NYSE'
    )
  ) |>
  ungroup() |>
  select(permno, sorting_date,
         portfolio_size, portfolio_bm)
```

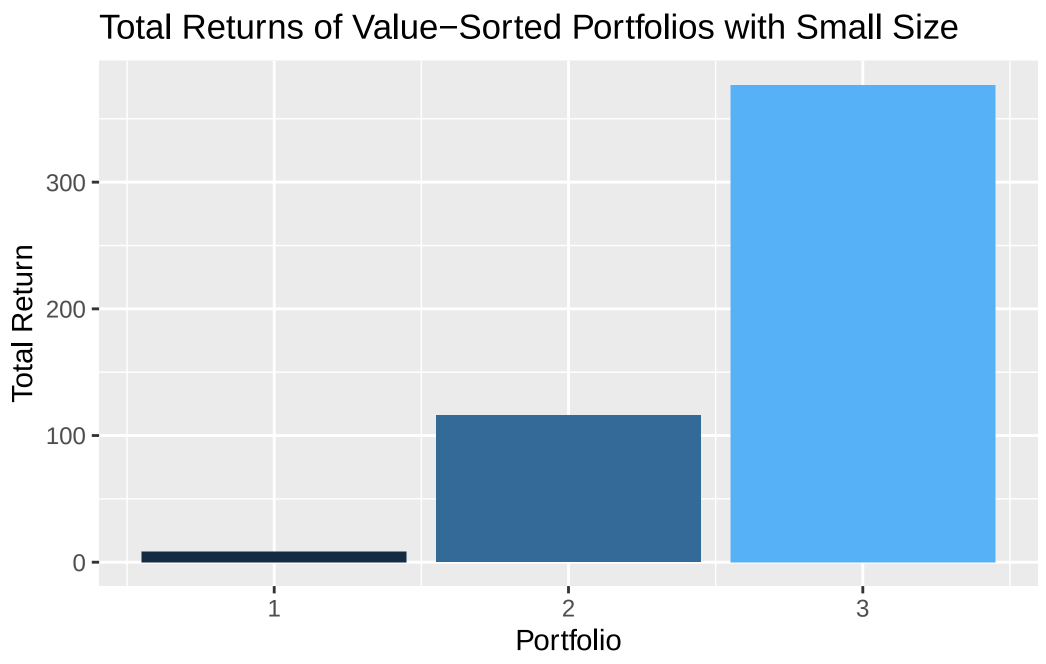
```r
# Merge the portfolios to the return data for the rest of the year
portfolios_value <- crsp_monthly |>
  mutate(sorting_date = case_when(
    month(month) <= 6 ~ ymd(str_c(year(month) - 1, "0701")),
    month(month) >= 7 ~ ymd(str_c(year(month), "0701"))
  )) |>
  inner_join(portfolios_value, by = c("permno", "sorting_date"))

# Compute return
portfolios_value <- portfolios_value |>
```

```
  group_by(portfolio_size, portfolio_bm, month) |>
  summarize(ret = weighted.mean(ret_excess, mktcap_lag), .groups = "drop")
```

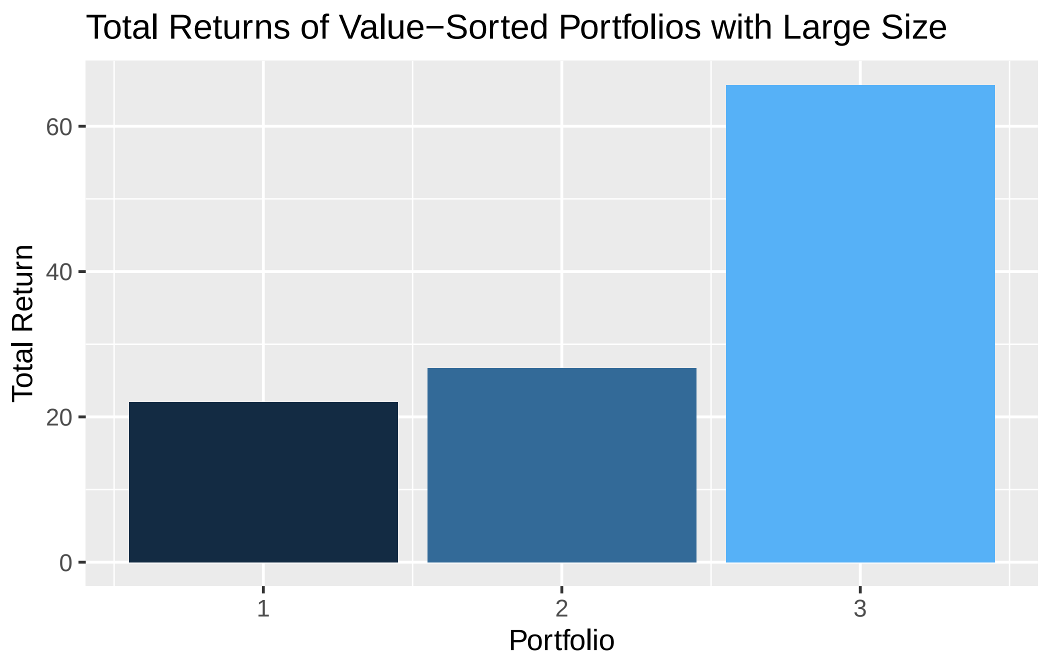
3.2.1 Interpretation

To examine the relation between value and future stock returns, we group the data based on portfolio size and book-to-market ratio, and then calculating the total return for each group.

```
value_summary <- portfolios_value |>
  group_by(portfolio_size, portfolio_bm) |>
  summarize(total_return = prod(1 + ret) - 1)

# group by size
value_summary_size1 <- value_summary |>
  filter(portfolio_size == 1)

value_summary_size2 <- value_summary |>
  filter(portfolio_size == 2)

# plot
value_summary_size1 |>
  ggplot(aes(x = portfolio_bm, y = total_return, fill = portfolio_bm)) +
  geom_bar(stat = "identity") +
  labs(title = "Total Returns of Value-Sorted Portfolios with Small Size",
       x = "Portfolio",
       y = "Total Return",
       fill = "Portfolio") +
  theme(legend.position = "None")
```



```r
value_summary_size2 |>
  ggplot(aes(x = portfolio_bm, y = total_return, fill = portfolio_bm)) +
  geom_bar(stat = "identity") +
  labs(title = "Total Returns of Value-Sorted Portfolios with Large Size",
       x = "Portfolio",
       y = "Total Return",
       fill = "Portfolio") +
  theme(legend.position = "None")
```



These results suggest a positive relation between value and future stock returns. After going long the portfolio of the highest book-to-market firms and short the five portfolios of the lowest book-to-market firms, this approach capitalizes on the expected higher returns of value stocks compared to growth stocks.

3.2.2 Summarize Value as a Premium

```
factors_value <- portfolios_value |>
  group_by(month) |>
  summarize(hml_replicated = mean(ret[portfolio_bm == 3]) - mean(ret[portfolio_bm == 1]))
```

3.2.3 Replication Performance Evaluation

We use the same performance evaluation introduced in size factor section to test our replication for value premium.

```
test_value <- factors_ff5_monthly |>
  inner_join(factors_value, by = "month")

model_hml <- lm(hml ~ hml_replicated, data = test_value)
summary(model_hml)
```

```
Call:
lm(formula = hml ~ hml_replicated, data = test_value)

Residuals:
      Min         1Q    Median        3Q       Max 
-0.022639 -0.003301 -0.000078  0.002935  0.034079 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)    0.0002338  0.0002231   1.048    0.295    
hml_replicated 1.0063762  0.0076804 131.032   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.005932 on 712 degrees of freedom
Multiple R-squared:  0.9602,    Adjusted R-squared:  0.9601 
F-statistic: 1.717e+04 on 1 and 712 DF,  p-value: < 2.2e-16
```

The results for the HML factor are really convincing as all three criteria outlined above are met and the R-squared is at 97%.

3.3 Compute Value Premium through an Univariate Sort

In order to examine the influence of different sort method, we will try to compute the value premium through an univariate sort in the following code. We still form three portfolios in the value dimension at the 33%- and 66%-percentiles but without using size the dependent variable.

```r
portfolios2 <- value_sorting_variables |>
  group_by(sorting_date) |>
  mutate(
    portfolio_bm2 = assign_portfolio(
      data = pick(everything()),
      sorting_variable = bm,
      n_portfolios = 3,
      exchanges = 'NYSE'
    )
  ) |>
  ungroup() |>
  select(permno, sorting_date, portfolio_bm2)
```

```
# Merge the portfolios to the return data for the rest of the year.
portfolios2 <- crsp_monthly |>
  mutate(sorting_date = case_when(
    month(month) <= 6 ~ ymd(str_c(year(month) - 1, "0701")),
    month(month) >= 7 ~ ymd(str_c(year(month), "0701"))
  )) |>
  inner_join(portfolios2, by = c("permno", "sorting_date"))

# compute return
portfolios_value2 <- portfolios2 |>
  group_by(portfolio_bm2, month) |>
  summarize(ret = weighted.mean(ret_excess, mktcap_lag), .groups = "drop")

factors_value2 <- portfolios_value2 |>
  group_by(month) |>
  summarize(
    hml_replicated2 = mean(ret[portfolio_bm2 == 3]) - mean(ret[portfolio_bm2 == 1]))
```

Factor 4. Investment

Definition

The investment factor (CMA) reflects the returns associated with firms' investment decisions. It captures the spread in returns between firms with conservative investment policies and those with aggressive ones. The investment factor (CMA) is defined as follows:

$$\text{Investment Premium} = \text{CMA} = \text{Conservative minus Aggresive}$$

Here's a breakdown of the components:

- **Conservative Firms**: These are companies that follow conservative investment policies, meaning they prioritize stability and safety in their investments, often exhibiting lower investment levels.
- **Aggressive Firms**: These represent companies with aggressive investment policies, characterized by higher investment levels and potentially riskier ventures.

The CMA factor essentially quantifies the returns generated by favoring stocks of firms considered conservative in their investment approach while shorting stocks of firms with more aggressive investment strategies. It's a measure of the spread in returns between these two types of companies, forming a factor used in financial models to explain variations in stock returns.

4.1 Load and Prepare Investment data

```r
inv_sorting_variables <- compustat |>
  mutate(sorting_date = ymd(str_c(year(datadate) + 1, "0701"))) |>
  select(gvkey, sorting_date, be, inv) |>
  inner_join(market_equity, by = c("gvkey", "sorting_date")) |>
  mutate(bm = be / me) |>
  select(permno, sorting_date, me, be, bm, inv)

inv_sorting_variables <- size |>
  inner_join(inv_sorting_variables, by = c("permno", "sorting_date")) |>
  drop_na() |>
  distinct(permno, sorting_date, .keep_all = TRUE)
```

4.2 Replicating `Fama-French` method

Fama-French use size-dependent bivariate sort for investment. We form two portfolios in the size dimension at the median and three portfolios in the dimension of each other sorting variable at the 33%- and 66%-percentiles.

```r
portfolios_inv <- inv_sorting_variables |>
  group_by(sorting_date) |>
  mutate(
    portfolio_size = assign_portfolio(
      data = pick(everything()),
      sorting_variable = size,
      n_portfolios = 2,
      exchanges = 'NYSE'
    )) |>
  group_by(sorting_date, portfolio_size) |>
  mutate(
    portfolio_inv = assign_portfolio(
      data = pick(everything()),
      sorting_variable = inv,
      n_portfolios = 3,
      exchanges = 'NYSE'
    )
  ) |>
  ungroup() |>
  select(permno, sorting_date,
          portfolio_size, portfolio_inv)
```

```
portfolios_inv <- crsp_monthly |>
  mutate(sorting_date = case_when(
    month(month) <= 6 ~ ymd(str_c(year(month) - 1, "0701")),
    month(month) >= 7 ~ ymd(str_c(year(month), "0701"))
  )) |>
  inner_join(portfolios_inv, by = c("permno", "sorting_date"))

inv_portfolios <- portfolios_inv |>
  group_by(portfolio_size, portfolio_inv, month) |>
  summarize(ret = weighted.mean(ret_excess, mktcap_lag), .groups = "drop")
```

4.2.2 Summarize Investment as a Premium

```
factors_inv <- inv_portfolios |>
  group_by(month) |>
  summarize(cma_replicated = mean(ret[portfolio_inv == 1]) - mean(ret[portfolio_inv == 3])
```

4.2.3 Replication Performance Evaluation

```
test_inv <- factors_ff5_monthly |>
  inner_join(factors_inv, by = "month")

model_inv <- lm(cma ~ cma_replicated, data = test_inv)
summary(model_inv)
```

```
Call:
lm(formula = cma ~ cma_replicated, data = test_inv)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0149405 -0.0030232 -0.0002845  0.0028176  0.0207878

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.0007229  0.0001776    4.07 5.23e-05 ***
cma_replicated 1.0114375  0.0089205  113.38  < 2e-16 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.004715 on 712 degrees of freedom
Multiple R-squared:  0.9475,    Adjusted R-squared:  0.9474
F-statistic: 1.286e+04 on 1 and 712 DF,  p-value: < 2.2e-16
```

The results for the CMA factor are really convincing as all three criteria outlined above are met and the R-squared is at 95%.

4.3 Compute Investment Premium through an Univariate Sort

In order to examine the influence of different sort method, we will try to compute the investment premium through an univariate sort in the following code. We still form three portfolios in the value dimension at the 33%- and 66%-percentiles but without using size the dependent variable.

```r
inv_portfolios2 <- inv_sorting_variables |>
  group_by(sorting_date) |>
  mutate(
    portfolio_inv2 = assign_portfolio(
      data = pick(everything()),
      sorting_variable = inv,
      n_portfolios = 3,
      exchanges = 'NYSE'
    )
  ) |>
  ungroup() |>
  select(permno, sorting_date, portfolio_inv2)

# Merge the data
inv_portfolios2 <- crsp_monthly |>
  mutate(sorting_date = case_when(
    month(month) <= 6 ~ ymd(str_c(year(month) - 1, "0701")),
    month(month) >= 7 ~ ymd(str_c(year(month), "0701"))
  )) |>
  inner_join(inv_portfolios2, by = c("permno", "sorting_date"))

# Compute return
inv_portfolios2 <- inv_portfolios2 |>
  group_by(portfolio_inv2, month) |>
  summarize(ret = weighted.mean(ret_excess, mktcap_lag), .groups = "drop")

factors_inv2 <- inv_portfolios2 |>
  group_by(month) |>
  summarize(cma_replicated2 = mean(ret[portfolio_inv2 == 3]) - mean(ret[portfolio_inv2 ==
```

Factor 5. Profitability

Definition

Based on Fama-French, operating profitability (OP) for June of year t is annual revenues minus cost of goods sold, interest expense, and selling, general, and administrative expenses divided by book equity for the last fiscal year end in t-1. (source)

The Profitability Premium is defined as:

$$\text{Profitability Premium} = \text{RMW} = \text{Robust minus Weak}$$

The relation between operating profitability and return will be presented through following replication of Fama-French profitability factor.

5.1 Load and Prepare Profitability data

```
op_sorting_variables <- compustat |>
  mutate(sorting_date = ymd(str_c(year(datadate) + 1, "0701"))) |>
  select(gvkey, sorting_date, be, op) |>
  inner_join(market_equity, by = c("gvkey", "sorting_date")) |>
  mutate(bm = be / me) |>
  select(permno, sorting_date, me, be, bm, op)

op_sorting_variables <- size |>
  inner_join(op_sorting_variables, by = c("permno", "sorting_date")) |>
  drop_na() |>
  distinct(permno, sorting_date, .keep_all = TRUE)
```

5.2 Replicating `Fama-French` method

Similar to value premium, Fama-French use size-dependent bivariate sort for profitability. We form two portfolios in the size dimension at the median and three portfolios in the dimension of each other sorting variable at the 33%- and 66%-percentiles.

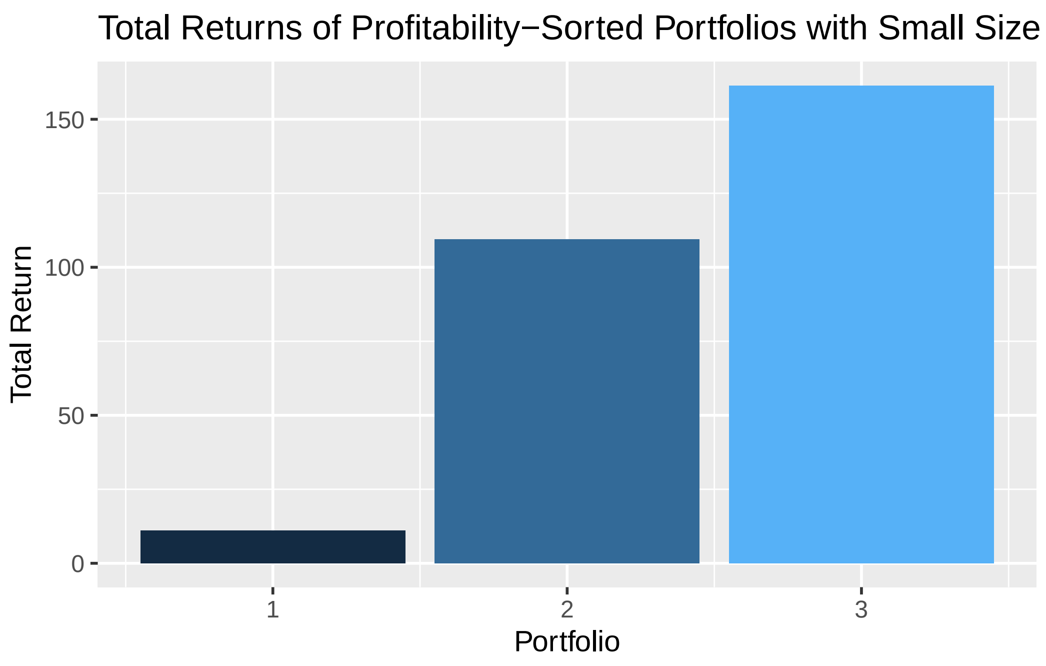
```r
portfolios_op <- op_sorting_variables |>
  group_by(sorting_date) |>
  mutate(
    portfolio_size = assign_portfolio(
      data = pick(everything()),
      sorting_variable = size,
      n_portfolios = 2,
      exchanges = 'NYSE'
    )) |>
  group_by(sorting_date, portfolio_size) |>
  mutate(
    portfolio_op = assign_portfolio(
      data = pick(everything()),
      sorting_variable = op,
      n_portfolios = 3,
      exchanges = 'NYSE'
    )
  ) |>
  ungroup() |>
  select(permno, sorting_date,
         portfolio_size, portfolio_op)
```

```r
portfolios_op <- crsp_monthly |>
  mutate(sorting_date = case_when(
    month(month) <= 6 ~ ymd(str_c(year(month) - 1, "0701")),
    month(month) >= 7 ~ ymd(str_c(year(month), "0701"))
  )) |>
  inner_join(portfolios_op, by = c("permno", "sorting_date"))

portfolios_op <- portfolios_op |>
  group_by(portfolio_size, portfolio_op, month) |>
  summarize(ret = weighted.mean(ret_excess, mktcap_lag), .groups = "drop")
```

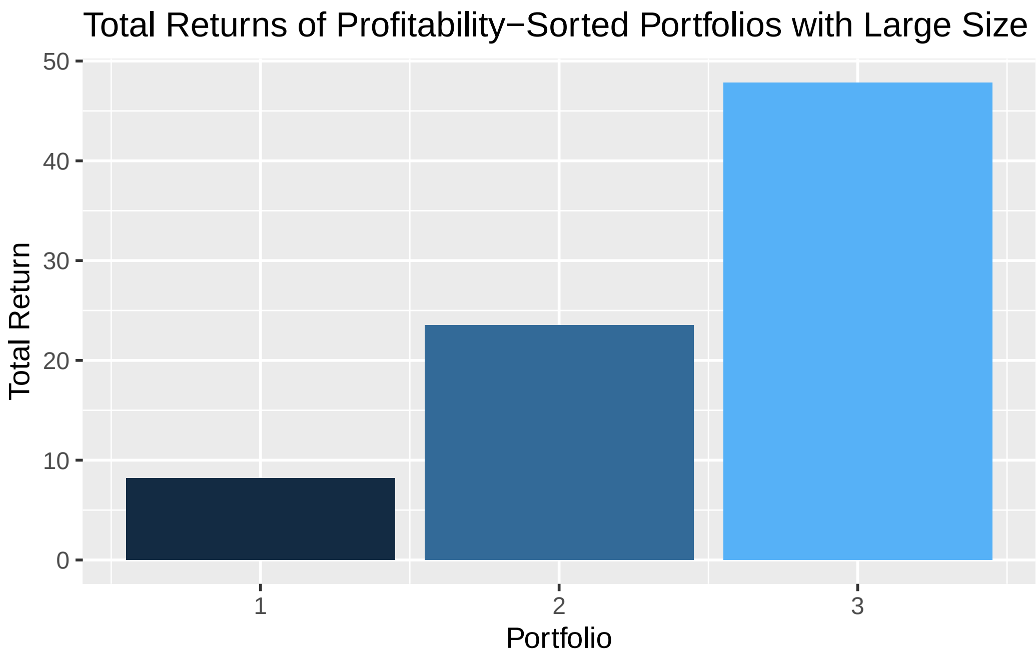
5.2.1 Interpretation

To examine the relation between profitability and future stock returns, we group the data based on portfolio size and profitability, and then calculating the total return for each group.

```r
op_summary <- portfolios_op |>
  group_by(portfolio_size, portfolio_op) |>
  summarize(total_return = prod(1 + ret) - 1)

# group by size
op_summary_size1 <- op_summary |>
  filter(portfolio_size == 1)

op_summary_size2 <- op_summary |>
  filter(portfolio_size == 2)

# plot
op_summary_size1 |>
  ggplot(aes(x = portfolio_op, y = total_return, fill = portfolio_op)) +
  geom_bar(stat = "identity") +
  labs(title = "Total Returns of Profitability-Sorted Portfolios with Small Size",
       x = "Portfolio",
       y = "Total Return",
       fill = "Portfolio") +
  theme(legend.position = "None")
```



```r
op_summary_size2 |>
  ggplot(aes(x = portfolio_op, y = total_return, fill = portfolio_op)) +
  geom_bar(stat = "identity") +
  labs(title = "Total Returns of Profitability-Sorted Portfolios with Large Size",
       x = "Portfolio",
       y = "Total Return",
       fill = "Portfolio") +
  theme(legend.position = "None")
```



These results suggest a positive relation between profitability and future stock returns. After going long the portfolio of the highest operating profitability firms and short the portfolios of the lowest operating profitability firms, this trading approach capitalizes on the expected higher returns of high operating profitability portfolio compared to low operating profitability portfolio.

5.2.2 Summarize Profitability as a Premium

```
factors_op <- portfolios_op |>
  group_by(month) |>
  summarize(rmw_replicated = mean(ret[portfolio_op == 3]) - mean(ret[portfolio_op == 1]))
```

5.2.3 Replication Performance Evaluation

In the previous section, we replicated profitability premiums following the procedure outlined by Fama and French. To test our replication performance, we look at the two time-series estimates in a regression analysis using `lm()`. If we did a good job, then we should see a non-significant intercept (rejecting the notion of systematic error), a coefficient close to 1 (indicating a high correlation), and an adjusted R-squared close to 1 (indicating a high proportion of explained variance).

```r
test_op <- factors_ff5_monthly |>
  inner_join(factors_op, by = "month")

model_rmw <- lm(rmw ~ rmw_replicated, data = test_op)
summary(model_rmw)
```

```

Call:
lm(formula = rmw ~ rmw_replicated, data = test_op)

Residuals:
       Min         1Q     Median         3Q        Max 
-0.0195192 -0.0033275  0.0001026  0.0034171  0.0165524 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)    7.304e-05  2.059e-04   0.355    0.723    
rmw_replicated 1.014e+00  9.617e-03 105.396   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.005457 on 712 degrees of freedom
Multiple R-squared:  0.9398,	Adjusted R-squared:  0.9397 
F-statistic: 1.111e+04 on 1 and 712 DF,  p-value: < 2.2e-16
```

The results for the RMW factor are really convincing as all three criteria outlined above are met and the R-squared is at 94%.

5.3 Compute Profitability Premium through an Univariate Sort

```r
portfolios_op2 <- op_sorting_variables |>
  group_by(sorting_date) |>
  mutate(
    portfolio_op2 = assign_portfolio(
      data = pick(everything()),
      sorting_variable = op,
      n_portfolios = 3,
      exchanges = 'NYSE'
    )
  ) |>
  ungroup() |>
  select(permno, sorting_date, portfolio_op2)

# merge the data
portfolios_op2 <- crsp_monthly |>
  mutate(sorting_date = case_when(
    month(month) <= 6 ~ ymd(str_c(year(month) - 1, "0701")),
    month(month) >= 7 ~ ymd(str_c(year(month), "0701"))
  )) |>
  inner_join(portfolios_op2, by = c("permno", "sorting_date"))

# compute return
portfolios_op2 <- portfolios_op2 |>
  group_by(portfolio_op2, month) |>
  summarize(ret = weighted.mean(ret_excess, mktcap_lag), .groups = "drop")

factors_op2 <- portfolios_op2 |>
  group_by(month) |>
  summarize(rmw_replicated2 = mean(ret[portfolio_op2 == 3]) - mean(ret[portfolio_op2 == 1]
```

Factor 6. Momentum

Momentum, in financial terms, is the tendency of an asset's price to persist in its direction over a certain period.

Carhart introduced this factor, on top of the factors of the 3-factor Fama and French model, as the mean returns of the strategy of going long on momentum and short on the other (contrarian) stocks.

$$\text{WML} = \text{Winner Minus Loser}$$

Carhart computed the momentum factor PR1YR, named WML (Winners minus Losers) by Fama and French, as

"The equal-weight average of firms with the highest 30 percent eleven-month returns lagged one month minus the equal-weight average of firms with the lowest 30 percent eleven-month returns lagged one month. The portfolios include all NYSE, Amex, and Nasdaq stocks and are re-formed monthly."

Motivation

Carhart's motivation to introduce this factor was the fact that the 3-factor model was unable to capture the cross-sectional variation in momentum-sorted portfolio returns. (source, source)

6.1 Load and Prepare Momentum data

We will use the momentum premium stored in french data library to evaluate our replication.

```
# Download the Momentum estimation by Fama and French
factors_mom_monthly <- download_french_data("Momentum Factor (Mom)")
```

```
New names:
New names:
* `` -> `...1`
```

```
factors_mom_monthly <- factors_mom_monthly$subsets$data[[1]]

factors_mom_monthly <- factors_mom_monthly %>%
  mutate(month=as.Date(paste0(as.character(date), '01'), format='%Y%m%d')) %>%
  filter(date>=min(crsp_monthly$month))
```

6.2 Replicating `Fama-French` method

Fama-French use size-independent bivariate sort for value.

```
# Replication
mom <- crsp_monthly %>%
  group_by(permno) %>%
  arrange(month, .by_group=T) %>%
  mutate(
    sorting_date = month %m+% months(1),
    ret = ret_excess,
    ret_cum_0_10 = roll_prod(1 + ret, width = 11)-1,
    ret_cum_2_12 = dplyr::lag(ret_cum_0_10, n=2)) |>
  select(permno, sorting_date, wml = ret_cum_2_12)

me <- crsp_monthly |>
  mutate(sorting_date = month %m+% months(1)) |>
  select(permno, sorting_date, me = mktcap)

mom_df <- crsp_monthly |>
  left_join(
    mom, by = c("permno", "month" = "sorting_date")) |>
  left_join(
    me, by = c("permno", "month" = "sorting_date")) |>
  select(
    permno, gvkey, month, ret_excess,
    mktcap_lag, me, wml, exchange) |>
  drop_na()
```

```
portfolios_mom <- mom_df |>
  group_by(month) |>
  mutate(
    portfolio_mm = assign_portfolio(
      data = pick(everything()),
      sorting_variable = "wml",
      n_portfolios = 3,
      exchanges = c("NYSE")),
    portfolio_me = assign_portfolio(
      data = pick(everything()),
      sorting_variable = "me",
      n_portfolios = 2,
      exchanges = c("NYSE"))) |>
  group_by(month, portfolio_mm, portfolio_me) |>
  summarize(
    ret = weighted.mean(ret_excess, mktcap_lag),
    .groups = "drop")
```

6.2.1 Summarize Momentum as a Premium

```
factors_mom <- portfolios_mom |>
  group_by(month, portfolio_mm) |>
  summarize(ret = mean(ret), .groups = "drop_last") |>
  summarize(
    wml_replicated = ret[portfolio_mm == max(portfolio_mm)] -
      ret[portfolio_mm == min(portfolio_mm)]
  )
```

6.2.2 Replication Performance Evaluation

```
# Test Replication
test_mom <- factors_mom |>
          left_join(factors_mom_monthly, by = "month")
model_mom <- lm(Mom ~ wml_replicated, data = test_mom)
summary(model_mom)
```

```
Call:
lm(formula = Mom ~ wml_replicated, data = test_mom)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.4624 -0.5145 -0.0244  0.5603  4.6370 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)      0.07846    0.03995   1.964   0.0499 *  
wml_replicated 109.39238    1.08286 101.022   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.077 on 740 degrees of freedom
Multiple R-squared:  0.9324,	Adjusted R-squared:  0.9323 
F-statistic: 1.021e+04 on 1 and 740 DF,  p-value: < 2.2e-16
```

The results for the WML factor are really convincing as all three criteria outlined above are met and the R-squared is at 93%.

7. Model Validation

7.1 A Summary of Factor Premium Variable Name

- Market: factors_market

- Size: factors_size (Fama-French Method - Univariate Sort)
- Value: factors_value (Fama French Method - Bivariate sort using size as dependent variable); factors_value2 (Univariate sort)
- Investment: factors_inv (Fama-French Method – Bivariate sort using size as dependent variable); factors_inv2 (Univariate sort)
- Profitability: factors_op (Fama-French Method – Bivariate sort using size as dependent variable); factors_op2 (Univariate sort)
- Momentum: factors_mom (Carhart's Method - Univariate Sort)

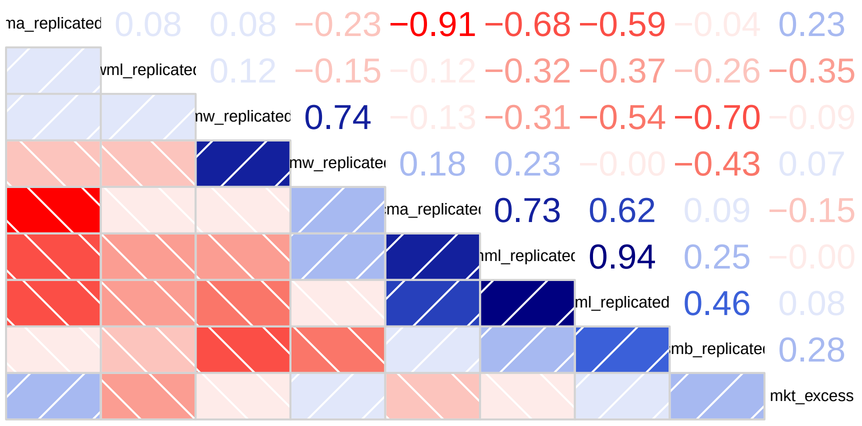
7.2 Examine the Correlation Between Factors

7.2.1 Correlation Matrix

```
# join all the premium together
full_joined_factors <- factors_market |>
  inner_join(factors_size, by = 'month') |>
  inner_join(factors_value, by = 'month') |>
  inner_join(factors_value2, by = 'month') |>
  inner_join(factors_inv, by = 'month') |>
  inner_join(factors_inv2, by = 'month') |>
  inner_join(factors_op, by = 'month') |>
  inner_join(factors_op2, by = 'month') |>
  inner_join(factors_mom, by = 'month')
```

```
# look at last 10 years factor correlations
n_year <- 12 * 10
full_joined_factors |>
  slice(c((n() - n_year):n())) |>
  select(-month) |>
  cor() |>
  round(2) -> rho

rho |>
  corrgram(order = TRUE, upper.panel = panel.cor,
           main = "Correlations of investment factors")
```

Correlations of investment factors



```
rho
```

```
                  mkt_excess smb_replicated hml_replicated hml_replicated2
mkt_excess              1.00           0.28           0.00            0.08
smb_replicated          0.28           1.00           0.25            0.46
hml_replicated          0.00           0.25           1.00            0.94
hml_replicated2         0.08           0.46           0.94            1.00
cma_replicated         -0.15           0.09           0.73            0.62
cma_replicated2         0.23          -0.04          -0.68           -0.59
rmw_replicated          0.07          -0.43           0.23            0.00
rmw_replicated2        -0.09          -0.70          -0.31           -0.54
wml_replicated         -0.35          -0.26          -0.32           -0.37
                  cma_replicated cma_replicated2 rmw_replicated rmw_replicated2
mkt_excess                 -0.15            0.23           0.07           -0.09
smb_replicated              0.09           -0.04          -0.43           -0.70
hml_replicated              0.73           -0.68           0.23           -0.31
hml_replicated2             0.62           -0.59           0.00           -0.54
cma_replicated              1.00           -0.91           0.18           -0.13
cma_replicated2            -0.91            1.00          -0.23            0.08
rmw_replicated              0.18           -0.23           1.00            0.74
rmw_replicated2            -0.13            0.08           0.74            1.00
wml_replicated             -0.12            0.08          -0.15            0.12
                  wml_replicated
mkt_excess                 -0.35
smb_replicated             -0.26
hml_replicated             -0.32
```

```
hml_replicated2          -0.37
cma_replicated           -0.12
cma_replicated2           0.08
rmw_replicated           -0.15
rmw_replicated2           0.12
wml_replicated            1.00
```

7.2.2 Correlation Analysis

1. **Market Excess Return (mkt_excess)**:
    - Shows a moderate positive correlation with SMB Replicated (0.28), suggesting that smaller companies may somewhat respond to market movements.
    - Displays a very weak or negligible correlation with HML Replicated and HML Replicated2, indicating little to no relationship between market excess returns and high book-to-market stocks.
    - Has a slight negative correlation with CMA Replicated (-0.15) and a slight positive correlation with CMA Replicated2 (0.23), implying a mixed relationship with conservative versus aggressive investment strategies.
    - Shows a weak relationship with RMW Replicated and RMW Replicated2 (0.07, -0.09), suggesting profitability factors have limited direct relation with market excess returns.
    - Exhibits a moderate negative correlation with WML Replicated (-0.35), indicating that momentum stocks tend to move differently from overall market trends.
2. **SMB (Small Minus Big) Replicated**:
    - Demonstrates strong positive correlations with HML Replicated2 (0.46) and a moderate correlation with HML Replicated (0.25), indicating small-cap stocks tend to have a relationship with value stocks.
    - Shows a strong negative correlation with RMW Replicated2 (-0.70), suggesting small companies are generally less profitable.
    - Has a negative correlation with WML Replicated (-0.26), suggesting that smaller companies may not always follow momentum trends.
3. **HML (High Minus Low) Replicated and Replicated2**:
    - These value factors show very high consistency between their replicated sorts (correlation of 0.94).

- They have a strong positive correlation with CMA Replicated, suggesting that high book-to-market stocks tend to align with conservative investment strategies.
- Display a negative correlation with CMA Replicated2, indicating differences in the measurement or impact of the conservative-aggressive investment factor.
- Show weak to moderate negative correlations with RMW Replicated and Replicated2, suggesting that value stocks might not be as profitable.

4. **CMA (Conservative Minus Aggressive) Replicated and Replicated2**:
    - Have a very strong negative correlation with each other (-0.91), indicating significant differences in the classifications based on different sorts.
    - Show varying correlations with other factors, reflecting the complexity and variability in how conservative versus aggressive investment strategies are captured and interact with other market dynamics.
5. **RMW (Robust Minus Weak) Replicated and Replicated2**:
    - Exhibit a very high positive correlation between their replicated sorts (0.74), indicating consistency in the profitability factor.
    - The strong negative correlation of RMW Replicated2 with SMB Replicated (-0.70) suggests that more profitable companies are often larger.
    - Display mixed correlations with WML Replicated, suggesting a complex relationship with momentum strategies.
6. **WML (Winner Minus Loser) Replicated**:
    - Shows negative correlations with most other factors, particularly with Market Excess Return (-0.35), SMB Replicated (-0.26), and both HML factors, indicating that momentum stocks often behave independently of size, value, and market excess return factors.

7.3 Model Comparison

We create a 10 industries portfolio and use the data from 2018-01-01 to 2022-12-31 to model each of these industries' excess returns on:

- One-factor model (mkt_excess)
- Three-factor model (mkt_excess, smb, hml)
- Four-factor model (mkt_excess, smb, hml, wml)
- Five-factor model (mkt_excess, smb, hml, rmw, cma)

- Six-factor model (mkt_excess, smb, hml, rmw, cma, wml)
- Univariate six-factor model (factors included are same as six-factor model but hml, rmw, and cma are calculated using the univariate sort)

To see if those factors explain our asset returns, we run linear regressions and extract the adjusted R^2 for each asset by models.

Based on the results of the correlation analysis we also decided to include to the comparison two more models, both include the market factor, but the first has also the value factor while the second one has the investment and the momentum factor.

```r
start_date = '2018-01-01'
end_date = '2022-12-31'

# Format the factors data
global_5_factors <- factors_ff5_monthly |>
  filter(month >= start_date & month <= end_date)
```

```r
returns <- crsp_monthly |>
  filter(month >= start_date & month <= end_date) |>
  group_by(month, industry) |>
  summarise(ret = mean(ret)) |>
  ungroup()
```

```
`summarise()` has grouped output by 'month'. You can override using the
`.groups` argument.
```

```r
# Join the data for analyses
data_joined_tidy <-
    returns %>%
    left_join(full_joined_factors, by = "month") %>%
    na.omit() |>
    group_by(industry) |>
    mutate(returns = ret) |>
    select(-ret)
```

```r
# define models
model_1 <- function(df) {
  lm(returns ~ mkt_excess, data = df)
}
```

```r
model_2 <- function(df) {
  lm(returns ~ mkt_excess + smb_replicated + hml_replicated, data = df)
}

model_3 <- function(df) {
  lm(returns ~ mkt_excess + smb_replicated + hml_replicated + wml_replicated, data = df)
}

model_4 <- function(df) {
  lm(returns ~ mkt_excess + smb_replicated + hml_replicated + rmw_replicated + cma_replica
}

model_5 <- function(df) {
  lm(returns ~ mkt_excess + smb_replicated + hml_replicated + rmw_replicated + cma_replica
}

model_6 <- function(df) {
  lm(returns ~ mkt_excess + smb_replicated + hml_replicated2 + rmw_replicated2 + cma_repli
}

model_7 <- function(df) {
  lm(returns ~ mkt_excess + hml_replicated, data = df)
}

model_8 <- function(df) {
  lm(returns ~ mkt_excess + cma_replicated + wml_replicated, data = df)
}
```

```r
# Perform analysis and extract adjusted R^2, AIC and BIC for each asset by models
models_results <- data_joined_tidy %>%
  group_by(industry) %>%
  nest() %>%
  mutate(one_factor_model = map(data, model_1),
         three_factor_model = map(data, model_2),
         four_factor_model = map(data, model_3),
         five_factor_model = map(data, model_4),
         six_factor_model = map(data, model_5),
         univariate_six_factor_model = map(data, model_6),
         mkt_hml_model = map(data, model_7),
         cma_wml_model = map(data,model_8)) %>%
  select(-data) |>
  pivot_longer(cols = -industry, names_to = "models",
               values_to = "results" ) |>
  mutate(glanced_results = map(results, glance)) %>%
  unnest(glanced_results) %>%
  select(industry, models, adj.r.squared, AIC, BIC)
```

From the results below we can observe that the model indeed works with adjusted R^2 that reaches 96% for industries like Finance and Manufacturing.

```
models_results %>%
  filter(industry == 'Finance')
```

```
# A tibble: 8 x 5
# Groups:   industry [1]
  industry models                      adj.r.squared   AIC   BIC
  <chr>    <chr>                               <dbl> <dbl> <dbl>
1 Finance  one_factor_model                    0.718 -240. -233.
2 Finance  three_factor_model                  0.927 -319. -308.
3 Finance  four_factor_model                   0.940 -329. -317.
4 Finance  five_factor_model                   0.948 -337. -322.
5 Finance  six_factor_model                    0.955 -345. -329.
6 Finance  univariate_six_factor_model         0.946 -334. -318.
7 Finance  mkt_hml_model                       0.829 -269. -260.
8 Finance  cma_wml_model                       0.794 -256. -246.
```

```
models_results %>%
  filter(industry == 'Manufacturing')
```

```
# A tibble: 8 x 5
# Groups:   industry [1]
  industry      models                      adj.r.squared   AIC   BIC
  <chr>         <chr>                               <dbl> <dbl> <dbl>
1 Manufacturing one_factor_model                    0.820 -243. -236.
2 Manufacturing three_factor_model                  0.957 -326. -316.
3 Manufacturing four_factor_model                   0.958 -327. -314.
4 Manufacturing five_factor_model                   0.958 -326. -312.
5 Manufacturing six_factor_model                    0.960 -328. -311.
6 Manufacturing univariate_six_factor_model         0.957 -324. -307.
7 Manufacturing mkt_hml_model                       0.829 -245. -236.
8 Manufacturing cma_wml_model                       0.836 -246. -236.
```

However, for the Agriculture industry the adjusted R^2 is significantly lower since it reaches at maximum just 69%. Also, unlike the previous industries, the four-factor morel seems to be the one performing the best.

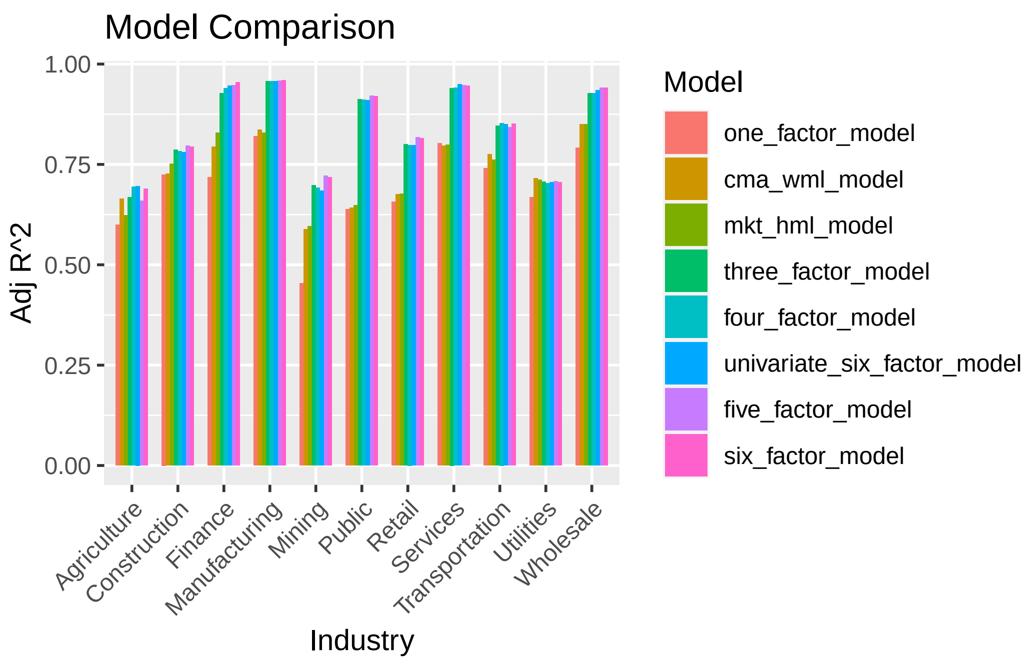
```r
models_results %>%
  filter(industry == 'Agriculture')
```

```
# A tibble: 8 x 5
# Groups:   industry [1]
  industry    models                        adj.r.squared   AIC   BIC
  <chr>       <chr>                                 <dbl> <dbl> <dbl>
1 Agriculture one_factor_model                      0.600 -207. -201.
2 Agriculture three_factor_model                    0.669 -217. -206.
3 Agriculture four_factor_model                     0.694 -221. -208.
4 Agriculture five_factor_model                     0.659 -213. -199.
5 Agriculture six_factor_model                      0.689 -218. -201.
6 Agriculture univariate_six_factor_model           0.696 -219. -203.
7 Agriculture mkt_hml_model                         0.624 -210. -202.
8 Agriculture cma_wml_model                         0.665 -216. -206.
```

Finally, it is worth noting that while the factors introduced by Fama and French significantly improved the results, the difference in adjusted R^2 with the introduction of more than 3 factors is not significant from the following plot, where we can also observe for which industries the models seem to work.

We would also like to mention that the new models suggested, did not offer better results compared to the pre-existing ones.

```r
models_results %>%
  group_by(industry) |>
  filter(industry != "Missing") |>
  ggplot(aes(x = industry, y = adj.r.squared, fill = reorder(models, adj.r.squared))) +
    geom_bar(stat = "identity", position = "dodge", width = 0.7) +
    labs(title = "Model Comparison",
         x = "Industry",
         y = "Adj R^2",
        fill = "Model") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Also, for most of the industries, the univariate sort six factor model's adjusted R^2 is lower than the original six factor model, which suggesting that using the univariate sort will lost some information.

```
models_results |>
   filter(models == 'six_factor_model' | models == 'univariate_six_factor_model') |>
  group_by(industry) |>
  pivot_wider(names_from = models, values_from = adj.r.squared) |>
  group_by(industry) |>
  summarise(six_factor_model = first(na.omit(six_factor_model)),
            univariate_six_factor_model = first(na.omit(univariate_six_factor_model))) |>
  mutate(diff_adj_r_squared = univariate_six_factor_model - six_factor_model) %>%
  select(industry, diff_adj_r_squared)
```

```
# A tibble: 12 x 2
   industry       diff_adj_r_squared
   <chr>                       <dbl>
 1 Agriculture               0.00702
 2 Construction             -0.0136
 3 Finance                  -0.00907
 4 Manufacturing            -0.00280
 5 Mining                   -0.0343
 6 Missing                  -0.00452
 7 Public                   -0.00994
 8 Retail                   -0.0176
 9 Services                  0.00316
10 Transportation           -0.000302
11 Utilities                 0.000271
12 Wholesale                -0.00688
```

Reference

https://www.tidy-finance.org/r/ https://people.duke.edu/~charvey/Teaching/BA453_200

https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1540-6261.1997.tb03808.x

https://www.sciencedirect.com/science/article/abs/pii/S0304405X12000931