Factor Models with R

MAFS6010R- Portfolio Optimization with R MSc in Financial Mathematics Fall 2018-19, HKUST, Hong Kong

Prof. Daniel P. Palomar

Hong Kong University of Science and Technology (HKUST)

- Macroeconomic factor model with single market factor
- Assessing investment funds
- Fundamental factor models: Fama-French
- Statistical factor models
- Final comparison of covariance matrix estimations via different factor models

---

This R session will illustrate the practical implementation and use of factor models.

The R package **covFactorModel**, available in GitHub (https://github.com/dppalomar/covFactorModel), is recommended.

(Useful R links: Cookbook R (http://www.cookbook-r.com/), Quick-R (https://www.statmethods.net/), R documentation (https://www.rdocumentation.org/), CRAN (https://cran.r-project.org/), METACRAN (https://r-pkg.org/).)

Macroeconomic factor model with single market factor

We will start with a simple example consisting of a single known factor (i.e., the market index). The model is

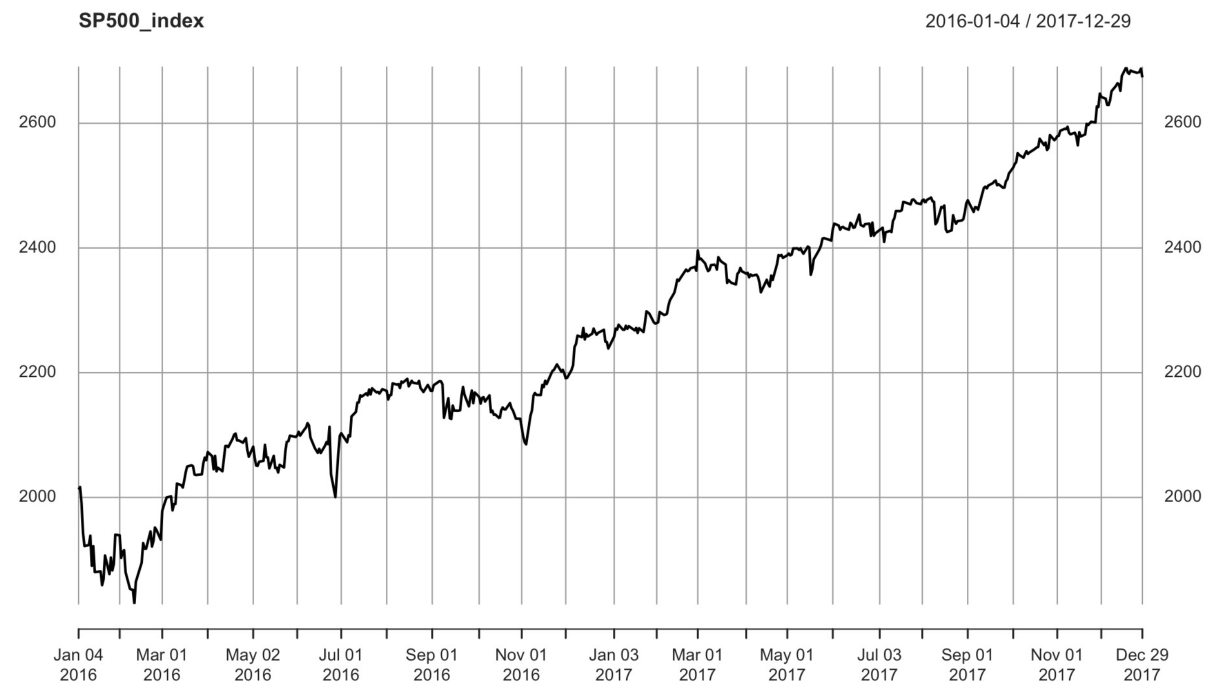
$$\mathbf{x}_t = \boldsymbol{\alpha} + \boldsymbol{\beta} f_t + \boldsymbol{\epsilon}_t, \quad t = 1, \ldots, T$$

where the explicit factor f_t is the S&P 500 index. We will do a simple least squares (LS) regression to estimate the intercept $\boldsymbol{\alpha}$ and the loading beta $\boldsymbol{\beta}$:

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{minimize}} \quad \sum_{t=1}^{T} \|\mathbf{x}_t - \boldsymbol{\alpha} - \boldsymbol{\beta} f_t\|^2$$

Most the code lines go into preparing the data rather than actually performing the factor modeling. Let's start getting the data ready:

```
library(xts)
library(quantmod)

# set begin-end date and stock namelist
begin_date <- "2016-01-01"
end_date <- "2017-12-31"
stock_namelist <- c("AAPL", "AMD", "ADI",  "ABBV", "AET", "A",  "APD", "AA","CF")
sector_namelist <- c(rep("Information Technology", 3), rep("Health Care", 3), rep("Materials", 3))

# download data from YahooFinance
data_set <- xts()
for (stock_index in 1:length(stock_namelist))
  data_set <- cbind(data_set, Ad(getSymbols(stock_namelist[stock_index],
                                            from = begin_date, to = end_date, auto.assign = FALSE)))
colnames(data_set) <- stock_namelist
indexClass(data_set) <- "Date"
str(data_set)
#> An 'xts' object on 2016-01-04/2017-12-29 containing:
#>   Data: num [1:503, 1:9] 100.3 97.8 95.8 91.8 92.3 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : NULL
#>   ..$ : chr [1:9] "AAPL" "AMD" "ADI" "ABBV" ...
#>   Indexed by objects of class: [Date] TZ: UTC
#>   xts Attributes:
#>  NULL
head(data_set)
#>                 AAPL  AMD      ADI     ABBV      AET        A      APD
#> 2016-01-04 100.27451 2.77 51.05404 52.05566 106.0680 39.70486 109.8707
#> 2016-01-05  97.76168 2.75 50.67892 51.83879 107.5435 39.56824 107.9061
#> 2016-01-06  95.84851 2.51 48.52197 51.84784 106.9999 39.74389 105.2782
#> 2016-01-07  91.80328 2.28 47.28407 51.69422 107.0484 38.05577 101.7488
#> 2016-01-08  92.28870 2.14 46.87144 50.28463 103.9419 37.65570 101.2215
#> 2016-01-11  93.78307 2.34 47.98742 48.68528 102.2431 37.02144 101.6212
#>                  AA       CF
#> 2016-01-04 23.00764 36.08226
#> 2016-01-05 21.96506 34.95080
#> 2016-01-06 20.40121 31.93058
#> 2016-01-07 19.59558 30.41601
#> 2016-01-08 19.12169 30.13092
#> 2016-01-11 18.95583 28.91036
tail(data_set)
#>                AAPL   AMD      ADI     ABBV      AET        A      APD
#> 2017-12-21 173.0230 10.89 87.35595 95.24335 179.2424 67.05882 158.9106
#> 2017-12-22 173.0230 10.54 87.55302 95.53518 178.4787 66.88999 159.1638
#> 2017-12-26 168.6334 10.46 87.33624 95.08771 178.9349 66.79068 158.7548
#> 2017-12-27 168.6631 10.53 87.79938 95.41844 179.3614 66.84033 159.4073
#> 2017-12-28 169.1377 10.55 88.07529 95.12663 179.7383 66.98931 160.3325
#> 2017-12-29 167.3086 10.28 87.73040 94.07604 178.9052 66.65984 160.7342
#>               AA       CF
#> 2017-12-21 48.99 40.37555
#> 2017-12-22 49.99 40.82579
#> 2017-12-26 50.38 41.94162
#> 2017-12-27 51.84 42.15696
#> 2017-12-28 54.14 41.70672
#> 2017-12-29 53.87 41.63820

SP500_index <- Ad(getSymbols("^GSPC", from = begin_date, to = end_date, auto.assign = FALSE))
colnames(SP500_index) <- "index"
head(SP500_index)
#>              index
#> 2016-01-04 2012.66
#> 2016-01-05 2016.71
#> 2016-01-06 1990.26
#> 2016-01-07 1943.09
#> 2016-01-08 1922.03
#> 2016-01-11 1923.67
plot(SP500_index)
```



```r
# compute the log-returns of the stocks and of the SP500 index as explicit factor
X <- diff(log(data_set), na.pad = FALSE)
N <- ncol(X)  # number of stocks
T <- nrow(X)  # number of days
f <- diff(log(SP500_index), na.pad = FALSE)
str(X)
#> An 'xts' object on 2016-01-05/2017-12-29 containing:
#>   Data: num [1:502, 1:9] -0.02538 -0.01976 -0.04312 0.00527 0.01606 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : NULL
#>   ..$ : chr [1:9] "AAPL" "AMD" "ADI" "ABBV" ...
#>   Indexed by objects of class: [Date] TZ: UTC
#>   xts Attributes:
#>  NULL
str(f)
#> An 'xts' object on 2016-01-05/2017-12-29 containing:
#>   Data: num [1:502, 1] 0.00201 -0.013202 -0.023986 -0.010898 0.000853 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : NULL
#>   ..$ : chr "index"
#>   Indexed by objects of class: [Date] TZ: UTC
#>   xts Attributes:
#> List of 2
#>  $ src    : chr "yahoo"
#>  $ updated: POSIXct[1:1], format: "2018-09-28 12:31:57"
```

Now we are ready to do the factor model fitting. The solution to the previous LS fitting is

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= \mathrm{cov}(\mathbf{x}_t, f_t)/\mathrm{var}(f_t)\\
\hat{\boldsymbol{\alpha}} &= \bar{x} - \hat{\boldsymbol{\beta}}\bar{f}\\
\hat{\boldsymbol{\epsilon}}_i &= \mathbf{x}_i - \alpha_i\mathbf{1} - \beta_i\mathbf{f}, \quad i = 1, \dots, N\\
\hat{\sigma}_i^2 &= \frac{1}{T-2}\hat{\boldsymbol{\epsilon}}_i^T\hat{\boldsymbol{\epsilon}}_i, \quad \hat{\boldsymbol{\Psi}} = \mathrm{diag}(\hat{\sigma}_1^2, \dots, \hat{\sigma}_\mathrm{N}^2)\\
\hat{\boldsymbol{\Sigma}} &= \mathrm{var}(f_t)\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T + \hat{\boldsymbol{\Psi}}
\end{aligned}$$

which can be readily implemented in R as follows:

```
beta <- cov(X,f)/as.numeric(var(f))
alpha <- colMeans(X) - beta*colMeans(f)
sigma2 <- rep(NA, N)
for (i in 1:N) {
  eps_i <- X[, i] - alpha[i] - beta[i]*f
  sigma2[i] <- (1/(T-2)) * t(eps_i) %*% eps_i
}
Psi <- diag(sigma2)
Sigma <- as.numeric(var(f)) * beta %*% t(beta) + Psi
print(alpha)
#>              index
#> AAPL  0.0003999091
#> AMD   0.0013825599
#> ADI   0.0003609970
#> ABBV  0.0006684635
#> AET   0.0004420515
#> A     0.0002810623
#> APD   0.0001786376
#> AA    0.0006438612
#> CF   -0.0006029692
print(beta)
#>          index
#> AAPL 1.0957906
#> AMD  2.1738304
#> ADI  1.2683036
#> ABBV 0.9022745
#> AET  1.0594902
#> A    1.3277200
#> APD  1.0239452
#> AA   1.8576780
#> CF   1.5702483
```

Alternatively, we can do the fitting using a more compact matrix notation (recall time is along the vertical axis in $\mathbf{X}$)

$$\mathbf{X}^T = \boldsymbol{\alpha}\mathbf{1}^T + \boldsymbol{\beta}\mathbf{f}^T + \mathbf{E}^T$$

and then write the LS fitting as

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{minimize}} \quad \|\mathbf{X}^T - \boldsymbol{\alpha}\mathbf{1}^T - \boldsymbol{\beta}\mathbf{f}^T\|_F^2.$$

More conveniently, we can define $\boldsymbol{\Gamma} = [\boldsymbol{\alpha}, \boldsymbol{\beta}]$ and the extended factors $\tilde{\mathbf{F}} = [\mathbf{1}, \mathbf{f}]$. The LS formulation can then be written as

$$\underset{\boldsymbol{\Gamma}}{\text{minimize}} \quad \|\mathbf{X}^T - \boldsymbol{\Gamma}\tilde{\mathbf{F}}^T\|_F^2$$

with solution

$$\boldsymbol{\Gamma} = \mathbf{X}^T\tilde{\mathbf{F}}(\tilde{\mathbf{F}}^T\tilde{\mathbf{F}})^{-1}$$

```
F_ <- cbind(ones = 1, f)
Gamma <- t(X) %*% F_ %*% solve(t(F_) %*% F_)  # better: Gamma <- t(solve(t(F_) %*% F_, t(F_) %*% X))
colnames(Gamma) <- c("alpha", "beta")
alpha <- Gamma[, 1]  # or alpha <- Gamma[, "alpha"]
beta <- Gamma[, 2]   # or beta <- Gamma[, "beta"]
print(Gamma)
#>              alpha      beta
#> AAPL  0.0003999091 1.0957906
#> AMD   0.0013825599 2.1738304
#> ADI   0.0003609970 1.2683036
#> ABBV  0.0006684635 0.9022745
#> AET   0.0004420515 1.0594902
#> A     0.0002810623 1.3277200
#> APD   0.0001786376 1.0239452
#> AA    0.0006438612 1.8576780
#> CF   -0.0006029692 1.5702483
E <- xts(t(t(X) - Gamma %*% t(F_)), index(X))  # residuals
Psi <- (1/(T-2)) * t(E) %*% E
Sigma <- as.numeric(var(f)) * beta %o% beta + diag(diag(Psi))
```

As expected, we get the same result regardless of the notation used. Alternatively, we can simply use the R package **covFactorModel** to do the work for us:

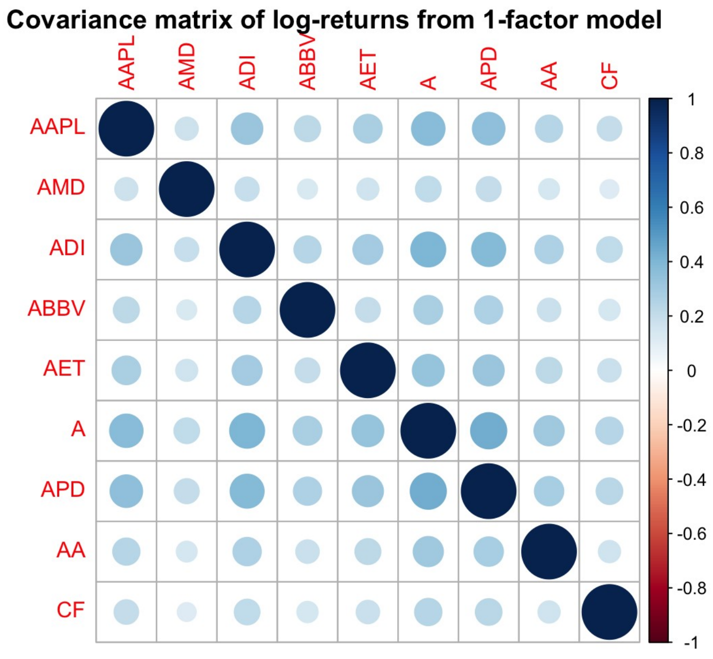
```
library(covFactorModel)
factor_model <- factorModel(X, type = "M", econ_fact = f)
cbind(alpha = factor_model$alpha, beta = factor_model$beta)
#>              alpha     index
#> AAPL  0.0003999091 1.0957906
#> AMD   0.0013825599 2.1738304
#> ADI   0.0003609970 1.2683036
#> ABBV  0.0006684635 0.9022745
#> AET   0.0004420515 1.0594902
#> A     0.0002810623 1.3277200
#> APD   0.0001786376 1.0239452
#> AA    0.0006438612 1.8576780
#> CF   -0.0006029692 1.5702483
```

Visualizing covariance matrices

It is interesting to visualize the estimated covariance matrix of the log-returns $\mathbf{\Sigma} = var(f_t)\boldsymbol{\beta}\boldsymbol{\beta}^T + \text{Diag}(\mathbf{\Psi})$, as well as that of the residuals $\mathbf{\Psi}$.

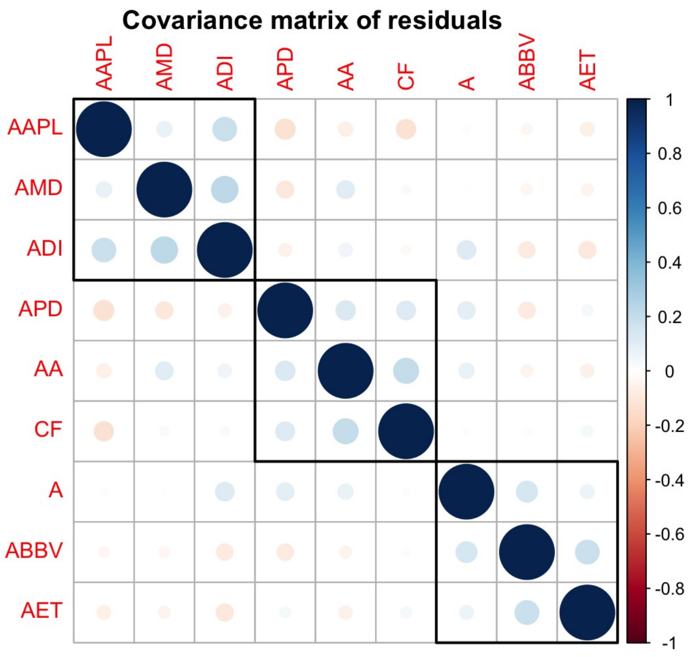
Let's start with the covariance matrix of the log-returns:

```
library(corrplot)  #install.packages("corrplot")
corrplot(cov2cor(Sigma), mar = c(0,0,1,0),
         main = "Covariance matrix of log-returns from 1-factor model")
```



We can observe that all the stocks are highly correlated, which is the effect of the market factor. In order to inspect finer details of the interdependency of the stocks, we should first remove the market or, equivalently, plot $\mathbf{\Psi}$:

```
corrplot(cov2cor(Psi), mar = c(0,0,1,0), order = "hclust", addrect = 3,
         main = "Covariance matrix of residuals")
```



```
cbind(stock_namelist, sector_namelist)  # to recall the sectors
#>       stock_namelist sector_namelist
#>  [1,] "AAPL"         "Information Technology"
#>  [2,] "AMD"          "Information Technology"
#>  [3,] "ADI"          "Information Technology"
#>  [4,] "ABBV"         "Health Care"
#>  [5,] "AET"          "Health Care"
#>  [6,] "A"            "Health Care"
#>  [7,] "APD"          "Materials"
#>  [8,] "AA"           "Materials"
#>  [9,] "CF"           "Materials"
```

Interestingly, we can observe that the automatic clustering performed on $\boldsymbol{\Psi}$ correctly identifies the sectors of the stocks.

Assessing investment funds

In this example, we will assess the performance of several investment funds based on factor models. We will use the S&P 500 as the explicit market factor and assume a zero risk-free return $r_f = 0$. In particular, we will consider the following six Exchange-Traded Funds (ETFs):

- SPY - SPDR S&P 500 ETF (index tracking)
- XIVH - Velocity VIX Short Vol Hedged (high alpha)
- SPHB - PowerShares S&P 500 High Beta Portfolio (high beta)
- SPLV - PowerShares S&P 500 Low Volatility Portfolio (low beta)
- USMV - iShares Edge MSCI Min Vol USA ETF
- JKD - iShares Morningstar Large-Cap ETF

We start by loading the data:

```r
library(xts)
library(quantmod)

# set begin-end date and stock namelist
begin_date <- "2016-10-01"
end_date <- "2017-06-30"
stock_namelist <- c("SPY","XIVH", "SPHB", "SPLV", "USMV", "JKD")

# download data from YahooFinance
data_set <- xts()
for (stock_index in 1:length(stock_namelist))
  data_set <- cbind(data_set, Ad(getSymbols(stock_namelist[stock_index],
                                            from = begin_date, to = end_date, auto.assign = FALSE)))
colnames(data_set) <- stock_namelist
indexClass(data_set) <- "Date"
head(data_set)
#>                 SPY   XIVH     SPHB     SPLV     USMV      JKD
#> 2016-10-03 207.7082 29.400 31.84952 39.42076 43.74514 123.0232
#> 2016-10-04 206.6494 30.160 31.76232 38.96071 43.31846 122.5425
#> 2016-10-05 207.5638 30.160 32.37276 38.87445 43.22148 123.0905
#> 2016-10-06 207.7082 30.160 32.30493 38.94155 43.25057 123.2347
#> 2016-10-07 206.9959 30.670 32.05300 38.83612 43.20209 122.9751
#> 2016-10-10 208.0740 31.394 32.35338 39.03739 43.41543 123.7635

SP500_index <- Ad(getSymbols("^GSPC", from = begin_date, to = end_date, auto.assign = FALSE))
colnames(SP500_index) <- "index"
head(SP500_index)
#>              index
#> 2016-10-03 2161.20
#> 2016-10-04 2150.49
#> 2016-10-05 2159.73
#> 2016-10-06 2160.77
#> 2016-10-07 2153.74
#> 2016-10-10 2163.66

# compute the log-returns of the stocks and of the SP500 index as explicit factor
X <- diff(log(data_set), na.pad = FALSE)
N <- ncol(X)  # number of stocks
T <- nrow(X)  # number of days
f <- diff(log(SP500_index), na.pad = FALSE)
```

Now we can compute the alpha and beta of all the ETFs:

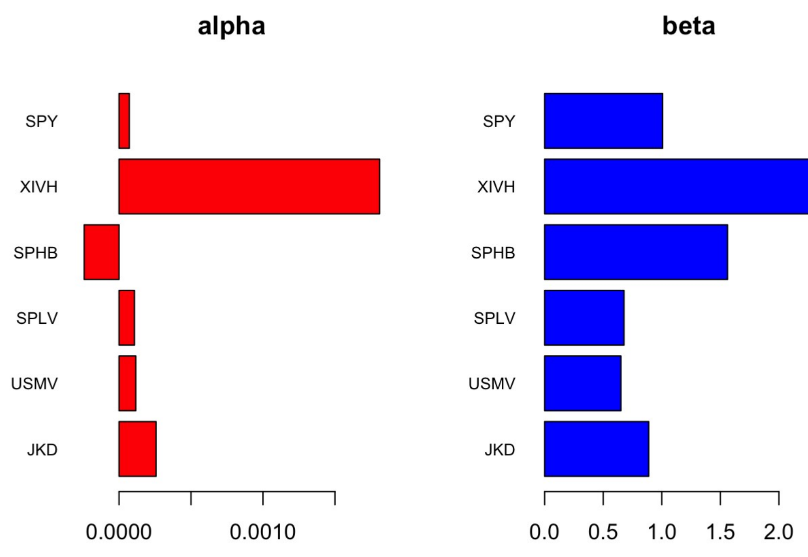
```r
F_ <- cbind(ones = 1, f)
Gamma <- t(solve(t(F_) %*% F_, t(F_) %*% X))
colnames(Gamma) <- c("alpha", "beta")
alpha <- Gamma[, 1]
beta <- Gamma[, 2]
print(Gamma)
#>              alpha      beta
#> SPY   7.142114e-05 1.0071408
#> XIVH  1.810392e-03 2.4971086
#> SPHB -2.422773e-04 1.5613574
#> SPLV  1.071122e-04 0.6777118
#> USMV  1.166182e-04 0.6511647
#> JKD   2.569642e-04 0.8883836
```

Some observations are now in order:

- SPY is an ETF that follows the S&P 500 and, as expected, it has an alpha almost zero and a beta almost one: $\alpha = 7.142211 \times 10^{-5}$ and $\beta = 1.0071423$.
- XIVH is an ETF with a high alpha and indeed the alpha computed is the highest among the ETFs (1-2 orders of magnitude higher): $\alpha = 1.810392 \times 10^{-3}$.
- SPHB is an ETF supposedly with a high beta and the computed beta is amongh the hightest but not the highest: $\beta = 1.5613531$. Interestingly, the computed alpha is negative! So this ETF seems to be dangerous and caution should be taken.
- SPLV is an ETF that aims at having a low volatility and indeed the computed beta is on the low side: $\beta = 0.6777072$.
- USMV is also an ETF that aims at having a low volatility and indeed the computed beta is the lowest: $\beta = 0.6511671$.
- JKD seems not to be extreme and shows a good tradeof.

To get a feeling we can use some visualization:

```
par(mfrow = c(1,2))  # two plots side by side
barplot(rev(alpha), horiz = TRUE, main = "alpha", col = "red", cex.names = 0.75, las = 1)
barplot(rev(beta), horiz = TRUE, main = "beta", col = "blue", cex.names = 0.75, las = 1)
```



```
par(mfrow = c(1,1))  # reset to normal single plot
```

We can also compare the different ETFs in a more systematic way using, for example, the Sharpe ratio. Recalling the factor model for one asset and one factor

$$x_t = \alpha + \beta f_t + \epsilon_t, \quad t = 1, \ldots, T,$$

we obtain

$$\begin{aligned} E[x_t] &= \alpha + \beta E[f_t] \\ \text{var}[x_t] &= \beta^2 \text{var}[f_t] + \sigma^2 \end{aligned}$$

and the Sharpe ratio follows:

$$SR = \frac{E[x_t]}{\sqrt{\text{var}[x_t]}} = \frac{\alpha + \beta E[f_t]}{\sqrt{\beta^2 \text{var}[f_t] + \sigma^2}} = \frac{\alpha/\beta + E[f_t]}{\sqrt{\text{var}[f_t] + \sigma^2/\beta^2}} \approx \frac{\alpha/\beta + E[f_t]}{\sqrt{\text{var}[f_t]}}$$

where we have assumed $\sigma^2 \ll \text{var}[f_t]$. Thus, a way to rank the different assets based on the Sharpe ratio is to rank them based on the ratio α/β:

```
idx_sorted <- sort(alpha/beta, decreasing = TRUE, index.return = TRUE)$ix
SR <- colMeans(X)/sqrt(diag(var(X)))
ranking <- cbind("alpha/beta" = (alpha/beta)[idx_sorted],
                 SR = SR[idx_sorted],
                 alpha = alpha[idx_sorted],
                 beta = beta[idx_sorted])
print(ranking)
#>         alpha/beta         SR         alpha      beta
#> XIVH  7.249952e-04 0.13919483  1.810392e-03 2.4971086
#> JKD   2.892492e-04 0.17682813  2.569642e-04 0.8883836
#> USMV  1.790917e-04 0.12280037  1.166182e-04 0.6511647
#> SPLV  1.580498e-04 0.10888344  1.071122e-04 0.6777118
#> SPY   7.091476e-05 0.14170567  7.142114e-05 1.0071408
#> SPHB -1.551709e-04 0.07400875 -2.422773e-04 1.5613574
```

The following observations are in order:

- In terms of α/β, XIVH is the best (with the largest alpha) and SPHB the worst (with a negative alpha!).
- In terms of exact Sharpe ratio (more exactly, Information ratio since we are ignoring the risk-free rate), JDK is the best, followed by SPY, which is surprising since it is just the market. This confirms the wisdom that the majority of the investment funds do not outperform the market.
- SPHB is clearly the worst according to any measure: negative alpha, negative alpha-beta ratio, and exact Sharpe ratio.

- JDK manages to have the best performance because its alpha is good (although not the best) while at the same time has a moderate beta of 0.88, so its exposure to the market if not extreme.
- XIVH and SPHB have extreme exposures to the market with large betas.
- USMV has the smallest exposure to the market with an acceptable alpha, and it's Sharpe ratio is close to the top second and third.

Fundamental factor models: Fama-French

This example will illustrate the popular Fama-French 3-factor model using 10 stocks from the S&P 500.

Let's start by loading the data:

```r
library(xts)
library(quantmod)

# set begin-end date and stock namelist
begin_date <- "2013-01-01"
end_date <- "2017-08-31"
stock_namelist <- c("AAPL", "AMD", "ADI",  "ABBV", "AET", "A",  "APD", "AA","CF")

# download data from YahooFinance
data_set <- xts()
for (stock_index in 1:length(stock_namelist))
  data_set <- cbind(data_set, Ad(getSymbols(stock_namelist[stock_index],
                                            from = begin_date, to = end_date, auto.assign = FALSE)))
colnames(data_set) <- stock_namelist
indexClass(data_set) <- "Date"

# download Fama-French factors from website
#url <- "http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/F-F_Research_Data_Factors_daily_CSV.zip"
#temp <- tempfile()
#download.file(url, temp, method = "libcurl", mode = "wb")
#unzip(temp, "F-F_Research_Data_Factors_daily.CSV")
#unlink(temp)
mydata <- read.csv("F-F_Research_Data_Factors_daily.CSV", skip = 4)
mydata <- mydata[-nrow(mydata), ]  # remove last row
fama_lib <- xts(x = mydata[, c(2,3,4)], order.by = as.Date(paste(mydata[, 1]), "%Y%m%d"))
str(fama_lib)
#> An 'xts' object on 1926-07-01/2017-11-30 containing:
#>   Data: num [1:24120, 1:3] 0.1 0.45 0.17 0.09 0.21 -0.71 0.62 0.04 0.48 0.04 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : NULL
#>   ..$ : chr [1:3] "Mkt.RF" "SMB" "HML"
#>   Indexed by objects of class: [Date] TZ: UTC
#>   xts Attributes:
#>  NULL
head(fama_lib)
#>            Mkt.RF   SMB   HML
#> 1926-07-01   0.10 -0.24 -0.28
#> 1926-07-02   0.45 -0.32 -0.08
#> 1926-07-06   0.17  0.27 -0.35
#> 1926-07-07   0.09 -0.59  0.03
#> 1926-07-08   0.21 -0.36  0.15
#> 1926-07-09  -0.71  0.44  0.56
tail(fama_lib)
#>            Mkt.RF   SMB   HML
#> 2017-11-22  -0.05  0.10 -0.04
#> 2017-11-24   0.21  0.02 -0.44
#> 2017-11-27  -0.06 -0.36  0.03
#> 2017-11-28   1.06  0.38  0.84
#> 2017-11-29   0.02  0.04  1.45
#> 2017-11-30   0.82 -0.56 -0.50

# compute the log-returns of the stocks and the Fama-French factors
X <- diff(log(data_set), na.pad = FALSE)
N <- ncol(X)  # number of stocks
T <- nrow(X)  # number of days
F <- fama_lib[index(X)]/100
```

Now we have the three explicit factors in matrix $\mathbf{F}$ and want to fit the model

$$\mathbf{X}^T = \boldsymbol{\alpha}\mathbf{1}^T + \mathbf{B}\mathbf{F}^T + \mathbf{E}^T$$

where now the loadings are a matrix of betas: $\mathbf{B} = [\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_N]^T = [\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3]$. As before, we can do a LS fitting as

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{minimize}} \quad \|\mathbf{X}^T - \boldsymbol{\alpha}\mathbf{1}^T - \mathbf{B}\mathbf{F}^T\|_F^2.$$

More conveniently, we define $\boldsymbol{\Gamma} = [\boldsymbol{\alpha}, \mathbf{B}]$ and the extended factors $\tilde{\mathbf{F}} = [\mathbf{1}, \mathbf{F}]$. The LS formulation can then be written as

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\text{minimize}} \quad \|\mathbf{X}^T - \boldsymbol{\Gamma}\tilde{\mathbf{F}}^T\|_F^2$$

with solution

$$\boldsymbol{\Gamma} = \mathbf{X}^T\tilde{\mathbf{F}}(\tilde{\mathbf{F}}^T\tilde{\mathbf{F}})^{-1}$$

```
F_ <- cbind(ones = 1, F)
Gamma <- t(solve(t(F_) %*% F_, t(F_) %*% X))
colnames(Gamma) <- c("alpha", "b1", "b2", "b3")
alpha <- Gamma[, 1]
B <- Gamma[, 2:4]
print(Gamma)
#>              alpha       b1          b2          b3
#> AAPL  3.279677e-04 0.974649 -0.26292305 -0.49365542
#> AMD   6.181760e-04 1.406210  0.80738336 -0.07240117
#> ADI  -2.285023e-05 1.212401  0.09025960 -0.20739124
#> ABBV  1.621379e-04 1.058234  0.02833773 -0.72152716
#> AET   5.070125e-04 1.021904 -0.23373650 -0.16659025
#> A     1.146137e-05 1.218142  0.10370977 -0.20487496
#> APD   6.281495e-05 1.022294 -0.04394067  0.11060838
#> AA   -4.607740e-05 1.338888  0.62560013  0.99088170
#> CF   -5.671564e-04 1.037482  0.49458776  0.82157606
```

Alternatively, we can simply use the R package **covFactorModel** to do the work for us:

```
library(covFactorModel)
factor_model <- factorModel(X, type = "M", econ_fact = F)
cbind(alpha = factor_model$alpha, beta = factor_model$beta)
#>              alpha   Mkt.RF         SMB         HML
#> AAPL  3.279677e-04 0.974649 -0.26292305 -0.49365542
#> AMD   6.181760e-04 1.406210  0.80738336 -0.07240117
#> ADI  -2.285023e-05 1.212401  0.09025960 -0.20739124
#> ABBV  1.621379e-04 1.058234  0.02833773 -0.72152716
#> AET   5.070125e-04 1.021904 -0.23373650 -0.16659025
#> A     1.146137e-05 1.218142  0.10370977 -0.20487496
#> APD   6.281495e-05 1.022294 -0.04394067  0.11060838
#> AA   -4.607740e-05 1.338888  0.62560013  0.99088170
#> CF   -5.671564e-04 1.037482  0.49458776  0.82157606
```

Statistical factor models

Let's now consider statistical factor models or implicit factor models where both the factors and the loadings are not available. Recall the principal factor method for the model $\mathbf{X}^T = \boldsymbol{\alpha}\mathbf{1}^T + \mathbf{B}\mathbf{F}^T + \mathbf{E}^T$ with K factors:

1. PCA:
   - sample mean: $\hat{\boldsymbol{\alpha}} = \bar{\mathbf{x}} = \frac{1}{T}\mathbf{X}^T\mathbf{1}_T$
   - demeaned matrix: $\bar{\mathbf{X}} = \mathbf{X} - \mathbf{1}_T\bar{\mathbf{x}}^T$
   - sample covariance matrix: $\hat{\boldsymbol{\Sigma}} = \frac{1}{T-1}\bar{\mathbf{X}}^T\bar{\mathbf{X}}$
   - eigen-decomposition: $\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\Gamma}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{\Gamma}}^T$
2. Estimates:
   - $\hat{\mathbf{B}} = \hat{\boldsymbol{\Gamma}}_1\hat{\boldsymbol{\Lambda}}_1^{1/2}$
   - $\hat{\boldsymbol{\Psi}} = \text{Diag}\left(\hat{\boldsymbol{\Sigma}} - \hat{\mathbf{B}}\hat{\mathbf{B}}^T\right)$
   - $\hat{\boldsymbol{\Sigma}} = \hat{\mathbf{B}}\hat{\mathbf{B}}^T + \hat{\boldsymbol{\Psi}}$
3. Update the eigen-decomposition: $\hat{\boldsymbol{\Sigma}} - \hat{\boldsymbol{\Psi}} = \hat{\boldsymbol{\Gamma}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{\Gamma}}^T$
4. Repeat Steps 2-3 until convergence.

```
K <- 3
alpha <- colMeans(X)
X_ <- X - matrix(alpha, T, N, byrow = TRUE)
Sigma_prev <- matrix(0, N, N)
Sigma <- (1/(T-1)) * t(X_) %*% X_
eigSigma <- eigen(Sigma)
while (norm(Sigma - Sigma_prev, "F")/norm(Sigma, "F") > 1e-3) {
  B <- eigSigma$vectors[, 1:K, drop = FALSE] %*% diag(sqrt(eigSigma$values[1:K]), K, K)
  Psi <- diag(diag(Sigma - B %*% t(B)))
  Sigma_prev <- Sigma
  Sigma <- B %*% t(B) + Psi
  eigSigma <- eigen(Sigma - Psi)
}
cbind(alpha, B)
#>              alpha
#> AAPL  8.972880e-04 -0.004809983  0.004422227 -0.004824742
#> AMD   1.372247e-03 -0.035072415 -0.012705646  0.001747896
#> ADI   6.533116e-04 -0.007559399  0.005131605 -0.004421491
#> ABBV  7.787929e-04 -0.004565552  0.006071597 -0.006507520
#> AET   1.086452e-03 -0.004759671  0.004902210 -0.004009799
#> A     6.902479e-04 -0.006081979  0.006195930 -0.004238986
#> APD   6.236567e-04 -0.004501732  0.005773292 -0.001999060
#> AA    6.277163e-04 -0.010500828  0.014463970 -0.004052819
#> CF   -4.783128e-05 -0.008070237  0.014738714  0.015050633
```

Again, we can simply use the R package **covFactorModel** to do the work for us:

```
library(covFactorModel)
factor_model <- factorModel(X, type = "S", K = K, max_iter = 10)
cbind(alpha = factor_model$alpha, beta = factor_model$beta)
#>              alpha      factor1      factor2      factor3
#> AAPL  8.972880e-04 -0.004809983  0.004422227 -0.004824742
#> AMD   1.372247e-03 -0.035072415 -0.012705646  0.001747896
#> ADI   6.533116e-04 -0.007559399  0.005131605 -0.004421491
#> ABBV  7.787929e-04 -0.004565552  0.006071597 -0.006507520
#> AET   1.086452e-03 -0.004759671  0.004902210 -0.004009799
#> A     6.902479e-04 -0.006081979  0.006195930 -0.004238986
#> APD   6.236567e-04 -0.004501732  0.005773292 -0.001999060
#> AA    6.277163e-04 -0.010500828  0.014463970 -0.004052819
#> CF   -4.783128e-05 -0.008070237  0.014738714  0.015050633
```

Final comparison of covariance matrix estimations via different factor models

We will finally compare the following different factor models:

- sample covariance matrix
- macroeconomic 1-factor model
- fundamental 3-factor Fama-French model
- statistical factor model

We will estimate the models during a training phase and then we will compare how well the estimated covariance matrices do compared to the sample covariance matrix of the test phase. The estimation error will be evaluated in terms of the Frobenius norm $\|\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}_{\text{true}}\|_F$ as well as the PRIAL (PeRcentage Improvement in Average Loss):

$$\text{PRIAL} = 100 \times \frac{\|\boldsymbol{\Sigma}_{\text{scm}} - \boldsymbol{\Sigma}_{\text{true}}\|_F^2 - \|\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Sigma}_{\text{true}}\|_F^2}{\|\boldsymbol{\Sigma}_{\text{scm}} - \boldsymbol{\Sigma}_{\text{true}}\|_F^2}$$

which goes to 0 when the estimation $\hat{\boldsymbol{\Sigma}}$ tends to the sample covariance matrix $\boldsymbol{\Sigma}_{\text{scm}}$ and goes to 100 when the estimation $\hat{\boldsymbol{\Sigma}}$ tends to the true covariance matrix $\boldsymbol{\Sigma}_{\text{true}}$.

Let's load the training and test sets:

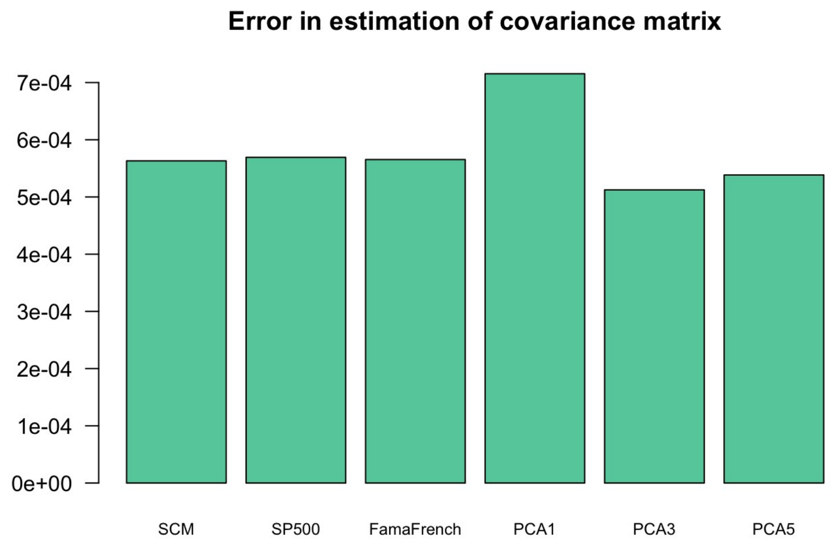
```r
library(xts)
library(quantmod)

# set begin-end date and stock namelist
begin_date <- "2013-01-01"
end_date <- "2015-12-31"
stock_namelist <- c("AAPL", "AMD", "ADI",  "ABBV", "AET", "A",  "APD", "AA","CF")

# prepare stock data
data_set <- xts()
for (stock_index in 1:length(stock_namelist))
  data_set <- cbind(data_set, Ad(getSymbols(stock_namelist[stock_index],
                                            from = begin_date, to = end_date, auto.assign = FALSE)))
colnames(data_set) <- stock_namelist
indexClass(data_set) <- "Date"
X <- diff(log(data_set), na.pad = FALSE)
N <- ncol(X)
T <- nrow(X)

# prepare Fama-French factors
mydata <- read.csv("F-F_Research_Data_Factors_daily.CSV", skip = 4)
mydata <- mydata[-nrow(mydata), ]  # remove last row
fama_lib <- xts(x = mydata[, c(2,3,4)], order.by = as.Date(paste(mydata[, 1]), "%Y%m%d"))
F_FamaFrench <- fama_lib[index(X)]/100

# prepare index
SP500_index <- Ad(getSymbols("^GSPC", from = begin_date, to = end_date, auto.assign = FALSE))
colnames(SP500_index) <- "index"
f_SP500 <- diff(log(SP500_index), na.pad = FALSE)

# split data into training and set data
T_trn <- round(0.45*T)
X_trn <- X[1:T_trn, ]
X_tst <- X[(T_trn+1):T, ]
F_FamaFrench_trn <- F_FamaFrench[1:T_trn, ]
F_FamaFrench_tst <- F_FamaFrench[(T_trn+1):T, ]
f_SP500_trn <- f_SP500[1:T_trn, ]
f_SP500_tst <- f_SP500[(T_trn+1):T, ]
```

Now let’s estimate the different factor models with the training data:

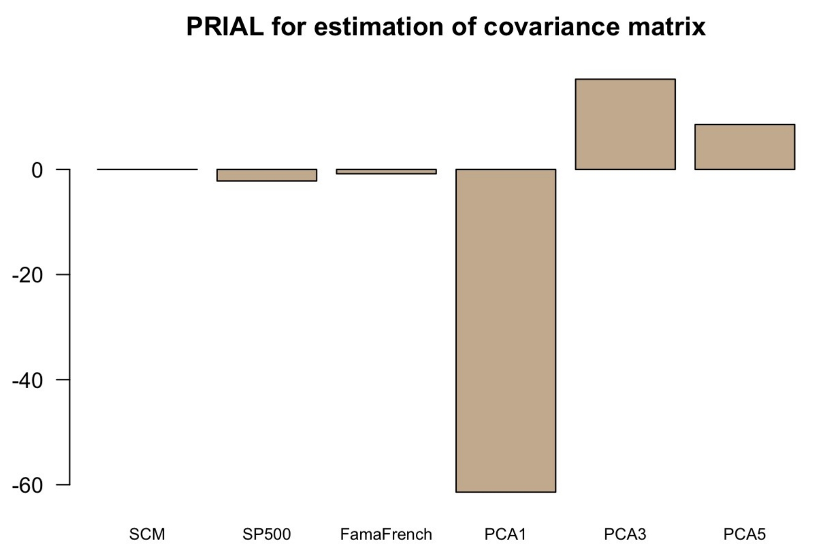
```r
# sample covariance matrix
Sigma_SCM <- cov(X_trn)

# 1-factor model
F_ <- cbind(ones = 1, f_SP500_trn)
Gamma <- t(solve(t(F_) %*% F_, t(F_) %*% X_trn))
colnames(Gamma) <- c("alpha", "beta")
alpha <- Gamma[, 1]
beta <- Gamma[, 2]
E <- xts(t(t(X_trn) - Gamma %*% t(F_)), index(X_trn))
Psi <- (1/(T_trn-2)) * t(E) %*% E
Sigma_SP500 <- as.numeric(var(f_SP500_trn)) * beta %o% beta + diag(diag(Psi))

# Fama-French 3-factor model
F_ <- cbind(ones = 1, F_FamaFrench_trn)
Gamma <- t(solve(t(F_) %*% F_, t(F_) %*% X_trn))
colnames(Gamma) <- c("alpha", "beta1", "beta2", "beta3")
alpha <- Gamma[, 1]
B <- Gamma[, 2:4]
E <- xts(t(t(X_trn) - Gamma %*% t(F_)), index(X_trn))
Psi <- (1/(T_trn-4)) * t(E) %*% E
Sigma_FamaFrench <- B %*% cov(F_FamaFrench_trn) %*% t(B) + diag(diag(Psi))

# Statistical 1-factor model
K <- 1
alpha <- colMeans(X_trn)
X_trn_ <- X_trn - matrix(alpha, T_trn, N, byrow = TRUE)
Sigma_prev <- matrix(0, N, N)
Sigma <- (1/(T_trn-1)) * t(X_trn_) %*% X_trn_
eigSigma <- eigen(Sigma)
while (norm(Sigma - Sigma_prev, "F")/norm(Sigma, "F") > 1e-3) {
  B <- eigSigma$vectors[, 1:K, drop = FALSE] %*% diag(sqrt(eigSigma$values[1:K]), K, K)
  Psi <- diag(diag(Sigma - B %*% t(B)))
  Sigma_prev <- Sigma
  Sigma <- B %*% t(B) + Psi
  eigSigma <- eigen(Sigma - Psi)
}
Sigma_PCA1 <- Sigma

# Statistical 3-factor model
K <- 3
alpha <- colMeans(X_trn)
X_trn_ <- X_trn - matrix(alpha, T_trn, N, byrow = TRUE)
Sigma_prev <- matrix(0, N, N)
Sigma <- (1/(T_trn-1)) * t(X_trn_) %*% X_trn_
eigSigma <- eigen(Sigma)
while (norm(Sigma - Sigma_prev, "F")/norm(Sigma, "F") > 1e-3) {
  B <- eigSigma$vectors[, 1:K] %*% diag(sqrt(eigSigma$values[1:K]), K, K)
  Psi <- diag(diag(Sigma - B %*% t(B)))
  Sigma_prev <- Sigma
  Sigma <- B %*% t(B) + Psi
  eigSigma <- eigen(Sigma - Psi)
}
Sigma_PCA3 <- Sigma

# Statistical 5-factor model
K <- 5
alpha <- colMeans(X_trn)
X_trn_ <- X_trn - matrix(alpha, T_trn, N, byrow = TRUE)
Sigma_prev <- matrix(0, N, N)
Sigma <- (1/(T_trn-1)) * t(X_trn_) %*% X_trn_
eigSigma <- eigen(Sigma)
while (norm(Sigma - Sigma_prev, "F")/norm(Sigma, "F") > 1e-3) {
  B <- eigSigma$vectors[, 1:K] %*% diag(sqrt(eigSigma$values[1:K]), K, K)
  Psi <- diag(diag(Sigma - B %*% t(B)))
  Sigma_prev <- Sigma
  Sigma <- B %*% t(B) + Psi
  eigSigma <- eigen(Sigma - Psi)
}
Sigma_PCA5 <- Sigma
```

Finally, let's compare the different estimations in the test data:

```
Sigma_true <- cov(X_tst)
error <- c(SCM = norm(Sigma_SCM - Sigma_true, "F"),
           SP500 = norm(Sigma_SP500 - Sigma_true, "F"),
           FamaFrench = norm(Sigma_FamaFrench - Sigma_true, "F"),
           PCA1 = norm(Sigma_PCA1 - Sigma_true, "F"),
           PCA3 = norm(Sigma_PCA3 - Sigma_true, "F"),
           PCA5 = norm(Sigma_PCA5 - Sigma_true, "F"))
print(error)
#>          SCM        SP500   FamaFrench         PCA1         PCA3
#> 0.0005631143 0.0005692461 0.0005654330 0.0007154259 0.0005124815
#>         PCA5
#> 0.0005385124
barplot(error, main = "Error in estimation of covariance matrix",
        col = "aquamarine3", cex.names = 0.75, las = 1)
```



```
ref <- norm(Sigma_SCM - Sigma_true, "F")^2
PRIAL <- 100*(ref - error^2)/ref
print(PRIAL)
#>         SCM       SP500  FamaFrench        PCA1        PCA3        PCA5
#>   0.0000000  -2.1896811  -0.8252093 -61.4121436  17.1746678   8.5469276
barplot(PRIAL, main = "PRIAL for estimation of covariance matrix",
        col = "bisque3", cex.names = 0.75, las = 1)
```



👉 The final conclusion is that using factor models for covariance matrix estimation may help. In addition, it is not clear that using explicit factors helps as compared to the statistical factor modeling.