Covariance Estimation

Rohit Arora

2015-07-12

Abstract

There exists a rich modern set of covariance matrix estimator methods for use in financial data. The purpose of `covmat` package is to implement some of these techniques such that they are readily available to be used with appropriate financial data. The purpose of this vignette is to demonstrate the usage of functions implemented in the `covmat` package.

Contents

1 Load Package

The latest version of the `covmat` package can be downloaded and installed through the following command:

```
library(devtools)
install_github("arorar/covmat")
```

2 Stambaugh Estimator

Longer monthly return data series are often available from well-established companies. However, if we turn to newer companies we run into the problem of unequal histories where newer companies have shorter return histories. To calculate a covariance matrix for portfolio optimization with assets having unequal histories we can naively truncate the data to the largest available cross-section. This means discarding data. However, we can do better with using all available data for all assets using a methodology proposed by (R. F. Stambaugh 1997).

2.1 Data

Say that we have a portfolio of 4 tech stocks `TWTR, LNKD, V, YHOO, GE` of which only 2 have a return history of 6 years, while the other 3 have been around for less than four years.

Lets start by visualizing the data. We can use the `plotmissing` function to do this. The second parameter of this function allows us to choose how we want to visualize the data. A value of 3 suggests a time series plot and 4 suggests a matrix plot.

```
plotmissing(data, which=c(3,4))
```

We will choose to visualize the timeseries in this case

Notice how some return series have missing values and shorter lengths compared to other series. In particular Twitter recently had its IPO and has a large number of missing values. LinkedIn had its IPO in 2011 and has lesser missing values. While GE and Yahoo have complete data histories and no missing values for the period under consideration.

2.2 Covariance estimation

To construct a valid covariance matrix we could truncate the data series making all of them about a year long and then calculate the sample covariance matrix. However, we can do better by using Stambaugh's method. Starting from the truncated sample covariance matrix, this technique produces improvements to the covariance matrix that utilizes all the available data along with using cross sectional dependency in returns data to construct a more accurate covariance matrix.

Firstly, we will use the `stambaugh.fit` function to construct the covariance matrices. This function takes in data and the type of covariance matrix that needs to be estimated. Additional arguments can be passed for robust estimation.

```
stambaugh.fit(R, method=c("classic","robust", "truncated"), ...)
```

Let us compare a classical covariance matrix computed using Stambaugh's technique with a truncated classical covariance estimator.

```
models1 <- stambaugh.fit(symdata, method = c("classic", "truncated"))
```

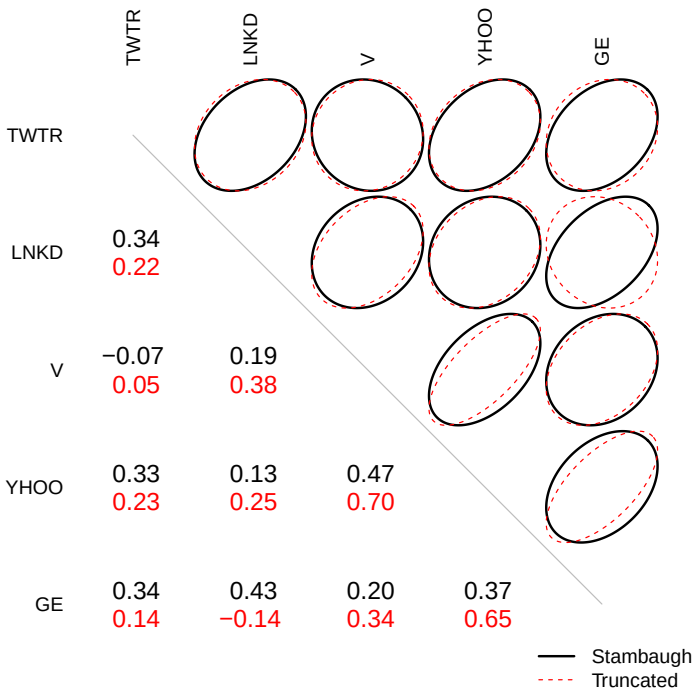
2.3 Plots

We can construct two types of plots, an ellipses plot and a distance plot. Each can be separately invoked using the same `plot` function but a separate `which` parameter.

```
plot(data, which=c(1,2))
```

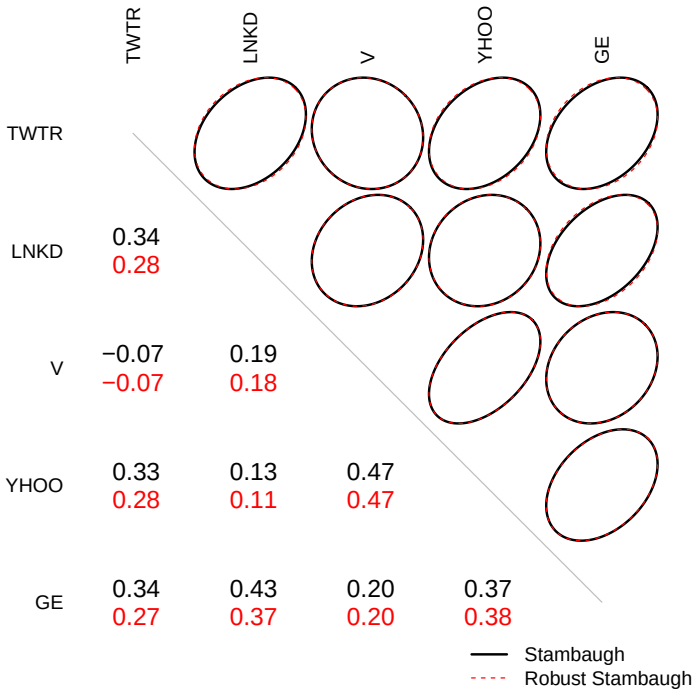
We can visually compare the covariances by examining their correlations using the ellipses plot. The ellipses are contours of standard bivariate normals overlayed for each model. Notice that the ellipses for truncated data can be significantly different from ellipses for covariance estimates computed using Stambaugh's technique. The difference is very prominent for certain pairs such as LinkedIn and GE where the sign of the correlation has completely reversed.

```
plot(models1,1)
```



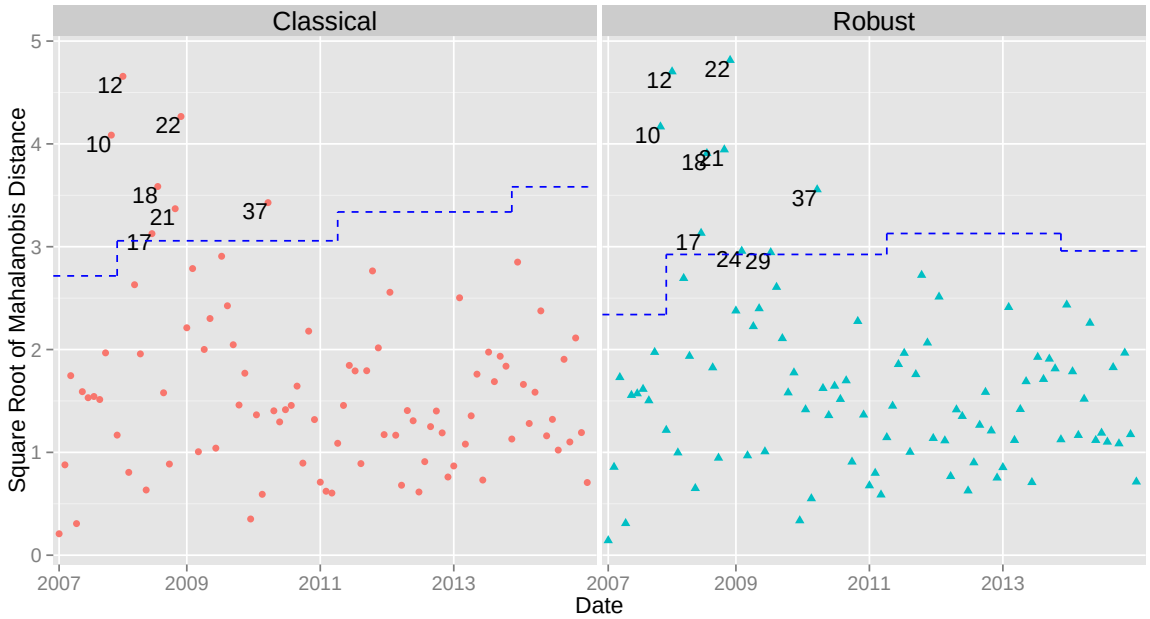
We will also compare the covariances of Stambaugh and Robust Stambaugh methods. Once again notice how the ellipses of Twitter and GE are significantly different.

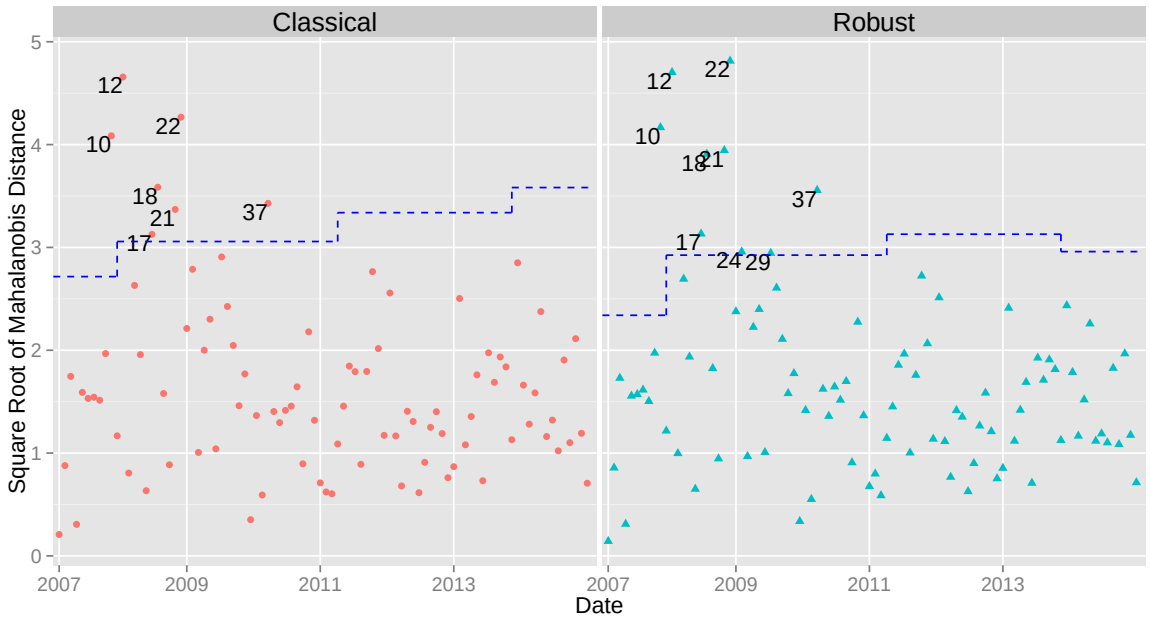
```
cov.control <- covRob.control(estim="mcd",alpha=0.9)
models <- stambaugh.fit(symdata, method = c("classic", "robust"),
                        cov.control=cov.control)
plot(models, 1)
```



We can also look at the distances of the individual stocks to examine the outliers for the same dataset. Notice that we will not use truncated models in this case for comparison as they have different data. Also we will pass a control parameter for robust covariance estimation. For outlier detection we need to evaluate the tail probability for a Chi-Square distribution. We will set it to 97.5%. This is indicated by the dashed line on the plot.

```
plot(models, 2, 0.975)
```





Notice how classical method suggests fewer outliers compared to Robust method. All outliers indicated by Classical method are also suggested as an outlier by the Robust method. Additional outliers indicated by the Robust method may be further examined.

3 FMMC estimator

A different approach to the problem of computing estimates for unequal return histories was proposed by (Jiang and Martin 2015). In this case we use longer factor histories of many risk factors and and choose a subset to construct a risk model for the asset returns. We can then simulate from such a risk model to construct longer return histories and use them to construct more accurate measure of covariance.

3.1 Data

For factors we will use the 5 factor model (Fama and French 2014). The five factor time series model tries to capture the market effect, size effect, value effect, profitability effect, and investment quality effect for a stock. In addition to the five factors we will also include the momentum factor as proposed in (Carhart 1997). All data for the factors is publicly available on Kenneth French's website.[1]

In addition to these we will also add a liquidity factor proposed in (Pastor and Stambaugh 2003). Data for which is freely available on Pastor's website.[2]

[1] http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/
[2] http://faculty.chicagobooth.edu/lubos.pastor/research/

We will also add a volatility factor by constructing the first order difference in VIX.

The factor data and symbol data is stored in the package and can be loaded as follows

```
data("returnsdata")
data("factordata")
```

Let us consider a portfolio of 3 stocks `LNKD, V, LAZ` all of which have different return histories. We need to align these series to the monthly factor data. The alignment can be done as follows

```
symbols <- c('LNKD', 'V', 'LAZ')
symdata <- symdata["2007-04-01/2014-12-31",symbols]

dates.sym.monthly <- format(index(symdata), "%Y%m")
dates.factors.monthly <- format(index(factor.data), "%Y%m")
index(symdata) <- index(factor.data)[which(dates.factors.monthly
                                           %in% dates.sym.monthly)]
```

3.2 Covariance estimation

To construct the covariance matrix we will use the `fmmc.cov` function. This function takes an xts object of return series, an xts object of factor series and an align parameter. We emphasize that the simulated returns from different asset returns although longer are still unequal in length. The align parameter can be used to truncate data from the beginning or end to align all longer histories to the same length.

```
fmmc.cov(R, factors, robust = FALSE, parallel = TRUE,
                     align=c("end", "begin"), ...)
```

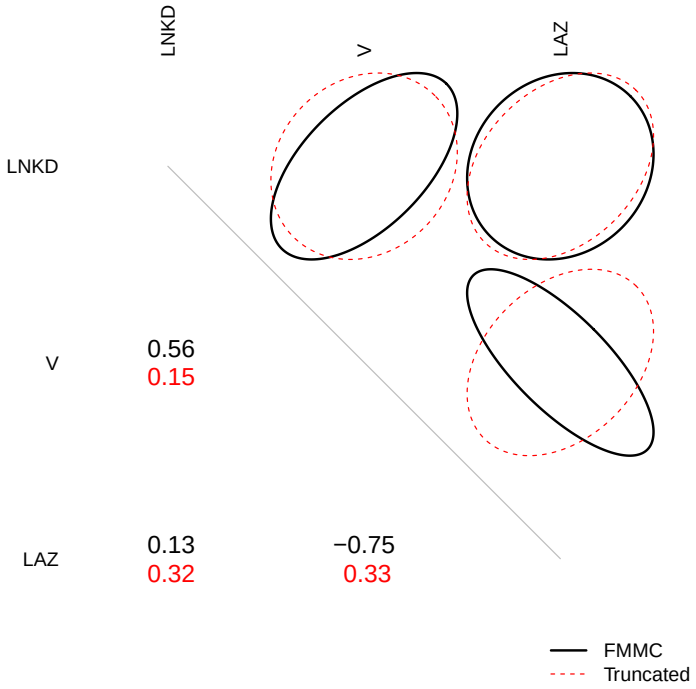
We can construct the covariance matrix as follows. Additional arguments can be passed to construct a robust covariance matrix. Notice that the parallel flag is turned on because FMMC is an embarrassingly parallel problem. Longer histories for each asset can be computed at the same time using all available cores of the CPU.

```
rets <- fmmc.cov(symdata, factor.data, parallel = FALSE)
```

3.3 Plots

We can also compare the performace of FMMC with the naive approach of constructing covariance estimates based on truncated data. Notice how FMMC estimates give a significantly different result from trucnated estimates for Lazard and Visa.

```
compare.cov(cov(rets),cov(na.omit(symdata)),c("FMMC","Truncated"))
```



4 Denoising using Random Matrix Theory

Random matrix theory provides a way to de-noise the sample covariance matrix . Let X be a matrix with T rows and N columns random matrix. C is the sample correlation matrix. Under the random matrix assumption, the eigenvalues of C must follow a Marchenko-Pastur density such that $N, T \to \infty, Q = N/T$. The density of eigenvalues is given by

$$f(\lambda) = \frac{Q}{2\pi\lambda\sigma^2}\sqrt{(\lambda_{max} - \lambda)(\lambda - \lambda_{\min})}$$

For a random matrix all eigenvaules will be within the range. The variance of these eigenvalues is 1. If any eigenvalue lies outside λ_{max} it is considered as a signal. We can choose these eigenvalues and replace the eigenvalues within the cutoff with either an average value or completely ignore them.

4.1 Data

To demonstrate the use of Random Matrix theory we will choose the `largesymdata` object which contains daily returns for ow Jones 30 index for a year.

```
data("largereturn")
```

4.2 Covariance estimation

To fit a covariance matrix we can use the `estRMT` fucntion.

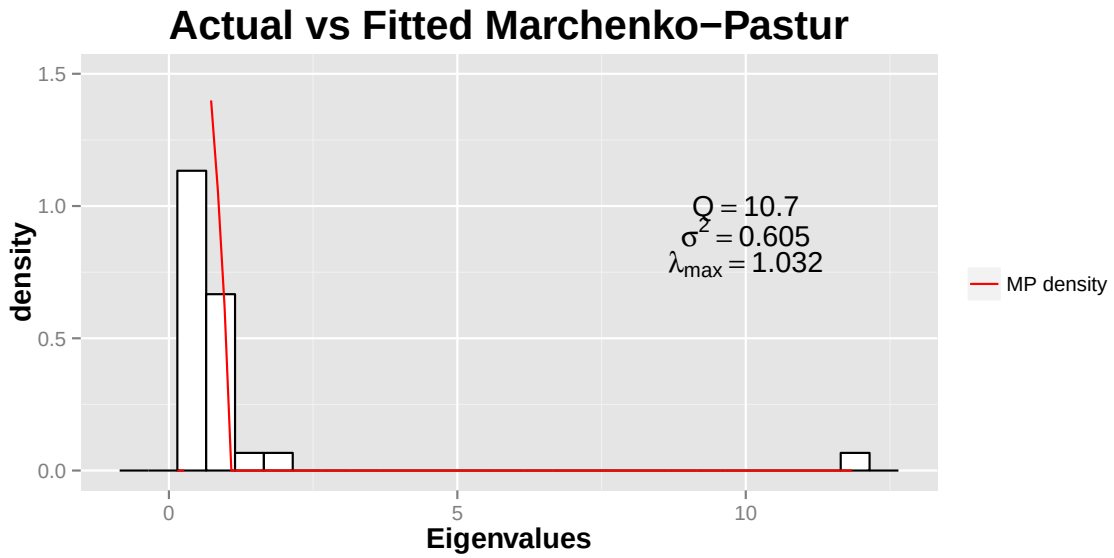
```
estRMT(R, Q =NA, cutoff = c("max", "each"),
       eigenTreat = c("average", "delete") ,
       numEig=1, parallel = TRUE)
```

This function takes serveral options, details of which can be found on the man page. However, in the simplest case we can pass a timeseries object of assets. In such a case we will assume that we know the largest eigenvalue and fit the distribution to the remaining eigenvalues. Values less than the cutoff are replaced with an average value.

4.3 Plots

Once we have fitted a model we can also investigate the fit visually using the `plot` function. The plot function takes in a fitted model and plots the fitted density overlayed on a histogram. It also displays some important fit parameters.

```
plot(model)
```



4.4 Evaluation

We will now demonstrate the use of RMT with a more elaborate example. Let us build a custom portfolio stratey using all 30 stocks from the Daily Dow Jones 30 index. We will used `largesymdata` object that contains daily data from 04/02/2014 to 07/10/2015. We will use the `PortfolioAnalytics` package for building the portfolio and backtesting the strategy.

Let us first construct a custom moment function where covariance is built by denoising using Random Matrix Theory. We assume no third/fourth order effects.

```
custom.portfolio.moments <- function(R, portfolio) {
  momentargs <- list()
  momentargs$mu  <-  matrix(as.vector(apply(R,2, "mean")), ncol = 1)
  momentargs$sigma  <-  estRMT(R)$cov
  momentargs$m3 <- matrix(0, nrow=ncol(R), ncol=ncol(R)^2)
  momentargs$m4 <- matrix(0, nrow=ncol(R), ncol=ncol(R)^3)

  return(momentargs)
}
```

We will construct a portfolio with the following specficatiion. No short sales are allowed. All cash needs to be invested at all times. As our objective, we will seek to maximize the quadratic utility which maximizes returns while controlling for risk.

```
pspec.lo <- portfolio.spec(assets = colnames(largesymdata))

#long-only
pspec.lo <- add.constraint(pspec.lo, type="full_investment")
pspec.lo <- add.constraint(pspec.lo, type="long_only")

pspec.lo <- add.objective(portfolio=pspec.lo, type="return", name="mean")
pspec.lo <- add.objective(portfolio=pspec.lo, type="risk", name="var")
```

Now lets backtest our strategy using an ordinary covariance matrix and a covariance matrix build by denoising using Random Matrix theory.

```
opt.ordinary <-
  optimize.portfolio.rebalancing(largesymdata, pspec.lo,
                                 optimize_method="quadprog",
                                 rebalance_on="months",
                                 training_period=120,
                                 trailing_periods=120)
opt.rmt <-
  optimize.portfolio.rebalancing(largesymdata, pspec.lo,
```

```
                                optimize_method="quadprog",
                                momentFUN = "custom.portfolio.moments",
                                rebalance_on="months",
                                training_period=120,
                                trailing_periods=120)
```

We can now extract weights and build cummulative returns using the `PerformanceAnalytics` package.

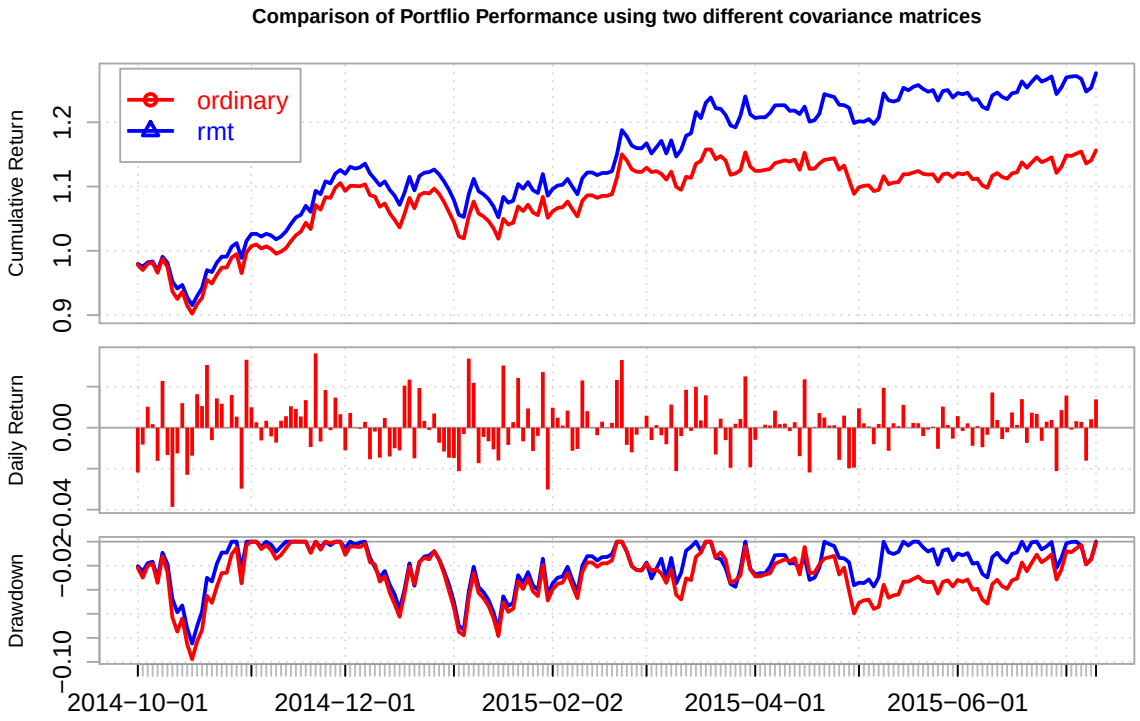
```
ordinary.wts <- na.omit(extractWeights(opt.ordinary))
ordinary <- Return.rebalancing(R=largesymdata, weights=ordinary.wts)

rmt.wts <- na.omit(extractWeights(opt.rmt))
rmt <- Return.rebalancing(R=largesymdata, weights=rmt.wts)

strat.rets <- merge.zoo(ordinary,rmt)
colnames(strat.rets) <- c("ordinary", "rmt")
```

In the chart below we can see that the cumulative returns generated using our strategy with filtering using Random Matrix Theory are superior to ordinary returns. They are also better with smaller drawdowns. This suggests that there is value in filtering a large sample covariance matrix before using it for optimizing portfolios.

```
charts.PerformanceSummary(strat.rets,wealth.index = T,
                          colorset = c("red","blue"),
                          main=paste(c("Comparison of Portflio ",
                                       "Performance using two ",
                                       "different covariance matrices"),
                                       collapse=""), cex.legend = 1.3,
                          cex.axis = 1.3, legend.loc = "topleft")
```



References

Carhart, Mark M. 1997. "On Persistence in Mutual Fund Performance." *Journal of Finance* 52 (1): 57–82.

Fama, Eugene F., and Kenneth R. French. 2014. "A Five-Factor Asset Pricing Model." *SSRN Electronic Journal*. doi:10.2139/ssrn.2287202.

Jiang, Yindeng, and Richard Doug Martin. 2015. "Better Risk and Performance Estimates with Factor Model Monte Carlo." *Journal of Risk*, May.

Pastor, Lubos, and Robert F Stambaugh. 2003. "Liquidity Risk and Expected Stock Returns." *Journal of Political Economy*, no. 111: 642–85. doi:10.1086/374184.

Stambaugh, Robert F. 1997. "Analyzing Investments Whose Histories Differ in Length." *Journal of Financial Economics* 45 (3). Elsevier BV: 285–331. doi:10.1016/s0304-405x(97)00020-2.