

Covariance Estimation

Rohit Arora

2015-08-21

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	3
2	Stambaugh Estimator	3
2.1	Data	3
2.2	Covariance estimation	4
2.3	Plots	5
3	FMMC estimator	7
3.1	Data	7
3.2	Covariance estimation	8
3.3	Plots	8
4	Denoising using Random Matrix Theory	9
4.1	Data	9
4.2	Covariance estimation	10
4.3	Plots	10
4.4	Evaluation	11
5	Independent Switching Dynamic Conditional Correlation Model	13
5.1	Data	13
5.2	Covariance estimation	14
5.3	Plots	14
5.4	Evaluation	15

6	Eigenvalue Shrinkage in Spiked Covariance Model	18
6.1	Data	19
6.2	Covariance estimation	19
6.3	Plots	20
7	Robust Exponential Smoothing of Multivariate Time Series	21
7.1	Covariance estimation	21
7.2	Evaluation	21
	References	24

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"))
```

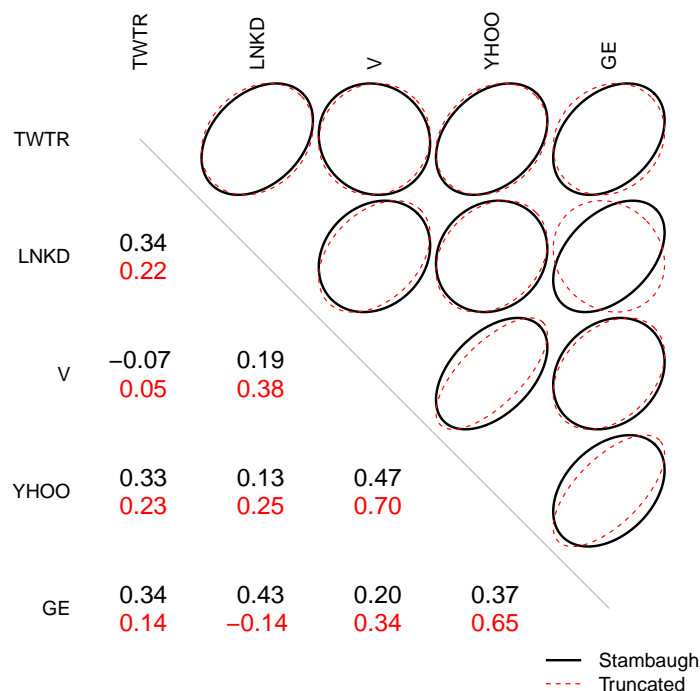
2.3 Plots

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))
```

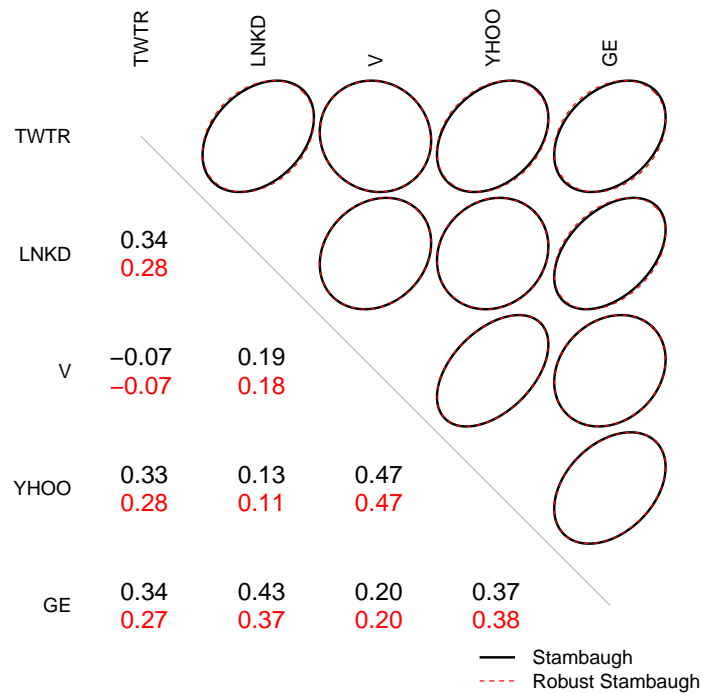
We can visually compare the covariances by examining their correlations using the ellipses plot. The ellipses are contours of standard bivariate normals overlayed for each model. Notice that the ellipses for truncated data can be significantly different from ellipses for covariance estimates computed 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)
```



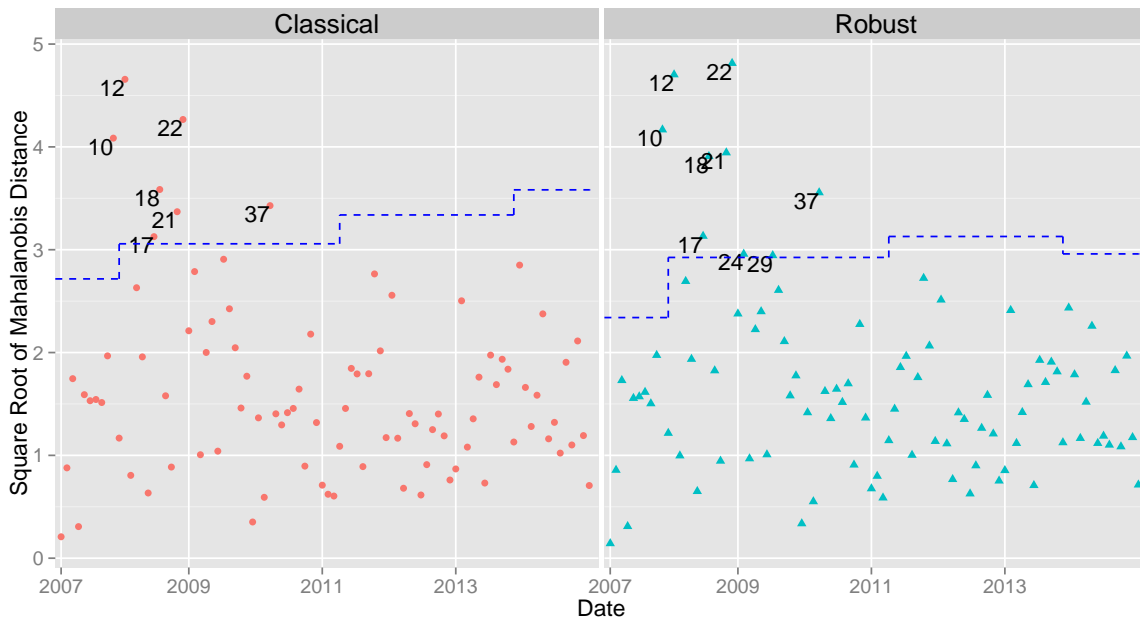
We will also compare the covariances of Stambaugh and Robust Stambaugh methods. Once again notice how the ellipses of Twitter and GE are significantly different.

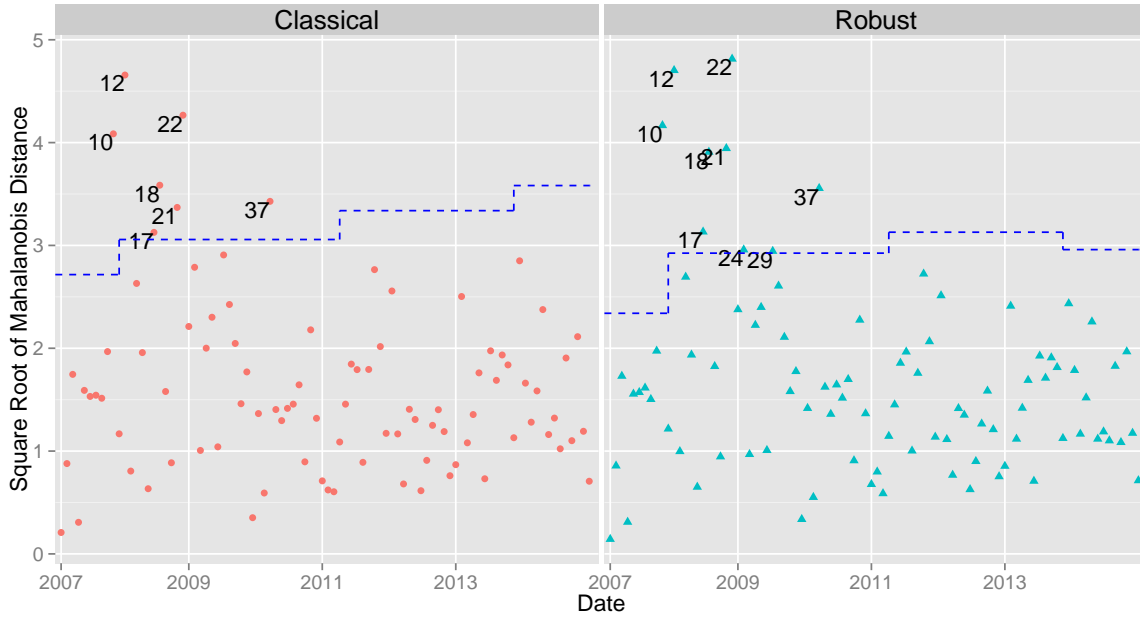
```
cov.control <- covRob.control(estim="mcd", alpha=0.9)
models <- stambaugh.fit(symdata, method = c("classic", "robust"),
                        cov.control=cov.control)
plot(models, 1)
```



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 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("missingdata")
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', 'YHOO')
symdata <- missingdata["2007-04-01/2014-12-31",symbols]
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"), ...)
```

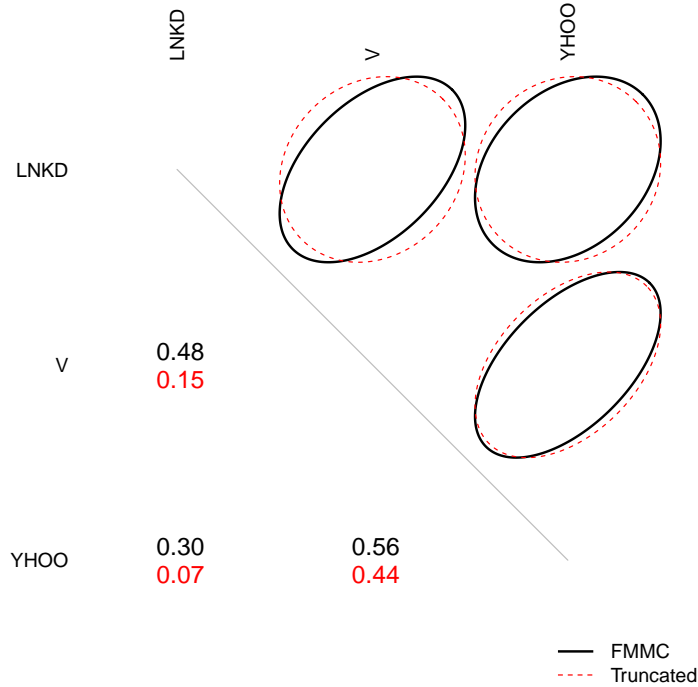
We can construct the covariance matrix as follows. Additional arguments can be passed to construct a robust covariance matrix. Notice that the parallel flag is turned on because FMMC is an embarrassingly parallel problem. Longer histories for each asset can be computed at the same time using all available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

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

3.3 Plots

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


```
compareCov(fmmc, 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 $N, T \rightarrow \infty, Q = N/T$. 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 eigenvalues 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 `dow30data` object which contains daily returns for Dow Jones 30 index for a year.

```
data("dow30data")
```

4.2 Covariance estimation

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

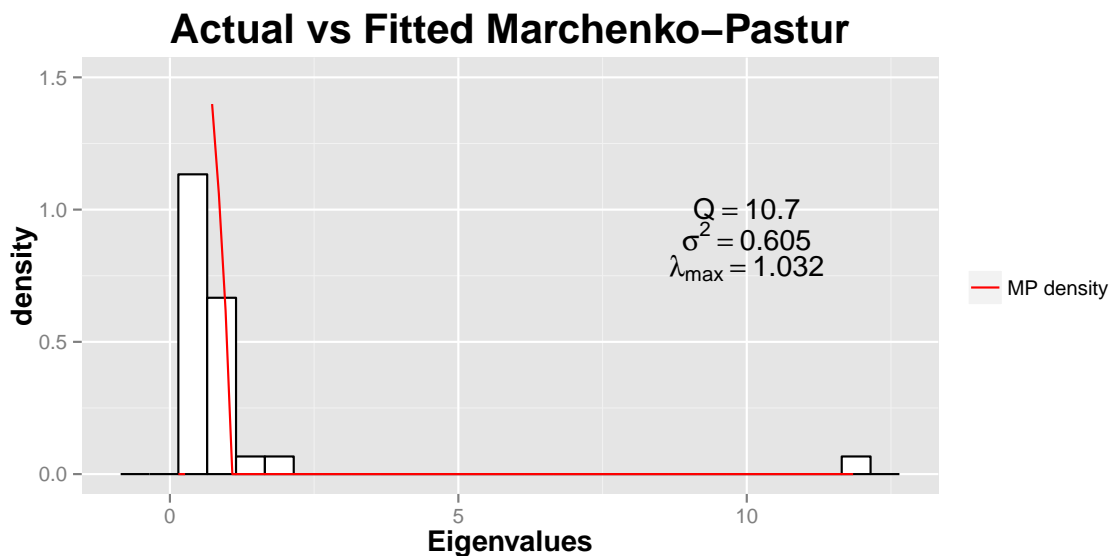
```
estRMT(R, Q = NA, cutoff = c("max", "each"),  
       eigenTreat = c("average", "delete"),  
       numEig=1, parallel = TRUE)
```

This function takes several 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 strategy using all 30 stocks from the Daily Dow Jones 30 index. We will use `dow30data` 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, parallel=FALSE)$cov  
  momentargs$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 specification. 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(dow30data))  
  
#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 let's backtest our strategy using an ordinary covariance matrix and a covariance matrix built by denoising using Random Matrix theory.

```
opt.ordinary <-  
  optimize.portfolio.rebalancing(dow30data, pspec.lo,  
                                optimize_method="quadprog",  
                                rebalance_on="months",  
                                training_period=120,  
                                trailing_periods=120)  
  
opt.rmt <-  
  optimize.portfolio.rebalancing(dow30data, 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 cumulative returns using the `PerformanceAnalytics` package.

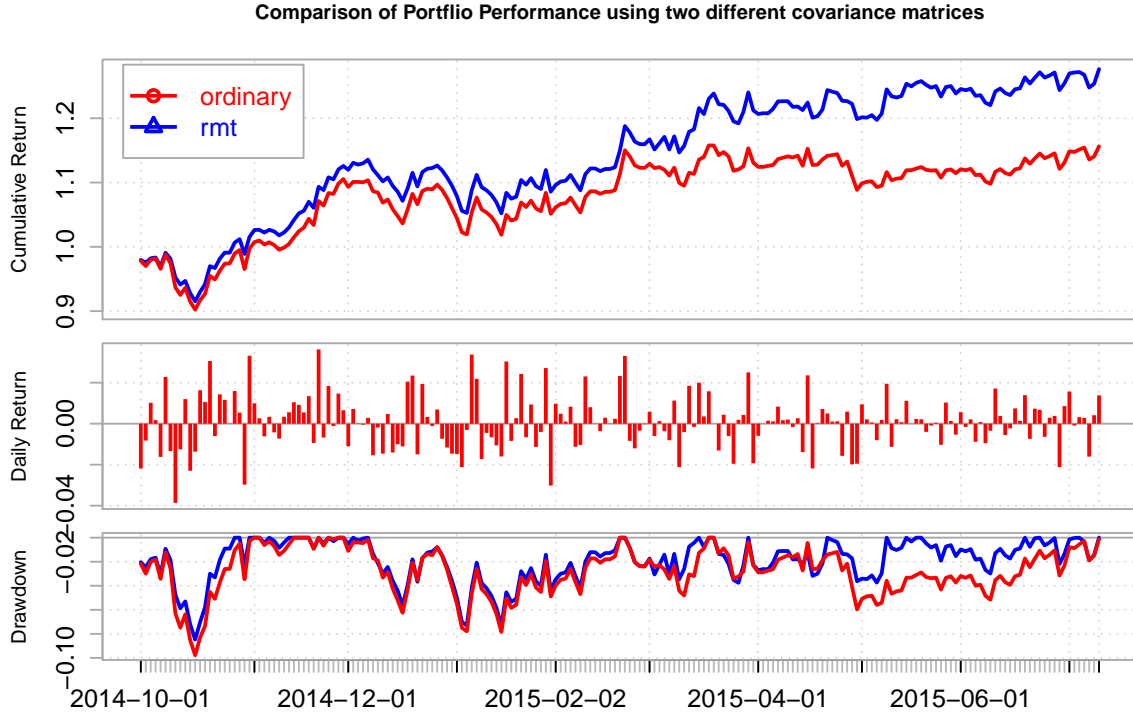
```
ordinary.wts <- na.omit(extractWeights(opt.ordinary))
ordinary <- Return.rebalancing(R=dow30data, weights=ordinary.wts)

rmt.wts <- na.omit(extractWeights(opt.rmt))
rmt <- Return.rebalancing(R=dow30data, weights=rmt.wts)

rmt.strat.rets <- merge.zoo(ordinary,rmt)
colnames(rmt.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(rmt.strat.rets,wealth.index = T,
                           colorset = c("red","blue"),
                           main=paste(c("Comparison of Portfolio ",
                                         "Performance using two ",
                                         "different covariance matrices"),
                                       collapse=""), cex.legend = 1.3,
                           cex.axis = 1.3, legend.loc = "topleft")
```



5 Independent Switching Dynamic Conditional Correlation Model

The IS-DCC model from (Lee 2010) has the same structure as the DCC model but it lets the constants be state dependent and hence makes it possible to model time-varying correlation with different dynamics for each regime. The model runs a separate DCC process for each state in parallel and avoids the path dependency problem and makes the model tractable.

Fitting the IS-DCC model to data corresponds to a two step process, where the first step is to estimate the volatility of each univariate time series separately using GARCH(1,1), as in the case for DCC. The second step corresponds to estimating the DCC(1,1) parameters for each state and estimating the transition probabilities corresponding to the Hidden Markov Model. This is done by maximising the log likelihood using the Expectation Maximization (EM) algorithm.

5.1 Data

To demonstrate the use of IS-DCC we will choose the `etfdata` object which contains daily returns for 9 exchange traded funds (SPDRs) that represent the U.S. stock market

from S&P, i.e. XLE, XLY, XLP, XLF, XLV,XLI, XLB, XLK and XLU. The Sector SPDRs divide the S&P 500 into nine sector index funds. The returns on assets are considered to be dependent on regimes which are in turn defined by market conditions. Daily returns on the ETFs from January 1, 2008 to December 31, 2010 were retrieved from Yahoo Finance.

```
data("etfdata")
```

5.2 Covariance estimation

To fit a covariance matrix we can use the `isdccfit` function.

```
isdccfit(R, numRegimes, transMatbounds = c(2,10),  
         dccBounds = c(0,1), w = NA, ...)
```

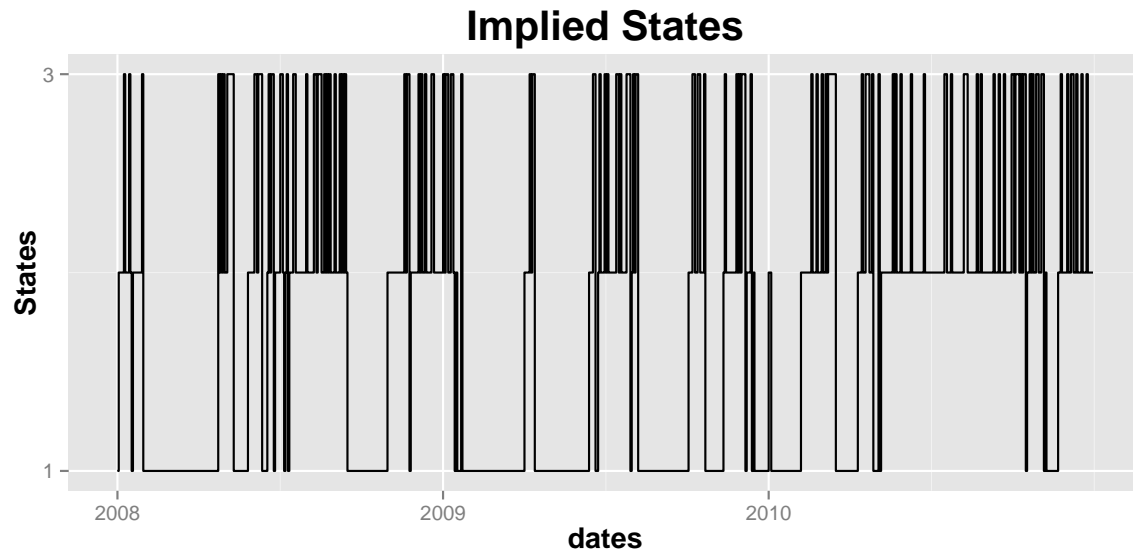
This function takes several options, details of which can be found on the man page. However, in the simplest case we can pass a timeseries object of asset returns and the number of regimes that we want to fit to the data. Since we use Expectation Maximization to fit data, convergence can be slow and additional parameters must be passed to the optimizer to speed by computation. We use `DEoptim` to fit parameters. If no control parameters are passed we use the `lhs` package to generate initial parameters uniformly in the parameters space and pass it as initial population to `DEoptim`.

Let us fit a simple model for demonstration that fits three regimes to the data.

5.3 Plots

Once we have fitted a model we can also investigate the regimes by examining the implied states using the `plot` function. The plot function takes in a fitted model and the type of plot. It then plots either the implied states or the implied probability of regimes.

```
plot(model.isdcc)
```



5.4 Evaluation

We will now build a custom portfolio strategy using `etfdata`. We will use a simple strategy such that at each period we will choose the covariance matrix from the regime which has the highest probability of occurrence. We will benchmark the strategy against the ordinary DCC model that does not use regimes.

Let us first construct two custom moment functions where covariance is built using DCC and IS-DCC

```
custom.portfolio.moments.isdcc <- function(R, portfolio) {

  momentargs <- list()
  momentargs$mu <- matrix(as.vector(apply(R,2, "mean")), ncol = 1)

  result <- isdccfit(R, numRegimes = 2, itermax=100)
  ldate <- as.character(tail(index(R),1))
  maxProbIndex <- which.max(result$filtProb[ldate,])
  momentargs$sigma <- result$cov[[ldate]][[maxProbIndex]]

  momentargs$m3 <- matrix(0, nrow=ncol(R), ncol=ncol(R)^2)
  momentargs$m4 <- matrix(0, nrow=ncol(R), ncol=ncol(R)^3)

  return(momentargs)
}
```

```

custom.portfolio.moments.dcc <- function(R, portfolio) {

  momentargs <- list()
  momentargs$mu <- matrix(as.vector(apply(R,2, "mean")), ncol = 1)

  garch11.spec <- ugarchspec(mean.model = list(armaOrder = c(0,0)),
                             variance.model = list(garchOrder = c(1,1)
                                                    , model = "sGARCH"),
                             distribution.model = "norm")

  dcc.garch11.spec <- dccspec(uspec = multispec( replicate(ncol(R),
                                                         garch11.spec) ),
                             dccOrder = c(1,1), distribution = "mvnorm")

  dcc.fit <- dccfit(dcc.garch11.spec, data = R)
  momentargs$sigma <- rcov(dcc.fit)[,as.character(tail(index(R),1))]

  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 specification. 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.

```

datap <- etfdata["2009-07/"]

pspec.lo.isdcc <- portfolio.spec(assets = colnames(datap))

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

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

```

Now let's backtest our strategy using an ordinary covariance matrix and covariance matrices built using DCC and IS-DCC models.


```

ordinary <-
  optimize.portfolio.rebalancing(datap, pspec.lo.isdcc,
                                optimize_method="quadprog",
                                rebalance_on="months",
                                training_period=120,
                                trailing_periods=120)

opt.dcc <-
  optimize.portfolio.rebalancing(datap, pspec.lo.isdcc,
                                optimize_method="quadprog",
                                momentFUN =
                                  "custom.portfolio.moments.dcc",
                                rebalance_on="months",
                                training_period=120,
                                trailing_periods=120)

opt.isdcc <-
  optimize.portfolio.rebalancing(datap, pspec.lo.isdcc,
                                optimize_method="quadprog",
                                momentFUN =
                                  "custom.portfolio.moments.isdcc",
                                rebalance_on="months",
                                training_period=120,
                                trailing_periods=120)

```

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

```

ord.wts <- na.omit(extractWeights(ordinary))
ord <- Return.rebalancing(R=datap, weights=ord.wts)

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

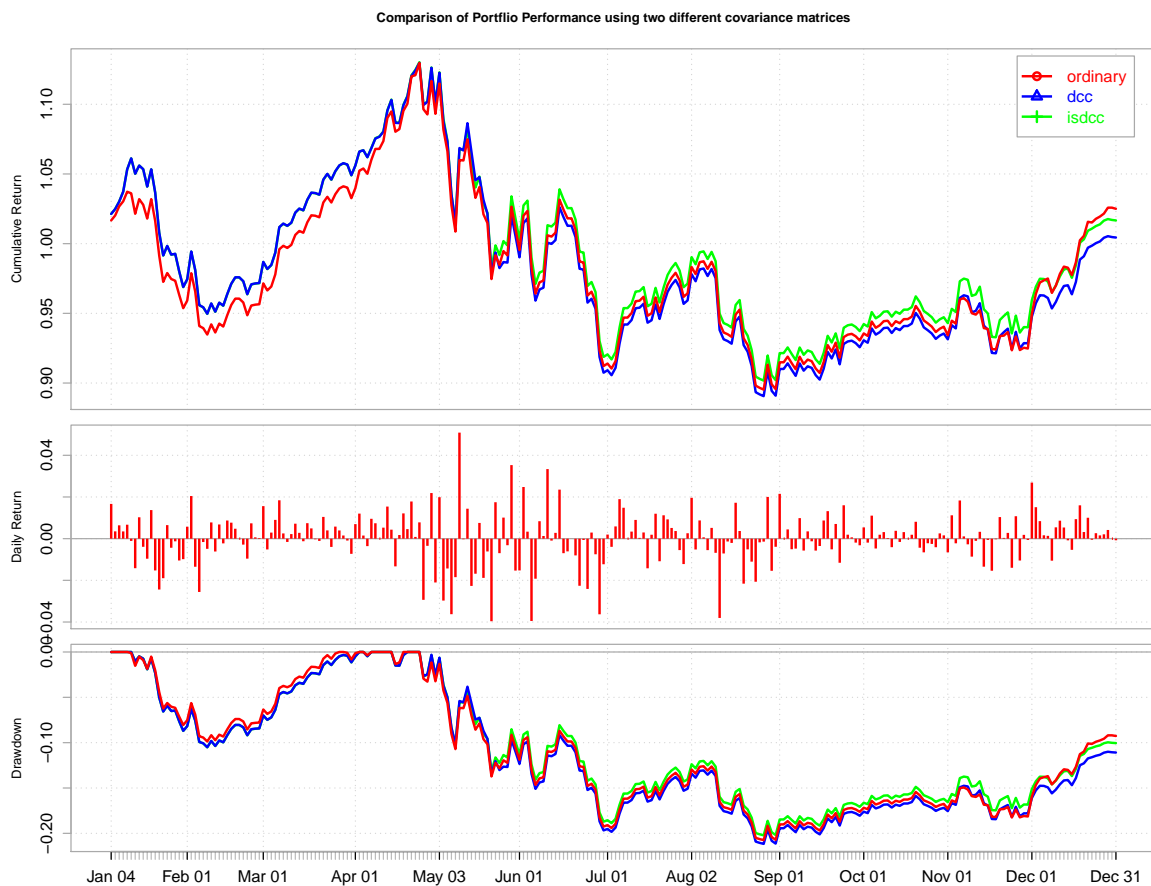
isdcc.wts <- na.omit(extractWeights(opt.isdcc))
isdcc <- Return.rebalancing(R=datap, weights=isdcc.wts)

isdcc.strat.rets <- merge.zoo(merge.zoo(ord, dcc), isdcc)
colnames(isdcc.strat.rets) <- c("ordinary", "dcc", "isdcc")

```

In the chart below we can see that the cumulative returns generated using our strategy with IS-DCC model are superior to ordinary returns as well as returns by the DCC model. This suggests that there is value in assuming the presence of regimes in data and exploring the idea further while optimizing portfolios.

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



6 Eigenvalue Shrinkage in Spiked Covariance Model

The MLE estimator of covariance matrix is not an accurate estimator when the ratio of number of variables to observations is large. It distorts the eigenstructure of the

population covariance matrix such that the largest sample eigenvalue is biased upward and the smallest sample eigenvalue is biased downward. However, empirical eigenvalues can be improved by shrinkage. (Donoho, Gavish, and Johnstone 2013) assume that the population covariance matrix follows a spiked covariance model and construct scalar non-linear shrinkers which shrink eigenvalues greater than the bulk edge of the Marchenko Pastur distribution and set values within bulk to 1. 26 loss functions under different losses and matrix norms are considered.

6.1 Data

To demonstrate the use of shrinkage in Spiked Covariance model we will choose the `rmtdata` object which contains simulated data with multivariate normal distribution. The data is generated as follows. We first generate a spiked covariance matrix with 100 dimensions and the following 15 eigenvalues as spikes, $\lambda \in \{48, 46, 44, 42, 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20\}$. Our spiked covariance model is as follows

$$C = \sum_{\lambda_i \in \lambda} \lambda_i v_i v_i^\top + I_{100}$$

We will use the spiked covariance matrix from above and generate 500 observations from a multivariate normal distribution. This object is saved for ease of use and is called as `rmtdata`.

```
data("rmtdata")
```

6.2 Covariance estimation

To fit a covariance matrix we can use the `estSpikedCovariance` function.

```
estSpikedCovariance(R, gamma = NA,
                    numOfSpikes = NA,
                    method = c("KNTTest", "median-fitting"),
                    norm = c("Frobenius", "Operator", "Nuclear"),
                    pivot = 1, statistical = NA,
                    fit = NA)
```

This function takes several options. In the simplest case we can pass a timeseries object of asset returns where all other parameters assume default value. The parameter `gamma` if missing is set to the ratio of variables to observations. For time series data, the choice of `gamma` can be important and one may want to control it such that the block of returns under consideration is stationary. If `numOfSpikes` is missing, then it is estimated using two methods, `KNTTest` or `median-fitting`. In case of `median-fitting` we

first count the number of breaks in the empirical histogram of eigenvalues using the Freedman-Diaconis algorithm. The initial number of spikes are calculated by counting the number of eigenvalues in the breaks after the first zero. The initial number of spikes are used to match the bulk edge and estimate σ^2 . This serves as a lower bound for σ^2 . σ^2 calculated by assuming no spikes is used as an upper bound. We then estimate variance by minimizing the absolute distance between the true median of the MP distribution and the median of the eigenvalues within the bulk. The details of KNTTest described in (Kritchman and Nadler 2009).

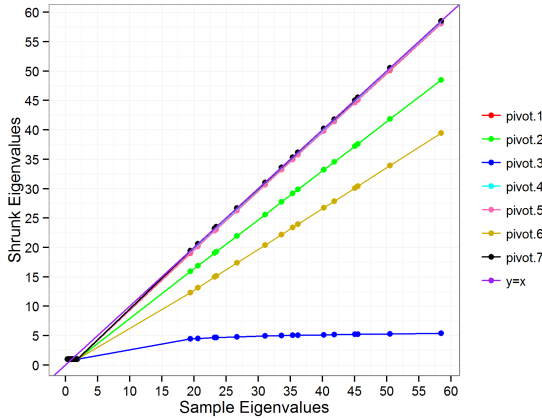
6.3 Plots

We can investigate the performance of 26 shrinkers by plotting them simultaneously

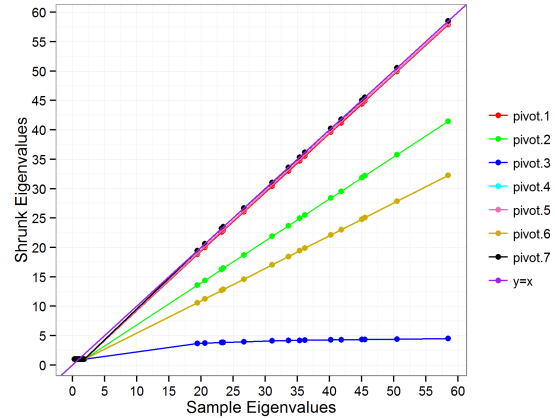
```
plotSpikedCovariance(rmtdata)
```

Optimal Shrinkers for 26 Component Loss Functions

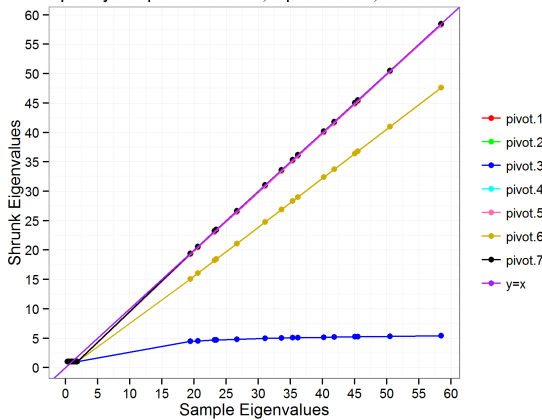
Discrepancy = Frobenius Norm, Spikes = 15, $\sigma^2 = 1.001125$



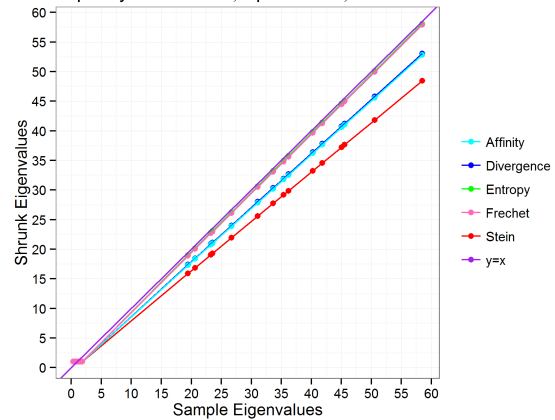
Discrepancy = Nuclear Norm, Spikes = 15, $\sigma^2 = 1.001125$



Discrepancy = Operator Norm, Spikes = 15, $\sigma^2 = 1.001125$



Discrepancy = Statistical, Spikes = 15, $\sigma^2 = 1.001125$



7 Robust Exponential Smoothing of Multivariate Time Series

Classical exponential smoothing is a popular technique used to forecast time series. However, presence of outliers can distort the forecasts leading to bad estimates. Robust methods result in much better forecasts than the classical approach in presence of outliers. (Croux, Gelper, and Mahieu 2010) suggest a methodology to combine these two techniques in a multivariate setting, where forecasting uses information from all components leading to more accurate forecasts.

7.1 Covariance estimation

To estimate a covariance matrix we can use the `estSpikedCovariance` function.

```
robustMultExpSmoothing(R, smoothMat = NA, startup_period = 10,  
                        training_period = 60 , seed = 9999, trials = 50,  
                        method = "L-BFGS-B", lambda = 0.2)
```

The smoothing parameter is optional and will be estimated if missing. To estimate the smoothing matrix we use the constraints that the smoothing matrix is symmetric and its eigenvalues lie between 0 and 1. We also use the fact that the orthogonal matrix in spectral decomposition of smoothing matrix can be parameterized using givens angles. These angles must lie between $-\pi/2$ and $\pi/2$. To estimate the smoothing matrix we set up an optimization problem as described in the paper to minimize the determinant of the covariance of one step ahead forecast errors. The method argument allows one to change the optimization algorithm used. We use the `optimx` package to solve the multivariate optimization problem. The package allows us to choose from 16 different algorithms. Experimental results show that Nelder-Mead and L-BFGS-B perform well for such a noisy function. One also needs to be careful with the reproducibility of the results. Estimation of smoothing matrix may lead to slightly different results on each run. However, the cleaned series or covariance matrix show only marginal differences. To estimate the matrix we start the optimization from random points and the parameter `trials` decides the number of runs and the parameter `seed` can be used for replicate the starting points. One needs to be careful with estimation of smoothing matrix for high dimensional data. The estimation requires to search roughly, dimension squared parameters, which can be slow. Semidefinite-programming could lead to better solutions. However, at the time of writing the package, support for such a solver which is open source in R is challenging to find.

7.2 Evaluation

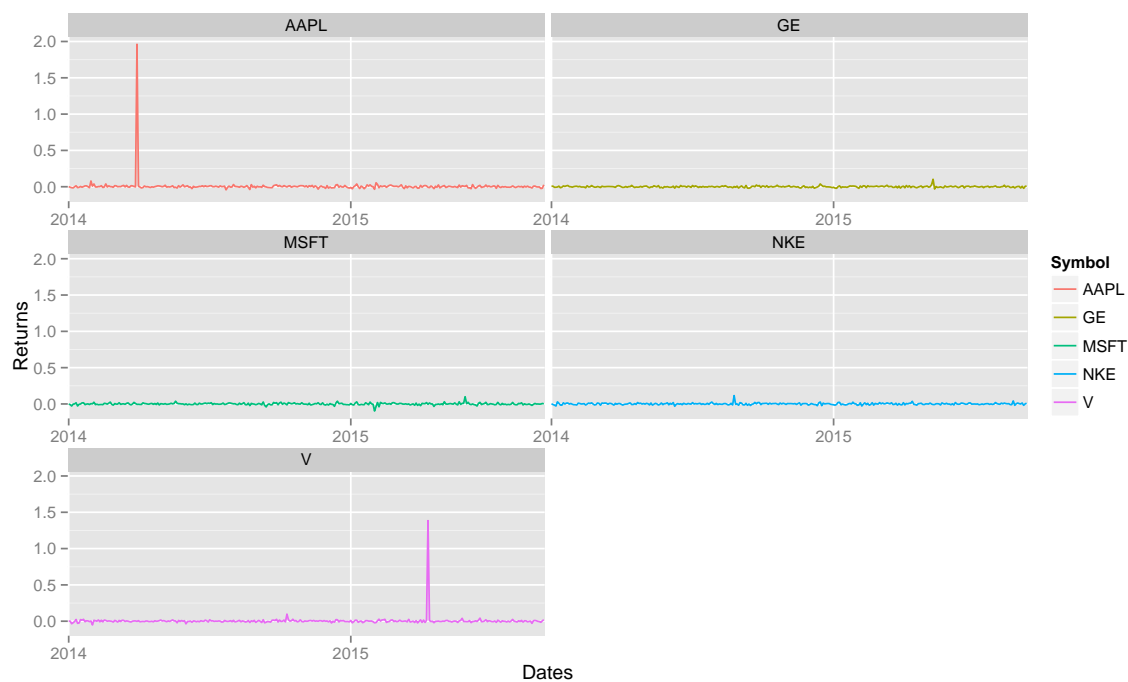
Let us use 5 stocks from Dow Jones 30 data, ie AAPL, GE, MSFT, NKE and V.

```
data("dow30data")
symbols <- c('AAPL', 'GE', 'MSFT', 'NKE', 'V')

R <- dow30data[,which(colnames(dow30data) %in% symbols)]
smoothfit <- robustMultExpSmoothing(R)
```

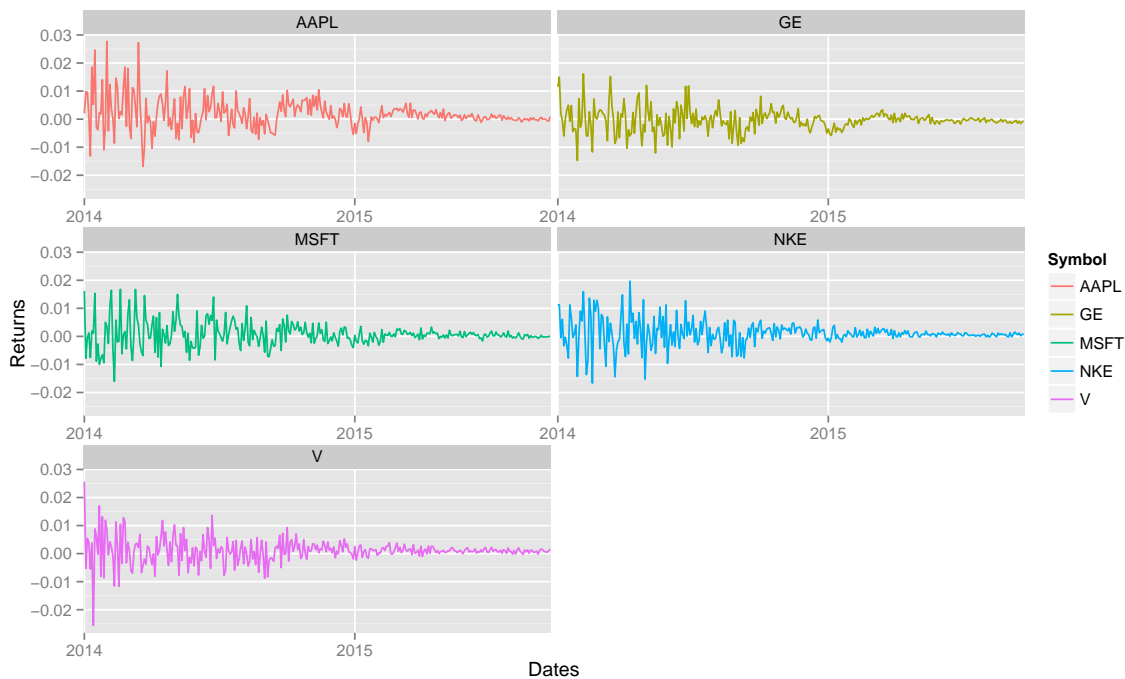
Now let's plot the returns. Notice the spikes in AAPL and V in the regular plot.

```
plotmissing(R)
```



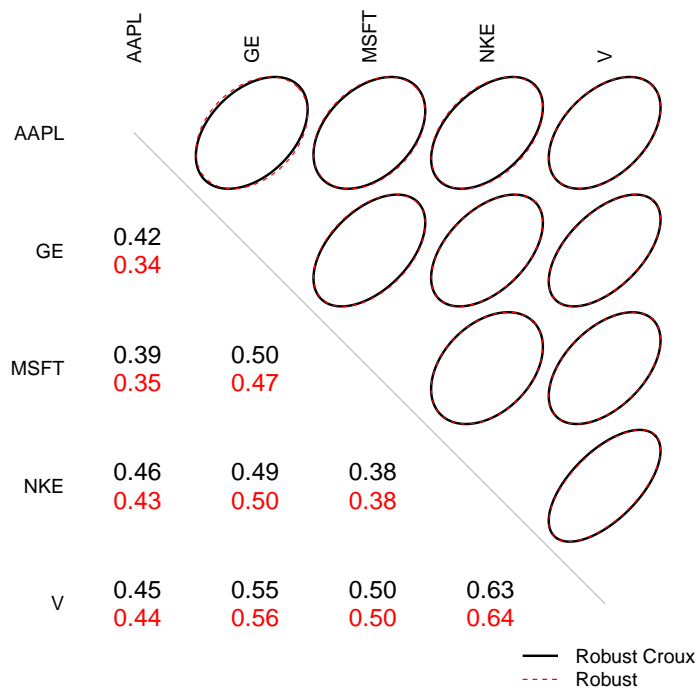
Let us also examine the cleaned time series. Notice that the spikes are missing.

```
plotmissing(smoothfit$cleanValues)
```



Let us also compare the ordinary robust covariance matrix against robust covariance matrix obtained using the procedure outlined in Croux. Notice the similarity between the two estimates.

```
rob <- covRob(R)$cov
compareCov(smoothfit$covMat, rob, labels = c("Robust Croux", "Robust"))
```



References

- Carhart, Mark M. 1997. “On Persistence in Mutual Fund Performance.” *Journal of Finance* 52 (1): 57–82.
- Croux, Christophe, Sarah Gelper, and Koen Mahieu. 2010. “Robust Exponential Smoothing of Multivariate Time Series.” *Computational Statistics & Data Analysis* 54 (12). Elsevier BV: 2999–3006. doi:[10.1016/j.csda.2009.05.003](https://doi.org/10.1016/j.csda.2009.05.003).
- Donoho, David L., Matan Gavish, and Iain M. Johnstone. 2013. “Optimal Shrinkage of Eigenvalues in the Spiked Covariance Model.” *Pre-Print*, November.
- Fama, Eugene F., and Kenneth R. French. 2014. “A Five-Factor Asset Pricing Model.” *SSRN Electronic Journal*. doi:[10.2139/ssrn.2287202](https://doi.org/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.
- Kritchman, S., and B. Nadler. 2009. “Non-Parametric Detection of the Number of Signals: Hypothesis Testing and Random Matrix Theory.” *IEEE Transactions on Signal Processing* 57 (10). Institute of Electrical & Electronics Engineers (IEEE): 3930–41. doi:[10.1109/tsp.2009.2022897](https://doi.org/10.1109/tsp.2009.2022897).
- Lee, Hsiang-Tai. 2010. “Regime Switching Correlation Hedging.” *Journal of Banking & Finance* 34 (11). Elsevier BV: 2728–41. doi:[10.1016/j.jbankfin.2010.05.009](https://doi.org/10.1016/j.jbankfin.2010.05.009).
- 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](https://doi.org/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](https://doi.org/10.1016/s0304-405x(97)00020-2).