Using CMEtest with PortfolioAnalytics

Mohamed Ishmael Diwan Belghazi

April 16, 2014

Abstract

The purpose of this vignette is to show how to pass custom location and scatter estimators to PortfolioAnalytics using CMEtest. In a first part we will show how to pass MLE and robust location and scatter to a global minimum variance object. In a second, we expose how to operate Random Matrix theory type filtering on scatter matrices using CMEtest.

Contents

1 Preliminaries

1.1 Loading packages

Loading Packages and sourcefiles.

```
# Loading packages
suppressMessages(require(PortfolioAnalytics))
# Loading optimization packages
suppressMessages(require(ROI))
suppressMessages(require(ROI.plugin.glpk))

## Warning:  there is no package called 'ROI.plugin.glpk'

suppressMessages(require(ROI.plugin.quadprog))

################# Loading test ##

suppressMessages(library(CMEtest))
options(width = 60)
```

For now, the package tawny is loaded only for the dataset.

1.2 Loading Data

We take 50 observations for 10 assets.

```
################# Loading data ##

# Loading data
data(sp500.subset)
returns <- sp500.subset[1:50, 1:10]
assets <- colnames(returns)
```

1.3 Showing available methods

One can easily query the list of available estimation methods. They come in four categories.

```
CMEhelp()

## ========================
## CMEtest avaiable Methods
## ========================
##
## Smoothing  : None
##  Smoothing  : Boudt
## Estimation : mle
##  Estimation : auto
##  Estimation : mcd
##  Estimation : weighted
##  Estimation : donostah
##  Estimation : pairwiseQC
##  Estimation : pairwiseGK
## Shrinking  : None
## Filtering  : None
##  Filtering  : MP
## -*-*-*-*-*-*-*-*-*-*-*-**-
## Version: 0.1
```

2 Defining portfolio

We define a Global minimum variance long only portfolio with Box Constraint. Since we are optimizing a var objective, we can use the ROI solver.

```
## Specifying portfolio
port_gmv <- portfolio.spec(assets = assets)
## specifying long only constraing
long_const = 0
## specifying unform upper box constraints
upper_box <- 0.6

## Setting constraints
```

```
port_gmv <- add.constraint(portfolio = port_gmv, type = "full_investment",
    enabled = TRUE)
port_gmv <- add.constraint(portfolio = port_gmv, type = "box",
    min = long_const, max = upper_box)
## Adding objective function
port_gmv <- add.objective(portfolio = port_gmv, type = "risk",
    name = "var")

## Showing portfolio specification
print(port_gmv)

## **************************************************
## PortfolioAnalytics Portfolio Specification
## **************************************************
##
## Call:
## portfolio.spec(assets = assets)
##
## Assets
## Number of assets: 10
##
## Asset Names
##  [1] "MMM"  "ABT"  "ANF"  "ADBE" "AMD"  "A"    "APD"  "AKS"
##  [9] "AA"   "AYE"
##
## Constraints
## Number of constraints: 2
## Number of enabled constraints: 2
## Enabled constraint types
##  - full_investment
##  - box
## Number of disabled constraints: 0
##
## Objectives
## Number of objectives: 1
## Number of enabled objectives: 1
## Enabled objective names
##  - var
## Number of disabled objectives: 0
```

We show the portfolio specifications

```
## Showing portfolio specification
print(port_gmv)

## **************************************************
## PortfolioAnalytics Portfolio Specification
## **************************************************
##
## Call:
## portfolio.spec(assets = assets)
##
## Assets
## Number of assets: 10
##
## Asset Names
##  [1] "MMM"  "ABT"  "ANF"  "ADBE" "AMD"  "A"    "APD"  "AKS"
##  [9] "AA"   "AYE"
##
## Constraints
## Number of constraints: 2
## Number of enabled constraints: 2
## Enabled constraint types
##  - full_investment
##  - box
## Number of disabled constraints: 0
##
## Objectives
## Number of objectives: 1
## Number of enabled objectives: 1
## Enabled objective names
##  - var
## Number of disabled objectives: 0
```

3 Using CMEtest

3.1 Construting specification

For the time being only robust estimation is implemented. Smoothing, shrinking and filtering will be implemented when the architecture will have matured.

We specify an mle (classical covariance) specification and call the summary function.

```
## Specifying sample cov

mleCovSpec <- CMEspec(smooth = 'None',
                      estim = 'mle',
                      shrink = 'None',
                      filter = 'None')
class(mleCovSpec)

## [1] "CMEspec"

## Showing summary
summary(mleCovSpec)

## /----------------------------------\
## |Matrix Covariance Estimators test |
## \----------------------------------/
##
##
##   Specification summary
##
##  Smoothing: None
##  Estimation: mle
##  Shrinking: None
##  Filtering: None
##
## -----------------------------------
```

We also specify an minimum covariance determinant robust estimator (mcd). The Minimum Covariance Determinant estimator is a robust estimator of a data sets covariance introduced by Rousseeuw(1984). The idea is to find a given proportion (h) of good observations which are not outliers and compute their empirical covariance matrix. This empirical covariance matrix is then rescaled to compensate the performed selection of observations (consistency step).

```r
robCovSpec <- CMEspec(smooth = "None", estim = "mcd", shrink = "None",
    filter = "None")

class(robCovSpec)

## [1] "CMEspec"

## Showing summary
summary(robCovSpec)

## /-----------------------------------\
## |Matrix Covariance Estimators test |
## \-----------------------------------/
##
##
##   Specification summary
##
##  Smoothing: None
##  Estimation: mcd
##  Shrinking: None
##  Filtering: None
##
## -----------------------------------
```

3.2 Creating moment functions for PortfolioAnalytics

Creating moment functions for Portfolio Analytics is straightforward using CMEtest. It is enough to call the MakeMomentFUN function on the specification object and specify the type of scatter matrix to return. Indeed, if a shrinking or filtering method is selected there is possibly three covariance matrix in a CMEtest object. The type "regular" is the scatter estimated by a mle or a robust method. The type "shrunk", if it exist, is the shrunk previously estimated scatted matrix. The type "filtered" refers to the filtered version of the previously shrunk, or just estimated scatter if no shrinking method were selected. Note that this is also applicable for the accessors "GetLoc" and "GetCor". However, if no type is specified the accessors just return the "regular" scatter (which can be robust or not, depending on the estimation method choosen in the specification).

```r
## Generating Moment functions. These functions will
## dynamically compute location and scatter when passed to
## optimize.portfolio.

MleMomentFUN <- MakeMomentFUN(mleCovSpec, type = "regular")
RobMomentFUN <- MakeMomentFUN(robCovSpec, type = "regular")
```

Note that for RobMomenFun, only the location and scatter are robust. The third and fourth moment are not computed using robust method. It does not matter if the optimization method and/or objective use at most the first two moments. An example where it matters, is when using the Edgeworth or Cornish-Fisher approximation of a non-normal distribution. The latter approximations being based on the four first cumulant.

4 Optimizing the portfolios!

4.1 Dynamic moment functions and optimization

We are now ready to passe the generated moment function. let us optimize the portfolios!

```
#################################### Let's Optimize the potfolios! ##

# mle version
opt_gmv_mle <- optimize.portfolio(R = returns, portfolio = port_gmv,
    optimize_method = "ROI", momentFUN = "MleMomentFUN", trace = TRUE)

## Robust version
opt_gmv_rob <- optimize.portfolio(R = returns, portfolio = port_gmv,
    optimize_method = "ROI", momentFUN = "RobMomentFUN", trace = TRUE)
```

Let us extract the weights For the mle estimator:

```
extractWeights(opt_gmv_mle)

##        MMM        ABT        ANF       ADBE        AMD
##  3.333e-01  3.309e-01  0.000e+00  9.018e-18  2.263e-18
##          A        APD        AKS         AA        AYE
##  7.747e-02 -3.090e-17  6.127e-02  4.973e-18  1.970e-01
```

For the mcd estimator:

```
extractWeights(opt_gmv_rob)

##       MMM       ABT       ANF      ADBE       AMD         A
## 2.754e-01 4.130e-01 4.865e-17 1.914e-16 3.182e-18 1.799e-01
##       APD       AKS        AA       AYE
## 1.047e-16 4.625e-02 0.000e+00 8.543e-02
```

4.2 Computed moment functions and optimization

Robust moment computing can be expensive. Thankfully, It is also possible to generate pre-computed moment functions. The process is very straightforward, instead of calling the moment making function on a CMEtest specification object it is enough to call it on a CMEtest estimation object.

Let us first explicitly compute the covariance. In order to do so, one has to use the Estimate() generic function on the specification object and on the choosen dataset.

```
## Let us compute the covariances
mleCovEst <- Estimate(mleCovSpec, returns)
class(mleCovEst)

## [1] "CMEest"

robCovEst <- Estimate(robCovSpec, returns)
class(robCovEst)

## [1] "CMEest"
```

Althought it is not necessary, let's get the empirical and robust correlations just for fun.

MLE estimator correlation:

```
## Using correlation getter on the estimation object.
GetCor(mleCovEst)

##           MMM     ABT     ANF    ADBE    AMD       A
## MMM  1.000000 0.38502 0.60797 0.47238 0.4823 0.48520
## ABT  0.385018 1.00000 0.14786 0.39328 0.1054 0.23548
## ANF  0.607971 0.14786 1.00000 0.46896 0.6639 0.35549
## ADBE 0.472381 0.39328 0.46896 1.00000 0.4371 0.51525
## AMD  0.482334 0.10536 0.66387 0.43709 1.0000 0.48161
## A    0.485197  0.23548 0.35549  0.51525 0.4816 1.00000
## APD  0.608784  0.31163 0.44302  0.39229 0.5097 0.32069
## AKS  0.006833 -0.20216 0.02507 -0.01556 0.0716 0.09742
## AA   0.405074  0.00953 0.19542  0.35568 0.2540 0.31496
## AYE  0.386649  0.27902 0.14487  0.32792 0.1103 0.27224
##         APD       AKS      AA    AYE
## MMM  0.6088  0.006833 0.40507 0.3866
## ABT  0.3116 -0.202156 0.00953 0.2790
## ANF  0.4430  0.025067 0.19542 0.1449
## ADBE 0.3923 -0.015555 0.35568 0.3279
## AMD  0.5097  0.071597 0.25399 0.1103
## A    0.3207  0.097422 0.31496 0.2722
## APD  1.0000  0.390666 0.54052 0.4898
## AKS  0.3907  1.000000 0.48921 0.2614
## AA   0.5405  0.489209 1.00000 0.4067
## AYE  0.4898  0.261419 0.40675 1.0000
```

Robust Mcd estimator correlation:

```
GetCor(robCovEst)

##          MMM      ABT       ANF     ADBE     AMD      A
## MMM  1.00000  0.40230  0.523854 0.480907 0.37599 0.7757
## ABT  0.40230  1.00000  0.473627 0.809454 0.21407 0.3731
## ANF  0.52385  0.47363  1.000000 0.669024 0.53826 0.6112
## ADBE 0.48091  0.80945  0.669024 1.000000 0.49305 0.5996
## AMD  0.37599  0.21407  0.538261 0.493053 1.00000 0.6301
## A    0.77566  0.37309  0.611197 0.599573 0.63010 1.0000
## APD  0.49211  0.48542  0.477815 0.540085 0.50858 0.5416
## AKS  0.02545 -0.15144 -0.001589 0.003096 0.02982 0.0249
## AA   0.38064  0.05229  0.450324 0.252668 0.43525 0.3957
## AYE  0.59107  0.43439  0.662041 0.586931 0.45350 0.4908
##         APD       AKS      AA    AYE
## MMM  0.4921  0.025448 0.38064 0.5911
## ABT  0.4854 -0.151441 0.05229 0.4344
## ANF  0.4778 -0.001589 0.45032 0.6620
## ADBE 0.5401  0.003096 0.25267 0.5869
## AMD  0.5086  0.029816 0.43525 0.4535
## A    0.5416  0.024896 0.39566 0.4908
## APD  1.0000  0.263010 0.32993 0.7560
## AKS  0.2630  1.000000 0.47584 0.4217
## AA   0.3299  0.475839 1.00000 0.4386
## AYE  0.7560  0.421735 0.43862 1.0000
```

Similarly, we can get the first moment. For the classical estimation:

```
GetLoc(mleCovEst)

##        MMM        ABT        ANF       ADBE        AMD
## -0.0015708  0.0010227 -0.0040658  0.0003023 -0.0090793
##          A        APD        AKS         AA        AYE
##  0.0022011 -0.0011369 -0.0077219 -0.0054409 -0.0024925
```

For the robust estimation:

```
GetLoc(robCovEst)

##        MMM        ABT        ANF       ADBE        AMD
## -0.0037958  0.0005135 -0.0088657 -0.0027712 -0.0121705
##          A        APD        AKS         AA        AYE
## -0.0014463 -0.0034711 -0.0099621 -0.0108129 -0.0038915
```

4.3 Generating precomputed moment functions

Now that we have the estimation object, generating the precomputed moment functions is as easy as before. One has just to call the MakeMomentFUN function on the CMEtest estimation object.

```
## Now we create Precomputed moment functions
MlePrecompMomentFUN <- MakeMomentFUN(mleCovEst, type = "regular")
RobPrecompMomentFUN <- MakeMomentFUN(robCovEst, type = "regular")
```

4.4 Optimizing portfolio with precomputed moment functions

```
# mle version
opt_gmv_mle <- optimize.portfolio(R = returns, portfolio = port_gmv,
    optimize_method = "ROI", momentFUN = "MlePrecompMomentFUN",
    trace = TRUE)

## Robust version
opt_gmv_rob <- optimize.portfolio(R = returns, portfolio = port_gmv,
    optimize_method = "ROI", momentFUN = "RobPrecompMomentFUN",
    trace = TRUE)


extractWeights(opt_gmv_mle)

##        MMM        ABT        ANF       ADBE        AMD
##  3.333e-01  3.309e-01  0.000e+00  9.018e-18  2.263e-18
##          A        APD        AKS         AA        AYE
##  7.747e-02 -3.090e-17  6.127e-02  4.973e-18  1.970e-01

extractWeights(opt_gmv_rob)

##        MMM        ABT        ANF       ADBE        AMD
##  2.937e-01  3.000e-01  0.000e+00 -9.872e-17  1.273e-17
##          A        APD        AKS         AA        AYE
##  1.828e-01  9.901e-02  1.245e-01  8.495e-18 -1.135e-17
```

5 Filtering with Random Matrix Theory

5.1 Usefulness and methodology

The quality of a dataset is defined as the ratio of returns to the number of assets. Having a reliable estimate of scatter is especially difficult when the number of assets is large as it will be very difficult to get enough observations to compensate for the quality of the dataset. In this case, it is hard to distinguish between measurement error, or noise, and true correlation.

A way to disentangle the noise from the estimation is to compare the density of the spectrum of the estimated correlation to that of a positive definite symmetric matrix with random Gaussian entries.

Fortunately, the distribution of the density of the eigenvalues of such a random matrix is known. It is called the Marchenko-Pastur distribution(MP) with two parameter σ and Q (the quality which is also a shape parameter).

The recipe is then to compare the two spectrums. All eigenvectors corresponding to eigenvalues below the MP eigenvalue(or the rightmost point if its support) are considered noisy. It is then enough, to spike the matrix by flattening the eigenvalues under the MP eigenvalue.

In practice difficulties arise when we try to fit the MP distribution to the empirical density of the eigenvalues. Trying to get maximum likelihood estimate of the MP density is quite unstable for two major reasons. Firstly, it is difficult to estimates the empirical density of distribution. Histograms and kernel methods, even in one dimension, involves in this case a rather large statistical error since the bulk of the eigenvalues will be strongly clustered while some very significative eigenvalues will be spread out. Secondly, some outlying eigenvalues are order of magnitude grater than the other eigenvalues. Typically, in financial data, those outlying eigenvalues corresponds to the market and sectors.

We offer two ways of fitting the MP distribution. The first one, following bouchaud(2000) is to just substract the contribution of the market eigenvalue of the variance to get σ^2 and use the quality of the dataset for Q. We call this method "analogic". In the second one, we infer the parameters by minimizing the Cramer-Von-Mises criterion between the cumulative distribution of the MP distribution and the empirical distribution of the eigenvalues. This is a minimum distance estimation. We call this method "MDE".

5.2 Specifying a CMEtest object for MP filtering

We will use the whole data set, 200 observation for 75 assets, for a quality of 2.6666. We will fit the distribution using the MDE method and we choose not to exclude the market eigenvalues, even if it is clearly an outlier. We will filter an mle correlation. Of course we could have filtered any available robust method instead.

```
data(sp500.subset)
filteredCorSpec <- CMEspec(estim = "mle", estimCtrl = list(corr = TRUE),
    filter = "MP", filterCtrl = list(fit.type = "MDE", norm.meth = "partial",
        exclude.market = FALSE))
```

The norm.meth variable refers to which eigenvalue flattening methodology we use. "partial" refers to replacing the noise eigenvalues by their average. If "full" is used then, the noise eigenvalues are replaced by one and then all the eigenvalues are renormalized so that their sum is equal to the number of assets.

5.3 Making filtered moment function for Portfolio Analytics

Now to get a filtered moment function is enough to do as above and replace "regular" by "filtered".

```
filteredMleMomentFUN <- MakeMomentFUN(filteredCorSpec, type = "filtered")
```

Please to note, that CMEtest takes care of converting correlation to covariance and conversly when it is necessary. Now it is enough to pass the function to a potfolio optimization as done previsouly.

5.4 Estimating and other uses

Let us estimates the specification.

```
filteredCorEst <- Estimate(filteredCorSpec, sp500.subset)
```

Let us get the filtered correlation matrix

```
head(GetCor(filteredCorEst, "filtered"))

##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]
## [1,] 1.0000 0.6289 0.5647 0.6777 0.4393 0.6674 0.6653
## [2,] 0.6289 1.0000 0.4359 0.5668 0.3273 0.5683 0.5372
## [3,] 0.5647 0.4359 1.0000 0.5838 0.4547 0.5573 0.5702
## [4,] 0.6777 0.5668 0.5838 1.0000 0.5702 0.7123 0.7087
## [5,] 0.4393 0.3273 0.4547 0.5702 1.0000 0.5012 0.4929
## [6,] 0.6674 0.5683 0.5573 0.7123 0.5012 1.0000 0.7073
##        [,8]   [,9]  [,10]  [,11]  [,12]  [,13]  [,14]
## [1,] 0.6121 0.6555 0.6206 0.5491 0.6070 0.6885 0.5965
## [2,] 0.4634 0.5387 0.5649 0.4022 0.4767 0.6503 0.4774
## [3,] 0.5217 0.5417 0.4219 0.5063 0.5432 0.5319 0.4728
## [4,] 0.6872 0.6946 0.6466 0.6672 0.7088 0.7016 0.6375
## [5,] 0.5058 0.4949 0.3629 0.5107 0.5568 0.4401 0.4266
## [6,] 0.6891 0.7089 0.6492 0.5980 0.6497 0.6834 0.6643
##       [,15]  [,16]  [,17]  [,18]  [,19]  [,20]  [,21]
## [1,] 0.5933 0.6356 0.5091 0.6504 0.5340 0.6162 0.5122
## [2,] 0.5646 0.5231 0.4486 0.6979 0.5789 0.5497 0.4000
## [3,] 0.5095 0.6050 0.5337 0.4748 0.3473 0.5488 0.3929
## [4,] 0.5474 0.6243 0.5417 0.5760 0.4769 0.6524 0.5597
## [5,] 0.3650 0.4168 0.4398 0.3288 0.2278 0.4367 0.3880
## [6,] 0.5663 0.6175 0.4925 0.6006 0.4934 0.6050 0.5967
##       [,22]  [,23]  [,24]  [,25]  [,26]  [,27]  [,28]
## [1,] 0.6890 0.6215 0.4949 0.5632 0.6702 0.7125 0.4785
## [2,] 0.6234 0.5689 0.3600 0.5727 0.6215 0.5814 0.4252
## [3,] 0.4898 0.5342 0.4916 0.4833 0.5890 0.6290 0.5189
## [4,] 0.7013 0.6555 0.5747 0.5360 0.6962 0.7855 0.5104
## [5,] 0.4084 0.4477 0.4811 0.4061 0.4848 0.5676 0.4465
## [6,] 0.7250 0.6110 0.5475 0.5394 0.6513 0.7417 0.4647
##       [,29]  [,30]  [,31]  [,32]  [,33]  [,34]  [,35]
## [1,] 0.6415 0.6075 0.5679 0.6212 0.6448 0.6811 0.5886
## [2,] 0.5355 0.6081 0.4623 0.5012 0.5454 0.6113 0.5909
## [3,] 0.5488 0.4855 0.5697 0.5539 0.4986 0.5475 0.3408
## [4,] 0.7025 0.5773 0.6189 0.7002 0.6975 0.6988 0.5955
## [5,] 0.4645 0.3577 0.4785 0.5092 0.4854 0.4391 0.3179
## [6,] 0.6529 0.5569 0.5669 0.6455 0.7126 0.6785 0.6127
##       [,36]  [,37]  [,38]  [,39]  [,40]  [,41]  [,42]
## [1,] 0.6443 0.6549 0.4796 0.4211 0.6487 0.5945 0.6027
## [2,] 0.5222 0.5907 0.4724 0.3387 0.5595 0.4722 0.5413
## [3,] 0.5704 0.6139 0.4740 0.4408 0.6196 0.5617 0.4894
## [4,] 0.6762 0.6231 0.5108 0.5339 0.6416 0.6327 0.6479
## [5,] 0.4391 0.4048 0.4230 0.5523 0.4635 0.4619 0.4777
## [6,] 0.6435 0.6003 0.4436 0.4661 0.6271 0.6051 0.6214
##       [,43]  [,44]  [,45]  [,46]  [,47]  [,48]  [,49]
## [1,] 0.4351 0.6320 0.5672 0.6702 0.6306 0.6694 0.6143
## [2,] 0.3632 0.6308 0.5075 0.5830 0.5194 0.6176 0.4954
## [3,] 0.4470 0.4651 0.5444 0.5605 0.5555 0.5655 0.4719
## [4,] 0.5372 0.6278 0.5893 0.7256 0.7131 0.6506 0.6637
## [5,] 0.5423 0.3763 0.4651 0.5181 0.5313 0.3923 0.4421
## [6,] 0.4764 0.6051 0.5685 0.6896 0.6587 0.6319 0.6909
##       [,50]  [,51]  [,52]  [,53]  [,54]  [,55]  [,56]
## [1,] 0.5974 0.6647 0.6818 0.6658 0.6233 0.5705 0.5257
## [2,] 0.4243 0.5467 0.5677 0.5275 0.5083 0.4291 0.3618
## [3,] 0.6536 0.6543 0.6171 0.5769 0.5786 0.6383 0.5615
## [4,] 0.6137 0.6811 0.7224 0.7655 0.6296 0.5889 0.6083
## [5,] 0.4614 0.4916 0.5426 0.5648 0.4285 0.4845 0.5308
## [6,] 0.5926 0.6436 0.7049 0.7077 0.6309 0.5569 0.5711
##       [,57]  [,58]  [,59]  [,60]  [,61]  [,62]  [,63]
## [1,] 0.5279 0.7083 0.4219 0.5742 0.5789 0.6728 0.5067
## [2,] 0.4135 0.7417 0.4146 0.4156 0.4316 0.6722 0.3690
## [3,] 0.5786 0.4979 0.3471 0.6276 0.5559 0.5369 0.5985
## [4,] 0.5211 0.6697 0.4355 0.5971 0.6728 0.6482 0.5493
## [5,] 0.4358 0.3876 0.3326 0.4237 0.5448 0.4354 0.4656
## [6,] 0.5205 0.6718 0.4218 0.5553 0.6196 0.6363 0.4909
##        [,64]  [,65]  [,66]  [,67]  [,68]  [,69]  [,70]
## [1,] 0.5811 0.5391 0.3484 0.6303 0.6006 0.6175 0.6219
## [2,] 0.4259 0.4486 0.2268 0.4909 0.4948 0.5758 0.4689
## [3,] 0.5344 0.5787 0.2341 0.5805 0.4419 0.4889 0.5182
## [4,] 0.6596 0.5634 0.3569 0.6762 0.6414 0.6211 0.6899
## [5,] 0.4837 0.4177 0.1461 0.4702 0.4046 0.4518 0.4544
## [6,] 0.6229 0.5057 0.4124 0.6489 0.6723 0.6318 0.6804
##        [,71]  [,72]  [,73]  [,74]  [,75]
## [1,] 0.6216 0.6868 0.6747 0.6060 0.6037
## [2,] 0.5040 0.6806 0.6309 0.6005 0.5808
## [3,] 0.4652 0.5421 0.5884 0.4010 0.5276
## [4,] 0.6638 0.6769 0.6781 0.6093 0.5871
## [5,] 0.4077 0.4390 0.4960 0.3521 0.4468
## [6,] 0.6912 0.6588 0.6603 0.6155 0.5877
```

Let us get all the non-noisy eigenvalues and eigenvectors.

```
print(filteredCorEst$.filterEstim$signalEigVals)

## [1] 42.149  4.219  2.635  1.674  1.503  1.293

print(head(filteredCorEst$.filterEstim$signalEigVecs))

##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] -0.12501 -0.019796  0.104109  0.004828 -0.015438
## [2,] -0.10795 -0.042980  0.268636 -0.181785 -0.012264
## [3,] -0.10891  0.119508 -0.007542  0.106655 -0.129703
## [4,] -0.13126 -0.007124 -0.016611  0.097015  0.107366
## [5,] -0.09285  0.079070 -0.114914  0.149570 -0.006528
## [6,] -0.12718 -0.058873 -0.005950  0.059643  0.005534
##           [,6]
## [1,] -0.03728
## [2,]  0.08238
## [3,] -0.08282
## [4,]  0.10796
## [5,]  0.36968
## [6,]  0.03220
```

Now this can be useful, as those eigenvectors can be seen as risk factors. As such, they can be either wrapped and be used to FactorAnalytics or be used within CMEtest in the Factor Model Monte Carlo methodology for unequal data.