Using Robust Covariance Matrix Estimators in PortfolioAnalytics

Yifu Kang and Doug Martin

August 23, 2024

Contents

Contents

1 Introduction

This vignette describes a method for using robust covariance matrix estimators when constructing a *fully-invested, long-only global minimum variance portfolio* (GmvLO) with the *PortfolioAnalytics* package. The method can also be used in a straightforward manner for any constrained minimum variance (MinVar) portfolio. We refer to any MinVar portfolio based on the use of a robust covariance matrix as a *robust MinVar portfolio*, with robust GmvLO portfolios as a leading special case.

The R code `demo_robustCovMatForPA.R` in the PortfolioAnalytics `demo` folder, reproduces the results in this Vignette.

It is well-known that outliers can adversely influence all classical estimates, including sample means and sample covariance matrix estimators, which are foundation elements of MinVar portfolios. It follows that outliers can adversely influence the performance of MinVar portfolios constructed using the classical returns means and covariance matrix estimators. With the availability of the robust covariance matrix estimators, which are not much influenced by outliers, PortfolioAnalytics users can construct *robust MinVar portfolios* which are not much influenced by outliers, and evaluate their performance relative to classic MVO portfolios.

The following "custom" robust mean vector and covariance matrix estimator functions are provided in PortfolioAnalytics:

1. `custom.covRob.MM`, using the function `covRobMM` in the RobStatTM package
2. `custom.covRob.Rocke`, using the function `covRobRocke` in the RobStatTM package
3. `custom.covRob.Mcd`, using the function `covMcd` in the robustbase package
4. `custom.covRob.TSGS`, using the function `TSGS` in the GSE package.

The above "custom" functions were are made possible by existence of a very useful "custom moments" functionality in PortfolioAnalytics, which is described in the Vignette "Custom Moment and Objective Functions" by Ross Bennett (2018), available on CRAN at https://cran.r-project.org/web/packages/PortfolioAnalytics/index.html. In the context of a returns mean vector and covariance matrix, the latter are thought of as the moments of orders one and two, loosely speaking the "mean" and "variance".

It is important to keep in mind that the methods described herein for using robust covariance matrix estimators, implicitly including corresponding robust mean vector estimators, may be used for other important covariance matrix estimators, such as shrinkage covariance matrix estimators, and covariance matrix estimators for unequal asset returns histories.

In Section 2 below, we show how to load the R packages and data needed for this Vignette, and how to specify and compute a classic *fully invested global minimum variance long-only portfolio* (GmvLO). In Section 3 we discuss the details of the above four custom robust covariance matrix estimator functions, and describe how to use them in calls to the PortfolioAnalytics `optimize.portfolio` function. In Section 4 we present cumulative gross returns back-tests of robust GmvLO portfolios based on each of the four types of robust covariance matrix estimators. Section 5 briefly describes how users can write their own simple custom robust covariance matrix estimator functions, which they may wish to use for their own special purposes.

Sections 3 and 4 show clearly that performance improvements to classic GmvLO portfolios based on sample mean and sample covariance estimators, can be obtained using robust covariance matrix estimator based GmvLO portfolios. These results are motivation for future in-depth empirical studies of robust constrained MinVar portfolios using PortfolioAnalytics.

2 Compute Classic GmvLO Portfolio Weights

2.1 Load Packages and Data

First, you load the packages needed for this Vignette with the code:

```
library(PCRA)
library(PortfolioAnalytics)
library(CVXR)
library(xts)
```

where it is assumed that the user has already installed those CRAN package on their computer.

In this vignette, we will use weekly returns for either 10 or 30 small capitalization CRSP® stocks, which are freely available as part of the `stocksCRSP` data set contained in the `PCRA` package.[1] The time period used here is the seven year period from January 6, 2006 to December 28, 2012.

```
stocksCRSPweekly <- getPCRAData("stocksCRSPweekly")
dateRange    <- c("2006-01-01", "2012-12-31")
stockItems <- c("Date", "TickerLast", "CapGroupLast", "Return", "MktIndexCRSP")
returnsAll <- selectCRSPandSPGMI("weekly",
                                 dateRange = dateRange,
                                 stockItems = stockItems,
                                 factorItems = NULL,
                                 subsetType = "CapGroupLast",
                                 subsetValues = "SmallCap",
                                 outputType = "xts")
returns <- returnsAll[,1:30]
MARKET <- returnsAll[, 107]
returns10 <- returnsAll[140:300, 21:30]
range(index(returns))
```

```
## [1] "2006-01-06" "2012-12-28"
```

```
range(index(returns10))
```

```
## [1] "2008-09-05" "2011-09-30"
```

In order to get a feeling for the data, we plot below the weekly time series of the `MARKET`, followed by a time series plot of the last 10 of the 30 stocks in `returns`, for the entire time period from 2006-01-06 to 2012-12-28.

```
tsPlotMP(MARKET)
```

[1] CRSP® is the acronym for the Center for Research in Security Prices, LLC.

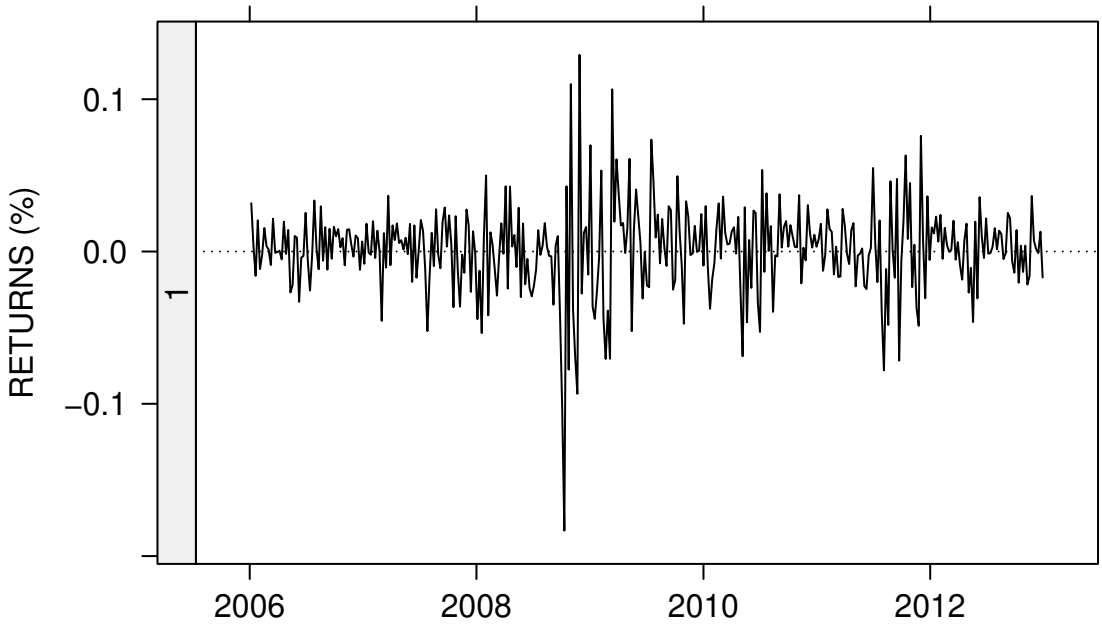


Figure 1. Weekly MARKET Returns from 2006-01-06 to 2012-12-28.

The `MARKET` time series reflects the volatility and disruption of the financial crisis which began late 2007 and early 2008, exploded in late 2008, and slowly subsided toward the end of 2009, with various lessor after-shocks lasting through 2011.

```
tsPlotMP(returns[, 21:30])
```

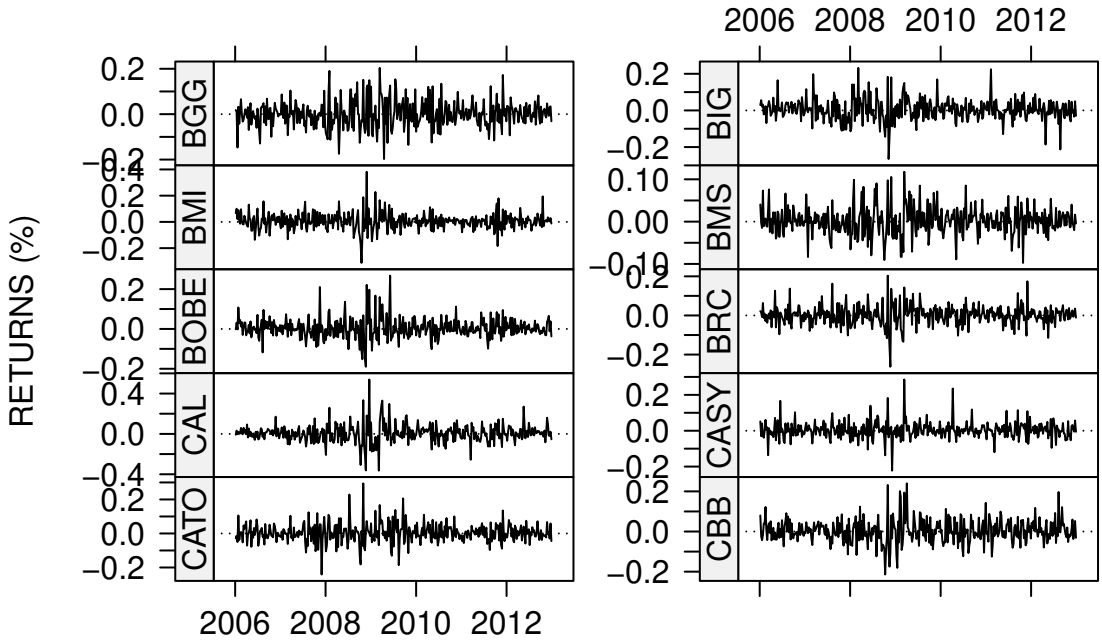


Figure 2. Weekly‘ Returns of 10 Smallcap Stocks from 2006-01-06 to 2012-12-28.

We plotted only the last 10 of the 30 stocks in `returns` just for the sake of a reasonably small plot size.[2] It is evident that the volatility and non-stationarity of the `MARKET` returns persist to various degrees in those 10 stocks.

In the back-tests of Section 4 we use all 30 stocks in `returns` for the full time period. But for purposes of introducing and evaluating the various robust portfolios based on several different robust covariance matrices in Section 3, we use those same 10 stocks as in the above figure, but only for the shorter time period 2008-09-05 to 2011-09-30.

```
tsPlotMP(returns10)
```

[2]The reader may experiment with plots of larger numbers of stocks with the code for this Vignette, which will be available in the Demo folder of the current release of PortfolioAnalytics.

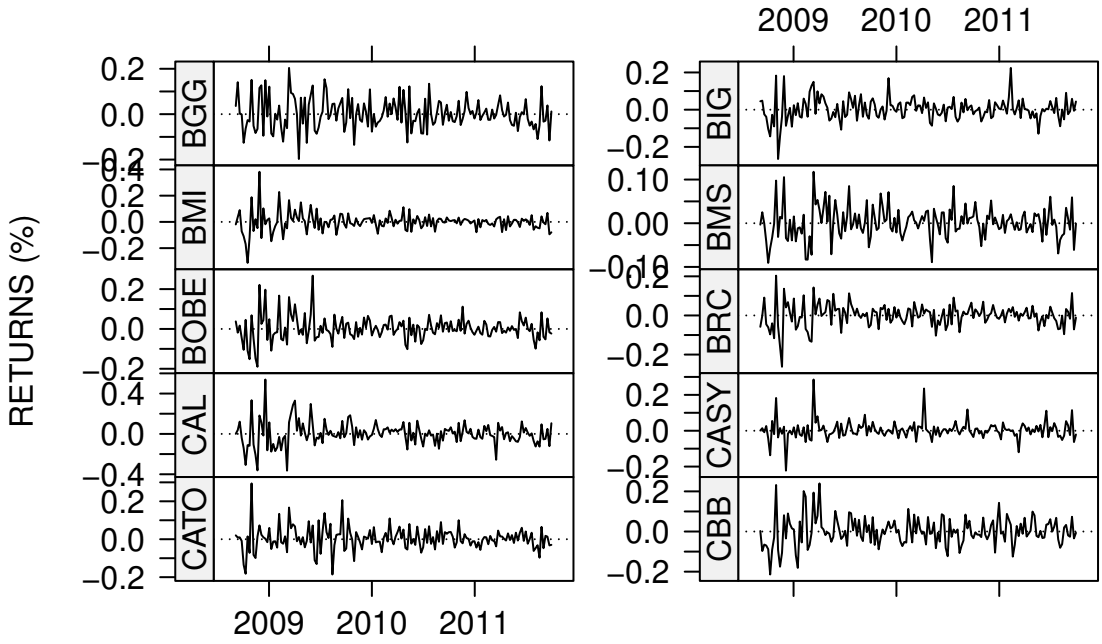


Figure 3. Weekly Returns of 10 Smallcap Stocks from 2008-09-05 to 2011-09-30.

2.2 Compute Classic GmvLO Weights

The mathematical formulation of a GmvLO portfolio is:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \mathbf{w}'\mathbf{C}\mathbf{w} \\ \text{s.t.} \quad & \mathbf{e}'\mathbf{w} = 1 \\ & \mathbf{w} \geq 0 \end{aligned} \tag{1}$$

where $\mathbf{w}$ is the portfolio weights vector, $\mathbf{C}$ is the sample covariance matrix of the returns, and $\mathbf{e}$ is a column vector of 1's.

You create the basic GmvLO portfolio specification object pspec with the following code.

```
funds <- colnames(returns10)
pspec <- portfolio.spec(funds)
pspec <- add.constraint(pspec, type="full_investment")
pspec <- add.constraint(pspec, type="long_only")
pspec <- add.objective(pspec, type="risk", name="var")
```

Then you compute the classic GmvLO portfolio weights and the standard deviation (StdDev) of the portfolio, based on the sample mean vector and sample covariance matrix, and store the result in the object `opt` with the code line:

```
opt <- optimize.portfolio(returns10, pspec, optimize_method = "CVXR")
```

Here the optional argument `optimize_method = "CVXR"` specifies use of an optimization solver from the package *CVXR*. For a given portfolio optimization problem it is often possible to specify any one of a number of different types of solver type, with some default solver. The default solver for minimum variance portfolios is the OSQP solver https://osqp.org/. For the solvers that may be used with CVXR see https://cvxr.rbind.io/, and for details on how to specify a CVXR solver in PortfolioAnalytics, see the Vignette "CVXR for PortfolioAnalytics" by Xinran Zhao.

Use of the function `class()` with the argument `opt`, reveals that the class of the `opt` object is `optimize.portfolio.CVXR`, and use of the following code line results in the standard print method for an object of that class:

```
opt
```

```
## **********************************
## PortfolioAnalytics Optimization
## **********************************
##
## Call:
## optimize.portfolio(R = returns10, portfolio = pspec, optimize_method = "CVXR")
##
## Optimal Weights:
##    BGG    BIG    BMI    BMS   BOBE    BRC    CAL   CASY   CATO    CBB
## 0.0000 0.0284 0.0000 0.7517 0.0000 0.0083 0.0000 0.1940 0.0000 0.0175
##
## Objective Measures:
##  StdDev
## 0.03474
```

The above display is not generally very useful since it takes up a lot of space, and does not provide the portfolio mean return and Sharpe ratio. Instead, you can use the convenience function `opt.outputMvo` which extracts the MVO portfolio weights and standard deviation, and computes the portfolio mean return and Sharpe ratio, and returns those values in a list. We use it here just to compactly display the GmvLO portfolio weights and its Sharpe ratio.

```
outCovClassic <- opt.outputMvo(opt, returns10, digits = 3, frequency = "weekly")
(WtsCovClassic <- outCovClassic$Wgts)
```

```
##   BGG   BIG   BMI   BMS  BOBE   BRC   CAL  CASY  CATO   CBB
## 0.000 0.028 0.000 0.752 0.000 0.008 0.000 0.194 0.000 0.018
```

We will provide the same kind of displays as above for the robust MVO portfolios in the next Section.

3 Robust GmvLO Portfolios

We refer to an MVO portfolios that use a robust covariance matrix estimator as a *Robust MVO Portfolio*. In this section we display the four custom functions for estimating a robust covariance matrix, and show how to use them in the `optimize.portfolio` function via its optional argument `momentFUN =`, to compute a robust GmvLO portfolio. Although the focus here is on GmvLO portfolios, the robust covariance matrix functions can be used in other types of constrained MVO portfolios.

3.1 Custom covRob.MM Estimator

The first method to compute robust estimators is based on the RobStatTM package function `covRob`, which was designed to work well for portfolios with less than 10 assets. Details about this estimator are provided in (**Maronna2019?**).

Here is the function, for which additional details may be found using `help(custom.covRob.MM)`:

```
custom.covRob.MM <- function(R, ...){
  out <- list()
  if(hasArg(tol)) tol = match.call(expand.dots = TRUE)$tol else tol = 1e-4
  if(hasArg(maxit)) maxit = match.call(expand.dots = TRUE)$maxit else maxit = 50
  robustCov <- RobStatTM::covRobMM(X=R, tol=tol, maxit=maxit)
  out$sigma <- robustCov$cov
  out$mu <- robustCov$center
  return(out)
}
```

Here is how to use the function in `optimize.portfolios`, with different optional argument parameters than the default values, and store the resulting portfolio weights and Sharpe ratio for later comparisons:

```
opt <- optimize.portfolio(returns10, pspec, optimize_method = "CVXR",
                          momentFUN = "custom.covRob.MM",
                          maxit = 100, tol = 1e-5)
outCovRobMM <- opt.outputMvo(opt, returns10, digits = 3, frequency = "weekly")
(WtsCovRobMM <- outCovRobMM$Wgts)
```

```
##   BGG   BIG   BMI   BMS  BOBE   BRC   CAL  CASY  CATO   CBB
## 0.000 0.080 0.027 0.418 0.000 0.062 0.000 0.403 0.000 0.010
```

3.2 Custom covRob.Rocke Estimator

This custom estimator is based on the RobStatTM package function `covRobRocke`, which was designed to work well for portfolios with 10 or more assets. Details about this estimator are provided in (**Maronna2019?**). Here is the function, for which additional details may be found using `help(customr.covRob.Rocke)`:

```
custom.covRob.Rocke <- function(R, ...){
  out <- list()
  if(hasArg(tol)) tol = match.call(expand.dots = TRUE)$tol else tol = 1e-4
  if(hasArg(maxit)) maxit = match.call(expand.dots = TRUE)$maxit else maxit = 50
  if(hasArg(initial)) initial = match.call(expand.dots = TRUE)$initial else initial = 'K'
  if(hasArg(maxsteps)) maxsteps = match.call(expand.dots = TRUE)$maxsteps else maxsteps = 5
  if(hasArg(propmin)) propmin = match.call(expand.dots = TRUE)$propmin else propmin = 2
  if(hasArg(qs)) qs = match.call(expand.dots = TRUE)$qs else qs = 50

  robustCov <- RobStatTM::covRobRocke(X = R, initial = initial,
                                      maxsteps = maxsteps, ropmin = propmin,
                                      qs = qs, tol = tol, maxit = maxit)

  out$sigma <- robustCov$cov
  out$mu <- robustCov$center
  return(out)
}
```

Here is how to use the function `custom.covRobRocke` in `optimize.portfolio`, with different optional argument parameters than the default values, and store the resulting portfolio weights and Sharpe ratio for later comparisons:

```
opt <- optimize.portfolio(returns10, pspec, optimize_method = "CVXR",
                          momentFUN = "custom.covRob.Rocke",
                          tol = 1e-5, maxit =100, maxsteps = 7)
outCovRobRocke <- opt.outputMvo(opt, returns10, digits = 3, frequency = "weekly")
(WtsCovRobRocke <- outCovRobRocke$Wgts)
```

```
##   BGG   BIG   BMI   BMS  BOBE   BRC   CAL  CASY  CATO   CBB
## 0.000 0.086 0.027 0.437 0.000 0.051 0.000 0.389 0.000 0.010
```

3.3 Custom covRob.Mcd Estimator

This custom estimator is based on the *robustbase* package function `covMcd`, which is a highly developed and well-documented robust covariance matrix estimator. The *Mcd* in `covMcd` stands for *Minimum Covariance Determinant* (MCD). This robust covariance matrix estimator uses a sophisticated fast algorithm to search for $\boldsymbol{h}$ observations out of $\boldsymbol{n}$ whose classical covariance matrix has the smallest possible determinant. See (**RousseeuwDriessen1999?**). Then the raw MCD estimates of location and scatter are computed based on the sample mean vector and sample covariance matrix of those $\boldsymbol{h}$ points. Then covMcd uses a "refinement step" to compute the final robust covariance matrix and mean vector. For more details concerning the covMcd function and its parameters, and its "fast" computation, please see the Reference Manual and the Vignette "covMcd-Generalizing the FastMCD" downloadable from the CRAN robustbase package at https://cran.r-project.org/web/packages/robustbase/index.html.

Here is the custom.covRob.Mcd function:

```
custom.covRob.Mcd <- function(R, ...){

  if(hasArg(control)) control = match.call(expand.dots = TRUE)$control else control = MycovRobMcd()
```

```
  if(hasArg(alpha)) alpha = match.call(expand.dots = TRUE)$alpha else alpha = control$alpha
  if(hasArg(nsamp)) nsamp = match.call(expand.dots = TRUE)$nsamp else nsamp = control$nsamp
  if(hasArg(nmini)) nmini = match.call(expand.dots = TRUE)$nmini else nmini = control$nmini
  if(hasArg(kmini)) kmini = match.call(expand.dots = TRUE)$kmini else kmini = control$kmini
  if(hasArg(scalefn)) scalefn=match.call(expand.dots=TRUE)$scalefn else scalefn=control$scalefn
  if(hasArg(maxcsteps)) maxcsteps = match.call(expand.dots = TRUE)$maxcsteps
  else maxcsteps = control$maxcsteps

  if(hasArg(initHsets)) initHsets = match.call(expand.dots = TRUE)$initHsets
  else initHsets = control$initHsets

  if(hasArg(seed)) seed=match.call(expand.dots=TRUE)$seed else seed=control$seed
  if(hasArg(tolSolve)) tolSolve=match.call(expand.dots=TRUE)$tolSolve else tolSolve=control$tolSolve
  if(hasArg(wgtFUN)) wgtFUN=match.call(expand.dots=TRUE)$wgtFUN else wgtFUN=control$wgtFUN

  if(hasArg(use.correction)) use.correction = match.call(expand.dots = TRUE)$use.correction
  else use.correction = control$use.correction

  robustMCD <- robustbase::covMcd(x = R, alpha = alpha,
                                  nsamp = nsamp, nmini = nmini,
                                  kmini = kmini, seed = seed,
                                  tolSolve = tolSolve, scalefn = scalefn,
                                  maxcsteps = maxcsteps,
                                  initHsets = initHsets,
                                  wgtFUN = wgtFUN, use.correction = use.correction)

  return(list(mu = robustMCD$center, sigma = robustMCD$cov))
}
```

The above function allows you to substitute any of the many default parameters with your alternative choices. For example, you can use the following code to compute a robust MVO portfolio with the custom.covRob.Mcd estimator, based on the choice of 0.75 for `alpha` and the choice 600 for `nsamp`.

```
opt <- optimize.portfolio(returns10, pspec, optimize_method = "CVXR",
                          momentFUN = "custom.covRob.Mcd",
                          alpha = 0.75, nsamp = 600)
outCovRobMcd <- opt.outputMvo(opt, returns10, digits = 3, frequency = "weekly")
(WtsCovRobMcd <- outCovRobMcd$Wgts)
```

```
##   BGG   BIG   BMI   BMS  BOBE   BRC   CAL  CASY  CATO   CBB
## 0.000 0.050 0.007 0.317 0.000 0.097 0.000 0.528 0.000 0.001
```

A challenge in using `custom.covRob.Mcd` is that it has a large set of optional argument parameter choices. In some studies one may want to use several or many different sets of such parameters, and in that case the following `MycovRobMcd` "helper" function can be useful.

```
MycovRobMcd <- function(alpha = 1/2, nsamp = 500, nmini = 300, kmini = 5,
                        scalefn = "hrv2012", maxcsteps = 200,
                        seed = NULL, tolSolve = 1e-14,
```

```
                        wgtFUN = "01.original", beta, use.correction = TRUE
){
  if(missing(beta) || !is.numeric(beta))
    beta <- 0.975

  return(list(alpha = alpha, nsamp = nsamp,
              nmini = as.integer(nmini), kmini = as.integer(kmini),
              seed = as.integer(seed),
              tolSolve = tolSolve, scalefn = scalefn,
              maxcsteps = as.integer(maxcsteps),
              wgtFUN = wgtFUN, beta = beta,
              use.correction = use.correction))
}
```

So for example, you could use `MycovMcd` to specify the choices 0.75 for `alpha` and 600 for `nsamp`, and then use it for the `optimize.portfolio` argument `control =` as follows.

```
covMcd.params <- MycovRobMcd(alpha = 0.75, nsamp = 600)
opt <- optimize.portfolio(returns, pspec, optimize_method = "CVXR",
                          momentFUN = "custom.covRob.Mcd",
                          control = covMcd.params)
```

3.4 Custom covRob.TSGS Estimator

The importance of the custom covRob.TSGS estimator is that it is designed to be able to deal with both of the following two distinct types of outliers. The first type of outliers are *Cross-Section Outliers* (CSO), which occur for most of the assets in portfolio at at the same time. For example, in 1987, a global severe stock market crash, known as Black Monday, drastically reduced the returns of most of the assets, which can thus be viewed as a CSO's. The second type of outliers are *Independent Outliers in Assets* (IOA), which occur independently, or uncorrelated, across assets. These types of outliers are sometimes referred to as "cell-wise contamination" in the robust statistic literature. Details concerning these two types of outliers, and robust methods for dealing with them in MVO portfolio optimization are discussed in Martin (2013).

The acronym TSGS stands for *2-step Generalized S-estimators* and this `TSGS` robust covariance matrix estimator function is contained in the package *GSE* which is designed to robustly estimate a covariance matrix with NA's. The first step of `TSGS` is to filter out the large IOA outliers and replace them with NA's. The second step is to deal with the NA's while down-weighting high-dimensional CSO outliers that are not detected in step 1. For details see Agostinelli (2015).

Here is the `custom.covRob.TSGS function`:

```
custom.covRob.TSGS <- function(R, ...){
 if(hasArg(control)) control=match.call(expand.dots=TRUE)$control else control=MycovRobTSGS()
 if(hasArg(filter)) filter=match.call(expand.dots=TRUE)$filter else filter=control$filter

 if(hasArg(partial.impute)) partial.impute = match.call(expand.dots = TRUE)$partial.impute
 else partial.impute = control$partial.impute

 if(hasArg(tol)) tol = match.call(expand.dots=TRUE)$tol else tol=control$tol
```

```
 if(hasArg(maxiter)) maxiter=match.call(expand.dots=TRUE)$maxiter else maxiter=control$maxiter
 if(hasArg(loss)) loss = match.call(expand.dots = TRUE)$loss else loss = control$loss
 if(hasArg(init)) init = match.call(expand.dots = TRUE)$init else init = control$init

  tsgsRob <- GSE::TSGS(x = R, filter = filter,
                       partial.impute = partial.impute, tol = tol,
                       maxiter = maxiter, method = loss,
                       init = init)

  return(list(mu = tsgsRob@mu, sigma = tsgsRob@S))

}
```

```
opt <- optimize.portfolio(returns10, pspec, optimize_method = "CVXR",
                          momentFUN = "custom.covRob.TSGS")
outCovRobTSGS <- opt.outputMvo(opt, returns10, digits = 3, frequency = "weekly")
(WtsCovRobTSGS <- outCovRobTSGS$Wgts)
```

```
##   BGG   BIG   BMI   BMS  BOBE   BRC   CAL  CASY  CATO   CBB
## 0.000 0.047 0.023 0.388 0.000 0.019 0.000 0.523 0.000 0.000
```

Just as in the case of the `custom.covRob.Mcd` function, for which there is the helper function `MycovRobMcd`, some users wish to use the following auxiliary function that helps with TSGS parameter setting.

```
MycovRobTSGS <- function(filter = c("UBF-DDC","UBF","DDC","UF"),
                partial.impute = FALSE, tol = 1e-4, maxiter = 150,
                loss = c("bisquare","rocke"),
                init = c("emve", "qc", "huber", "imputed", "emve_c")){

  filter <- match.arg(filter)
  loss <- match.arg(loss)
  init <- match.arg(init)

  return(list(filter = filter, partial.impute = partial.impute,
              tol = tol, maxiter = as.integer(maxiter),
              loss = loss,init))
}
```

3.5 Comparison of the Classic and Robust GmvLO Portfolios

The following table compares the portfolio weights of the covClassic, covRobMM, covRobRocke, covRobMcd, and covRobTSGS robust GmvLO portfolios. All five of these portfolios are highly concentrated in the two stocks with tickers BMS and CASY.

```
dat <- data.frame(rbind(WtsCovClassic, WtsCovRobMM, WtsCovRobRocke,
                        WtsCovRobMcd, WtsCovRobTSGS))
print.data.frame(dat)
```

```
##                 BGG   BIG   BMI   BMS BOBE   BRC CAL  CASY CATO   CBB
## WtsCovClassic     0 0.028 0.000 0.752    0 0.008   0 0.194    0 0.018
## WtsCovRobMM       0 0.080 0.027 0.418    0 0.062   0 0.403    0 0.010
## WtsCovRobRocke    0 0.086 0.027 0.437    0 0.051   0 0.389    0 0.010
## WtsCovRobMcd      0 0.050 0.007 0.317    0 0.097   0 0.528    0 0.001
## WtsCovRobTSGS     0 0.047 0.023 0.388    0 0.019   0 0.523    0 0.000
```

The large differences in WtsCovClassic weights values for the stocks BMS and CASY, relative to weights values for the four robust covariance, is striking. In view of the time series plots of the BMS and CASY returns in Figure 3, it appears that these differences are due to the influence of returns outliers on classic sample covariance matrix.

The following table compares the classic and robust GmvLO portfolios Annualized Mean Returns, Standard Deviations and Sharpe Ratios.

```
dat.mat <- rbind(outCovClassic[2:4], outCovRobMM[2:4],  outCovRobRocke[2:4],
                 outCovRobMcd[2:4], outCovRobTSGS[2:4])
dat <- as.data.frame(dat.mat)
row.names(dat) <- c("GmvLOcovClassic", "GmvLOcovRobMM", "GmvLOcovRobRocke",
                    "GmvLOcovRobMcd", "GmvLOcovRobTSGS")
print.data.frame(dat)
```

```
##                    Mean StdDev    SR
## GmvLOcovClassic   0.107   0.25 0.428
## GmvLOcovRobMM     0.132  0.266 0.494
## GmvLOcovRobRocke  0.131  0.265 0.495
## GmvLOcovRobMcd    0.145  0.279 0.518
## GmvLOcovRobTSGS   0.148  0.277 0.536
```

The covClassic based portfolio has the smallest SR (Sharpe Ratio), with noticeably larger and nearly equal SR's for the covRobMM and covRobRocke based GmvLO portfolios, and increasingly larger SR values for the covRobMcd and covRobTSGS based GmvLO portfolios. A perusal of the Mean and StdDev columns of the table show that while the StdDev values increase slowly going down the column, the Mean values increase more rapidly, which results in the increase in values going down the SR column.

4 Robust GmvLO Portfolio Backtests

In this section we conduct backtests of the classic GvmLO portfolio, and the three robust GMvLO portfolios based on the covRobRocke, covRobMcd and covRobTSGS robust covariance matrix estimators, using the function `optimize.portfolio.rebalancing` in PortfolioAnalytics. The training period for the backtests is 100 weeks, and the portfolio is rebalanced weekly. To show the backtest results for different robust covariance matrix estimators in a simple way, we created the function shown below to compute and plot the results. The function has a parameter called `plot` to control which robust estimator is used in the backtest: 1 is for the default choice `covRob.Rocke`, 2 is for the choice`covRob.Mcd`, and 3 is for `covRob.TSGS`.

The computing times of the covClassic, covRobRocke, covRobMcd and covRobTSGS robust portfolio backtests, on a DELL XPS 17 with 6 cores, up to 5.0 GHz, 12MB cache were: 46 seconds, 2.0 minutes, 55 seconds, and 2.4 mintues, respectively.

```
# Plot function
robPlot <- function(GMV, MARKET, plot=1){
  # Optimize Portfolio at Monthly Rebalancing and 5-Year Training
  if(plot == 1){
    momentEstFun = 'custom.covRob.Rocke'
    name = "GmvLOCovRobRocke"
  }else if(plot == 2){
    momentEstFun = 'custom.covRob.Mcd'
    name = "GmvLOCovMcd"
  }else if(plot == 3){
    momentEstFun = 'custom.covRob.TSGS'
    name = "GmvLOTSGS"
  }else{
    print("plot should be 1, 2 or 3")
    return()
  }

  bt.gmv.rob <- optimize.portfolio.rebalancing(returns, pspec,
                                               optimize_method = "CVXR",
                                               rebalance_on = "weeks",
                                               training_period = 100,
                                               momentFUN = momentEstFun)

  # Extract time series of portfolio weights
  wts.gmv.rob <- extractWeights(bt.gmv.rob)
  # Compute cumulative returns of portfolio
  GMV.rob <- Return.rebalancing(returns, wts.gmv.rob)

  # Combine GMV.LO and MARKET cumulative returns
  ret.comb <- na.omit(merge(GMV.rob, GMV, MARKET, all=F))
  names(ret.comb) <- c(name, "GmvLO", "MARKET")

  plot <- backtest.plot(ret.comb, colorSet = c("darkgreen", "black",
                        "red"), ltySet = c(1,2,3))

  return(list(ret = ret.comb, plot = plot))
}
```

4.1 covRob.Rocke GmvLO Backtest

We use default settings of the `custom.covRob.Rocke` function for this backtest. From the plot, we can see that during year 2008, when the well-known finance crisis occurred, the classic GmvLO portfolio with the standard sample covariance matrix estimator, and the `custom.covRob.Rocke` MVO portfolio have similar performances based on weekly returns for the prior two years. However,the robust portfolio based on `custom.covRob.Rocke` outperforms the one based on the standard sample covariance matrix after the crisis with regard to both gross cumulative return and the worst drawdown. This is due to the fact that the robust covariance matrix down-weights influential outliers, rejecting sufficiently large outliers, which prevents to robust MVO portfolio from being adversely influenced by the outliers which

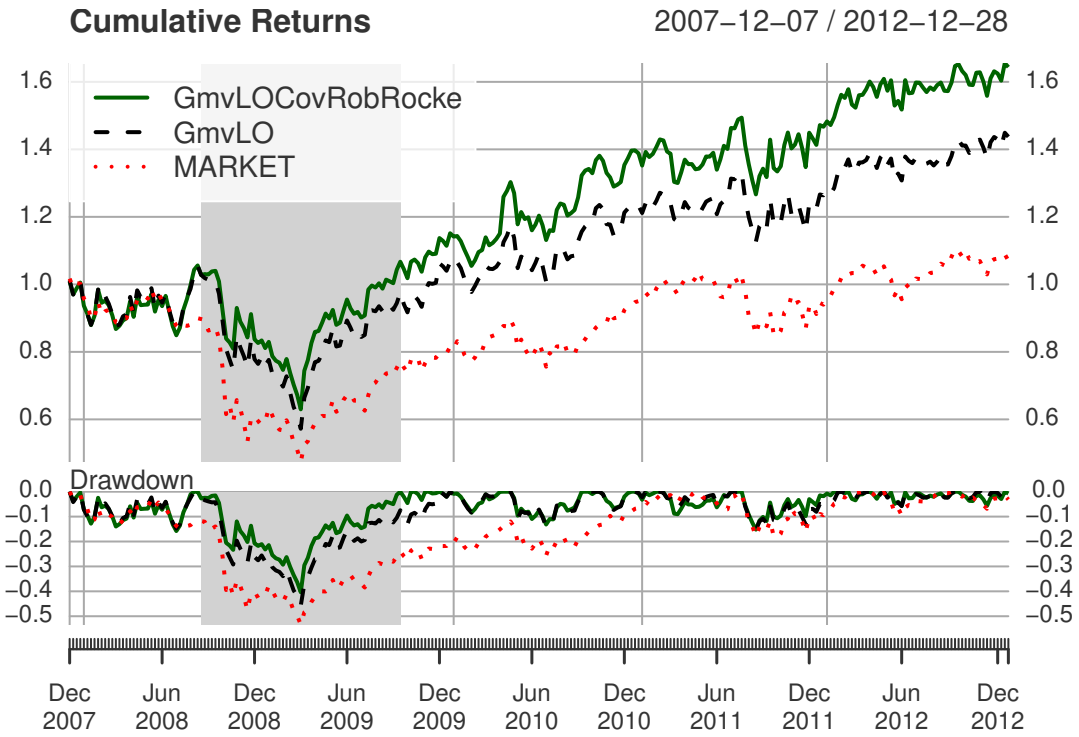
occur during the financial crisis.

```
bt.gmv <- optimize.portfolio.rebalancing(returns, pspec,
                                         optimize_method = "CVXR",
                                         rebalance_on="weeks",
                                         training_period = 100)
```

```
## Warning: executing %dopar% sequentially: no parallel backend registered
```

```
# Extract time series of portfolio weights
wts.gmv <- extractWeights(bt.gmv)
# Compute cumulative returns of portfolio
GMV <- Return.rebalancing(returns, wts.gmv)

res.covRob <- robPlot(GMV=GMV, MARKET=MARKET, plot=1)
res.covRob$plot
```



The shaded area in the above plot indicates the longest drawdown of the `covRob.Rocke GmvLO portfolio, which outperforms the covClassic GmvLO portfolio optimization in this longest drawdown. The following` table.Drawdowns`function, with optional argument` top =1‘, computes the statistics of the longest drawdown of the covRob.Rocke GmvLO portfolio, and in particular shows that the length of this drawdown is 57 weeks.

```
table.Drawdowns(res.covRob$ret$GmvLOCovRobRocke, top=1)
```

```
##         From     Trough         To   Depth Length To Trough Recovery
## 1 2008-08-22 2009-03-06 2009-09-18 -0.4038     57        29       28
```

By way of comparison, the following use of `table.Drawdowns` computes the statistics of the longest drawdown of the covClassic GmvLO portfolio, whose drawdown length is the much longer 68 weeks.

```
table.Drawdowns(res.covRob$ret$GmvLO, top=1)
```

```
##         From     Trough         To   Depth Length To Trough Recovery
## 1 2008-08-22 2009-03-06 2009-12-04 -0.4576     68        29       39
```

Of course the greater than 3 years 176 week drawdown of the MARKET, computed below, is vastly worse than that of both the covClassic and covRobRocke GmvLO portfolios.

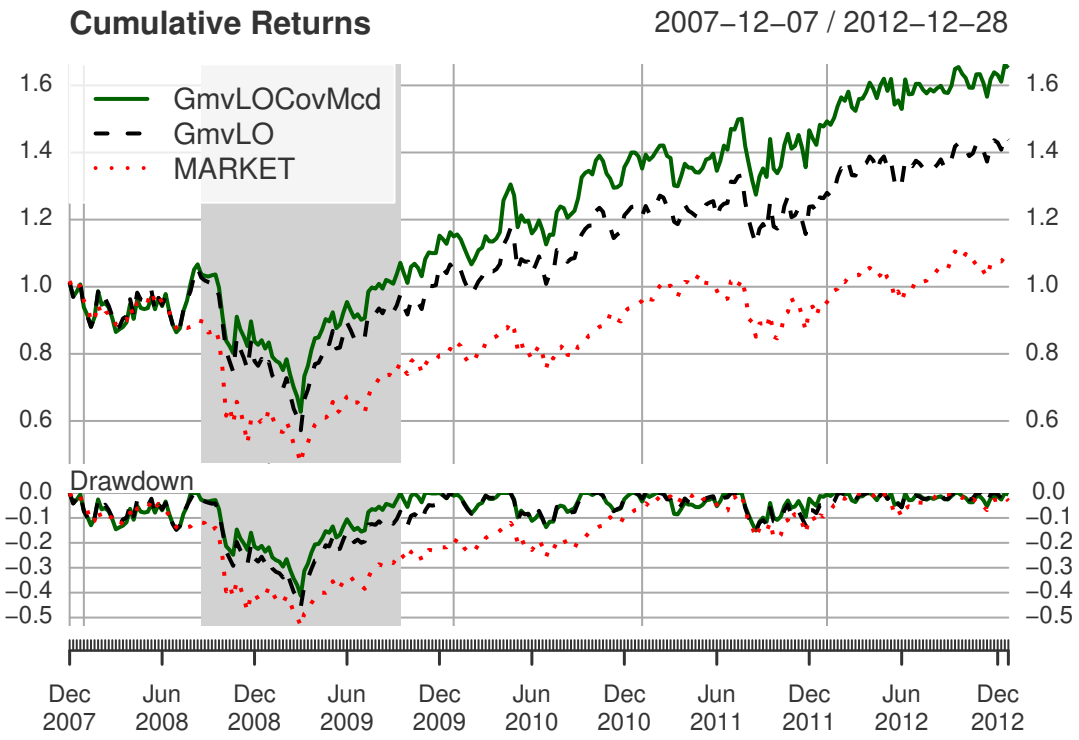
```
table.Drawdowns(res.covRob$ret$MARKET, top=1)
```

```
##         From     Trough         To   Depth Length To Trough Recovery
## 1 2007-12-14 2009-03-06 2011-04-22 -0.5336    176        65      111
```

4.2 covRob.Mcd GmvLO Backtest

We use `alpha = 0.5` for this estimator, which results in a breakdown point of close to, but slightly less than 0.5. From the plots below, we see that the resulting robust MVO portfolio performs better than the robust MVO portfolio based on `covRob.Rocke`.

```
set.seed(1234)
res.covMcd = robPlot(GMV=GMV, MARKET=MARKET, plot=2)
res.covMcd$plot
```



In this case, use of the `table.Drawdowns` function shows that the longest covRobMcd GmvLO portfolio has the same 57 week duration as the covRobRocke GmvLO portfolio.

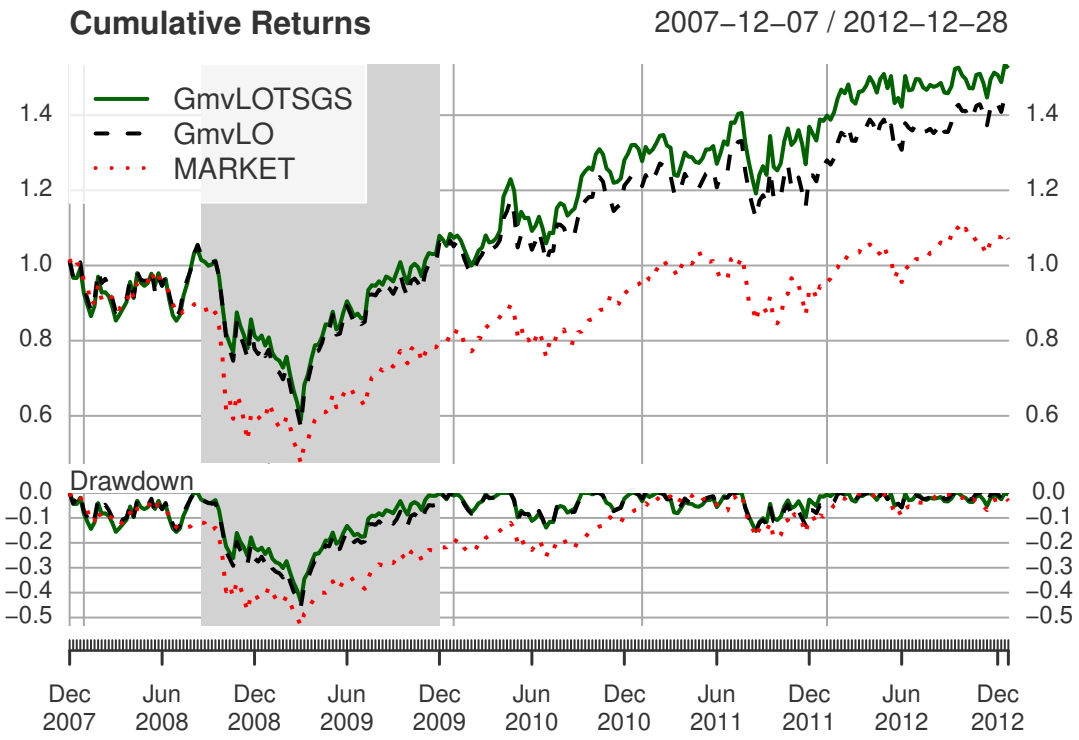
```
# longest drawdown for robust based portfolio
table.Drawdowns(res.covMcd$ret$GmvLOCovMcd, top=1)
```

```
##          From     Trough         To   Depth Length To Trough Recovery
## 1 2008-08-22 2009-03-06 2009-09-18 -0.4121     57        29       28
```

4.3 covRob.TSGS GmvLO

For `custom.covRob.TSGS`, we use the default settings Similar to *covRob*, optimized portfolios with standard sample covariance matrix and *TSGS* estimators have similar performance before the end of 2008. Furthermore, the *TSGS* based portfolio shows a better cumulative return after year 2008.

```
res.TSGS = robPlot(GMV=GMV, MARKET=MARKET, plot=3)
res.TSGS$plot
```



The code line below reveals that longest drawdown for `covRob.TSGS` based portfolio lasts 68 weeks, rather worse than for the covRobRocke and covRobMcd GmvLO portfolios.

```
# longest drawdown for robust based portfolio
table.Drawdowns(res.TSGS$ret$GmvLOTSGS, top=1)
```

```
##          From     Trough         To   Depth Length To Trough Recovery
## 1 2008-08-22 2009-03-06 2009-12-04 -0.4352     68        29       39
```

5 Writing Your Own Custom Covariance Matrix Function

There are any one of the following reasons, among others, why you may want to write your own custom covariance matrix function:

1. You want to frequently compute a variety of constrained MVO portfolios using one of the 4 types of robust covariance matrices discussed in Section 3, for a particular set of non-default optional argument values.
2. You want to use a robust covariance matrix for MVO portfolio construction that is not one of the above 4 types, but is available in some R package.
3. You want to use a shrinkage covariance method that is available in another R package, or in some R function.

The key for doing so is that the return of the function must be a list containing the mean vector object named `mu` and and the covariance matrix object named `sigma`. The following two sections provide simple examples of the first two cases above.

5.1 A User Custom Function for the covRobRocke Estimator

Here is the function using non-default values for `maxit =` and `maxsteps =`:

```
user.covRob.Rocke <- function(R){
  out <- list()
  robustCov <- RobStatTM::covRobRocke(X = R, maxit = 200, maxsteps = 10)
  out$sigma <- robustCov$cov
  out$mu <- robustCov$center
  return(out)
}
```

Then the function `user.covRob.Rocke` would be used for example in `portfolio.optimize` as follows (try it out):

```
opt <- optimize.portfolio(returns, pspec,
                   optimize_method = "CVXR",
                   momentFUN = "user.covRob.Rocke")
```

5.2 A User Custom Function for covRobOGK in `robustbase`

The user function `covRobOGK` presented below computes portfolio assets pairwise robust covariance matrix estimates. This method was originally introduced by (**GnanadesikanKettenring1972?**), and improved with an "orthogonalizing step" by (**MaronnaZamar2002?**). For details, see the `covRobOGK` page in the robustbase Reference Manual at https://cran.r-project.org/web/packages/robustbase/index.html. Here is the user function based on the robustbase function `covOGK` with default argument values:

```
user.covRob.OGK <- function(R){
  robustCov <- robustbase::covOGK(x = R)
  return(list(mu = robustCov$center, sigma = robustCov$cov))
}
```

Then the function `user.covRob.OGK` would be used for example in `portfolio.optimize` as follows (try it out):

```
opt <- optimize.portfolio(returns, pspec, optimize_method = "CVXR",
                          momentFUN = "user.covMcd.OGK")
```

5.3 Print the Value of Moment Settings

This is how you see the numerical values of the mean vector `mu` and the covariance matrix `sigma`. Even if you don't provide any `momentFUN=`, this command will print out the default moment setting values for you.

```
opt$moment_values
```

```
## $momentFun
## [1] "custom.covRob.TSGS"
##
## $mu
##  [1] -0.0005193308  0.0014613728 -0.0035289253  0.0007576777 -0.0033464762
##  [6]  0.0009911580 -0.0011553683  0.0008238260  0.0014597271 -0.0017177511
##
## $sigma
##               [,1]         [,2]         [,3]         [,4]         [,5]
##  [1,] 0.0040763786 0.0013090397 0.0017309318 0.0011847546 0.0018318186
##  [2,] 0.0013090397 0.0021727762 0.0007747888 0.0006712464 0.0010882947
##  [3,] 0.0017309318 0.0007747888 0.0020800312 0.0007578735 0.0012178132
##  [4,] 0.0011847546 0.0006712464 0.0007578735 0.0009619914 0.0008591836
##  [5,] 0.0018318186 0.0010882947 0.0012178132 0.0008591836 0.0017978093
##  [6,] 0.0019243653 0.0008294012 0.0011930676 0.0007976439 0.0011716479
##  [7,] 0.0031897096 0.0018021058 0.0018812388 0.0014177725 0.0021712513
##  [8,] 0.0009314466 0.0004238248 0.0004131787 0.0003445779 0.0005588212
##  [9,] 0.0018996619 0.0010464272 0.0011667203 0.0007917387 0.0011848044
## [10,] 0.0021299916 0.0008284009 0.0014139384 0.0007589050 0.0013154664
##               [,6]         [,7]         [,8]         [,9]        [,10]
##  [1,] 0.0019243653 0.0031897096 0.0009314466 0.0018996619 0.0021299916
##  [2,] 0.0008294012 0.0018021058 0.0004238248 0.0010464272 0.0008284009
##  [3,] 0.0011930676 0.0018812388 0.0004131787 0.0011667203 0.0014139384
##  [4,] 0.0007976439 0.0014177725 0.0003445779 0.0007917387 0.0007589050
##  [5,] 0.0011716479 0.0021712513 0.0005588212 0.0011848044 0.0013154664
##  [6,] 0.0022411980 0.0022244449 0.0003809158 0.0012585869 0.0013665916
##  [7,] 0.0022244449 0.0059335616 0.0009693145 0.0022467678 0.0017440089
##  [8,] 0.0003809158 0.0009693145 0.0008543004 0.0005848772 0.0005015430
##  [9,] 0.0012585869 0.0022467678 0.0005848772 0.0021317879 0.0012153081
## [10,] 0.0013665916 0.0017440089 0.0005015430 0.0012153081 0.0035770974
```

Acknowledgements

The PortfolioAnalytics custom robust covariance matrix estimators R code discussed in Section 3, along with some other code in this Vignette, was written by Yifu Kang while he was a student in the University of Washington Department of Applied Mathematics Computational Finance and Risk Management M.S. degree program (MS-CFRM). Yifu's work reported here was funded by a 2022 Google Summer of Code (GSoC 2022) project. University of Washington Applied Mathematics faculty members Doug Martin and Steve Murray were Kang's Mentors for this project.

Steve Murray discovered the existence of the Custom Moments functionality of PortfolioAnalytics, and suggested that Yifu pursue implementation of robust covariance matrix estimators in PortfolioAnalytics with that functionality.

Another MS-CFRM student, Xinran Zhao, who was funded by a different PortfolioAnalytics GSoC 2022 project, assisted Yifu on PortfolioAnalytics implementation of his custom robust covariance matrix functions.

Finally, this project could not have been completed without the help of Brian Peterson, who is an Author and the Maintainer of the PortfolioAnalytics package on CRAN.

References

Agostinelli, Yohai, Claudio. 2015. "Robust Estimation of Multivariate Location and Scatter in the Presence of Cellwise and Casewise Contamination." *Test: An Official Journal of the Spanish Society of Statistics and Operations Research.*

Martin, R. Douglas. 2013. "Robust Covariances: Common Risk Versus Specific Risk Outliers." R-finance Conference 2013.