Custom Moment and Objective Functions

Ross Bennett

August 3, 2014

Abstract

The purpose of this vignette is to demonstrate how to write and use custom moment functions and custom objective functions.

Contents

1 Getting Started

1.1 Load Packages

Load the necessary packages.

```
library(PortfolioAnalytics)

## Loading required package:  zoo
##
## Attaching package:  'zoo'
##
## The following objects are masked from 'package:base':
```

```
##
##    as.Date, as.Date.numeric
##
## Loading required package:  xts
## Loading required package:  PerformanceAnalytics
##
## Attaching package:  'PerformanceAnalytics'
##
## The following object is masked from 'package:graphics':
##
##    legend
```

1.2 Data

The edhec data set from the PerformanceAnalytics package will be used as example data.

```
data(edhec)


# Use the first 4 columns in edhec for a returns object
R <- edhec[, 1:4]
colnames(R) <- c("CA", "CTAG", "DS", "EM")
head(R, 5)

##                 CA    CTAG      DS      EM
## 1997-01-31 0.0119  0.0393  0.0178  0.0791
## 1997-02-28 0.0123  0.0298  0.0122  0.0525
## 1997-03-31 0.0078 -0.0021 -0.0012 -0.0120
## 1997-04-30 0.0086 -0.0170  0.0030  0.0119
## 1997-05-31 0.0156 -0.0015  0.0233  0.0315

# Get a character vector of the fund names
funds <- colnames(R)
```

2 Setting the Portfolio Moments

The PortfolioAnalytics framework to estimate solutions to constrained optimization problems is implemented in such a way that the moments of the returns are calculated only once. The `set.portfolio.moments` function computes the first, second, third, and fourth moments depending on the objective function(s) in the `portfolio` object. The moments are then used by lower level optimization functions. `set.portfolio.moments` implements methods to compute moments based on sample estimates, higher moments from fitting a statistical factor model based on the work of Kris Boudt (NEED REFERENCE HERE), the Black Litterman model, and the Fully Flexible Framework based on the work of Attilio Meucci.

The moments of the returns are computed based on the objective(s) in the `portfolio` object and return a list where each element is the respective moment estimate.

```
args(set.portfolio.moments)

## function (R, portfolio, momentargs = NULL, method = c("sample",
##     "boudt", "black_litterman", "meucci"), ...)
## NULL
```

```
# Construct initial portfolio with basic constraints.
init.portf <- portfolio.spec(assets=funds)
init.portf <- add.constraint(portfolio=init.portf, type="full_investment")
init.portf <- add.constraint(portfolio=init.portf, type="long_only")

# Portfolio with standard deviation as an objective
SD.portf <- add.objective(portfolio=init.portf, type="risk", name="StdDev")

# Portfolio with expected shortfall as an objective
ES.portf <- add.objective(portfolio=init.portf, type="risk", name="ES")
```

Here we see the names of the object that is returned.

```
sd.moments <- set.portfolio.moments(R, SD.portf)
names(sd.moments)

## [1] "mu"    "sigma"

es.moments <- set.portfolio.moments(R, ES.portf)
```

```
names(es.moments)

## [1] "mu"    "sigma" "m3"    "m4"
```

3 Custom Moment Functions

In many cases for constrained optimization problems, one may want to estimate moments for a specific use case or further extend the idea of `set.portfolio.moments`. A user defined custom moment function can have any arbitrary named arguments, however the argument names `R` and `portfolio` will be detected and matched in an efficient manner.

Here we define a function to compute the covariance matrix using a robust estimate.

```
sigma.robust <- function(R, ...){
  out <- list()
  set.seed(1234)
  out$sigma <- MASS::cov.rob(R, method="mcd", ...)$cov
  return(out)
}
```

Now we can use the custom moment function in `optimize.portfolio` to estimate the solution to the minimum standard deviation portfolio.

```
opt.sd <- optimize.portfolio(R, SD.portf,
                             optimize_method="ROI",
                             momentFUN="sigma.robust")

## ROI: R Optimization Infrastructure
## Registered solver plugins:  cplex, glpk, quadprog, symphony.
## Default solver:  ROI_NULL.
##
## Attaching package:  'ROI'
##
## The following object is masked from 'package:PortfolioAnalytics':
##
##    objective

opt.sd
```

```
## **********************************
## PortfolioAnalytics Optimization
## **********************************
##
## Call:
## optimize.portfolio(R = R, portfolio = SD.portf, optimize_method = "ROI",
##     momentFUN = "sigma.robust")
##
## Optimal Weights:
##     CA   CTAG     DS     EM
## 0.6598 0.1441 0.1961 0.0000
##
## Objective Measure:
##   StdDev
## 0.008646
```

Here we extract the weights and compute the portfolio standard deviation to verify.

```
weights <- extractWeights(opt.sd)
sigma <- sigma.robust(R)$sigma


sqrt(t(weights) %*% sigma %*% weights)

##           [,1]
## [1,] 0.008646

extractObjectiveMeasures(opt.sd)$StdDev

##   StdDev
## 0.008646
```

4 Custom Objective Functions

A key feature of `PortfolioAnalytics` is that the name for an objective can be any valid Rfunction. `PortfolioAnalytics` was designed to be flexible and modular, and custom objective functions are a key example of this.

TODO: add content and example code