Vignette: Portfolio Optimization with CVaR budgets in PortfolioAnalytics

Kris Boudt, Peter Carl and Brian Peterson

June 1, 2010

Contents

1 General information **1**

2 Setting of the objective function **4**

2.1 Weight constraints . . . 4

2.2 Minimum CVaR objective function . . . 5

2.3 Minimum CVaR concentration objective function . . . 7

2.4 Risk allocation constraints . . . 8

3 Optimization **9**

3.1 Minimum CVaR portfolio under an upper 40% CVaR allocation constraint . . . 9

3.2 Minimum CVaR concentration portfolio . . . 12

3.3 Dynamic optimization . . . 13

1 General information

Risk budgets are a central tool to estimate and manage the portfolio risk allocation. They decompose total portfolio risk into the risk contribution of each position. Boudt et al. (2010) propose several portfolio allocation strategies that use an appropriate transformation of the portfolio Conditional Value at Risk (CVaR) budget as an objective or constraint in the portfolio optimization problem. This document explains how risk allocation optimized portfolios can be obtained under general constraints in the `PortfolioAnalytics` package of Boudt et al. (2012).

`PortfolioAnalytics` is designed to provide numerical solutions for portfolio problems with complex constraints and objective sets comprised of any R function. It can e.g. construct portfolios that minimize a risk objective with (possibly non-linear) per-asset constraints on returns and drawdowns (Carl et al., 2010). The generality of possible constraints and objectives is a distinctive characteristic of the package with respect to RMetrics `fPortfolio` of Wuertz et al. (2010). For standard Markowitz optimization problems, use of `fPortfolio` rather than `PortfolioAnalytics` is recommended.

`PortfolioAnalytics` solves the following type of problem

$$\min_{w} g(w) \quad s.t. \quad \begin{cases} h_1(w) \leq 0 \\ \vdots \\ h_q(w) \leq 0. \end{cases} \tag{1}$$

`PortfolioAnalytics` first merges the objective function and constraints into a penalty augmented objective function

$$L(w) = g(w) + \text{penalty} \sum_{i=1}^{q} \lambda_i \max(h_i(w), 0), \tag{2}$$

where λ_i is a multiplier to tune the relative importance of the constraints. The default values of penalty and λ_i (called `multiplier` in `PortfolioAnalytics`) are 10000 and 1, respectively.

The minimum of this function is found through the *Differential Evolution* (DE) algorithm of Storn and Price (1997) and ported to R by Mullen et al. (2009). DE is known for remarkable performance regarding continuous numerical problems (Price et al., 2006). It has recently been advocated for optimizing portfolios under non-convex settings by Ardia et al. (2010) and Yollin (2009), among others. We use the R implementation of DE in the `DEoptim` package of Ardia and Mullen (2009).

The latest version of the `PortfolioAnalytics` package can be downloaded from R-forge through the following command:

```
install.packages("PortfolioAnalytics", repos="http://R-Forge.R-project.org")
```

Its principal functions are:

- `portfolio.spec(assets)`: the portfolio specification starts with creating a `portfolio` object with information about the assets. The first argument `assets` is either a number indicating the number of portfolio assets or a vector holding the names of the assets. The `portfolio` object is a list holding the constraints and objectives.

- `add.constraint(portfolio, type)`: Constraints are added to the `portfolio` object by the function `add.constraint`. Basic constraint types include leverage constraints that specify the sum of the weights have to be between `min_sum` and `max_sum` and box constraints where the asset weights have to be between `min` and `max`.

- `add.objective(portfolio, type, name)`: New objectives are added to the `portfolio` objected with the function `add.objective`. Many common risk budget objectives and constraints are prespecified and can be identified by specifying the `type` and `name`.

- `constrained_objective(w, R, portfolio)`: given the portfolio weight and return data, it evaluates the penalty augmented objective function in (2).

- `optimize.portfolio(R, portfolio)`: this function returns the portfolio weight that solves the problem in (1). R is the multivariate return series of the portfolio components.

- `optimize.portfolio.rebalancing(R, portfolio, rebalance_on, trailing_periods)`: this function solves the multiperiod optimization problem. It returns for each rebalancing period the optimal weights and allows the estimation sample to be either from inception or a moving window.

Next we illustrate these functions on monthly return data for bond, US equity, international equity and commodity indices, which are the first 4 series in the dataset `indexes`. The first step is to load the package `PortfolioAnalytics` and the dataset. An important first note is that some of the functions (especially `optimize.portfolio.rebalancing`) requires the dataset to be a `xts` object (Ryan and Ulrich, 2010).

```
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

data
class




head





tail


```

In what follows, we first illustrate the construction of the penalty augmented objective function. Then we present the code for solving the optimization problem.

2 Setting of the objective function

2.1 Weight constraints

```
# Create the portfolio specification object
Wcons <- portfolio.spec( assets = colnames(indexes) )
# Add box constraints
Wcons <- add.constraint( portfolio=Wcons, type='box', min = 0, max=1 )
# Add the full investment constraint that specifies the weights must sum to 1.
Wcons <- add.constraint( portfolio=Wcons, type="full_investment")
```

Given the weight constraints, we can call the value of the function to be minimized. We consider the case of no violation and a case of violation. By default, `normalize=TRUE` which means that if the sum of weights exceeds `max_sum`, the weight vector is normalized by multiplying it with `sum(weights)/max_sum` such that the weights evaluated in the objective function satisfy the `max_sum` constraint.

```
constrained_objective( w = rep(1/4,4) , R = indexes, portfolio = Wcons)

## [1] 0

constrained_objective( w = rep(1/3,4) , R = indexes, portfolio = Wcons)

## [1] 0

constrained_objective( w = rep(1/3,4) , R = indexes, portfolio = Wcons,
                       normalize=FALSE)

## [1] 3333
```

The latter value can be recalculated as penalty times the weight violation, that is: $10000 \times 1/3$.

2.2 Minimum CVaR objective function

Suppose now we want to find the portfolio that minimizes the 95% portfolio CVaR subject to the weight constraints listed above.

```
ObjSpec = add.objective( portfolio = Wcons , type="risk",name="CVaR",
                         arguments=list(p=0.95), enabled=TRUE)
```

The value of the objective function is:

```
constrained_objective( w = rep(1/4,4) , R = indexes, portfolio = ObjSpec)

##       [,1]
## ES 0.1253
```

This is the CVaR of the equal-weight portfolio as computed by the function `ES` in the `PerformanceAnalytics` package of Carl and Peterson (2009)

```
library(PerformanceAnalytics)
out<-ES(indexes, weights = rep(1/4,4),p=0.95,
        portfolio_method="component")
out$MES
```

```
##          [,1]
## [1,] 0.1253
```

All arguments in the function `ES` can be passed on through `arguments`. E.g. to reduce the impact of extremes on the portfolio results, it is recommended to winsorize the data using the option clean="boudt".

```
out<-ES(indexes, weights = rep(1/4,4),p=0.95, clean="boudt",
        portfolio_method="component")
out$MES
```

```
##           [,1]
## [1,] 0.07125
```

For the formulation of the objective function, this implies setting:

```
ObjSpec = add.objective( portfolio = Wcons , type="risk",name="CVaR",
                         arguments=list(p=0.95,clean="boudt"), enabled=TRUE)
constrained_objective( w = rep(1/4,4) , R = indexes[,1:4] , portfolio = ObjSpec)
```

```
##         [,1]
## ES 0.07125
```

An additional argument that is not available for the moment in `ES` is to estimate the conditional covariance matrix through the constant conditional correlation model of Bollerslev (1990).

For the formulation of the objective function, this implies setting:

```
ObjSpec = add.objective( portfolio = Wcons , type="risk",name="CVaR",
                         arguments=list(p=0.95,clean="boudt"),
                         enabled=TRUE, garch=TRUE)
constrained_objective( w = rep(1/4,4) , R = indexes[,1:4] , portfolio = ObjSpec)
```

```
##
## Attaching package:  'timeDate'
##
## The following objects are masked from 'package:PerformanceAnalytics':
##
##    kurtosis, skewness
```

```
##
##
## Attaching package: 'timeSeries'
##
## The following object is masked from 'package:zoo':
##
##     time<-
##
## Loading required package: MASS
##
## Attaching package: 'fBasics'
##
## The following object is masked from 'package:base':
##
##     norm
```

2.3 Minimum CVaR concentration objective function

Add the minimum 95% CVaR concentration objective to the objective function:

```
ObjSpec = add.objective( portfolio = Wcons , type="risk_budget_objective",
                         name="CVaR", arguments=list(p=0.95, clean="boudt"),
                         min_concentration=TRUE, enabled=TRUE)
```

The value of the objective function is:

```
constrained_objective( w = rep(1/4,4) , R = indexes, portfolio = ObjSpec,
                       trace=TRUE)

## $out
## [1] 8.023
##
## $weights
## [1] 0.25 0.25 0.25 0.25
```

```
##
## $objective_measures
## $objective_measures$CVaR
## $objective_measures$CVaR$MES
##          [,1]
## [1,] 0.07125
##
## $objective_measures$CVaR$contribution
##        US Bonds    US Equities Int'l Equities    Commodities
##       0.0005939      0.0207483      0.0246365      0.0252713
##
## $objective_measures$CVaR$pct_contrib_MES
##        US Bonds    US Equities Int'l Equities    Commodities
##        0.008335       0.291205       0.345775       0.354685
```

We can verify that this is effectively the largest CVaR contribution of that portfolio as follows:

```
ES(indexes[,1:4],weights = rep(1/4,4),p=0.95,clean="boudt",
   portfolio_method="component")

## $MES
##          [,1]
## [1,] 0.07125
##
## $contribution
##        US Bonds    US Equities Int'l Equities    Commodities
##       0.0005939      0.0207483      0.0246365      0.0252713
##
## $pct_contrib_MES
##        US Bonds    US Equities Int'l Equities    Commodities
##        0.008335       0.291205       0.345775       0.354685
```

2.4 Risk allocation constraints

We see that in the equal-weight portfolio, the international equities and commodities investment cause more than 30% of total risk. We could specify as a constraint that no asset can contribute more than 30% to total portfolio risk with the argument `max_prisk=0.3`. This involves the construction of the following objective function:

```
ObjSpec = add.objective( portfolio = Wcons , type="risk_budget_objective",
                         name="CVaR", max_prisk = 0.3,
                         arguments=list(p=0.95,clean="boudt"), enabled=TRUE)
constrained_objective( w = rep(1/4,4) , R = indexes, portfolio = ObjSpec)

## [1] 1005
```

This value corresponds to the penalty parameter which has by default the value of 10000 times the exceedances: $10000 * (0.045775103 + 0.054685023) \approx 1004.601$.

3 Optimization

The penalty augmented objective function is minimized through Differential Evolution. Two parameters are crucial in tuning the optimization: `search_size` and `itermax`. The optimization routine

1. First creates the initial generation of `NP = search_size/itermax` guesses for the optimal value of the parameter vector, using the `random_portfolios` function generating random weights satisfying the weight constraints.

2. Then DE evolves over this population of candidate solutions using alteration and selection operators in order to minimize the objective function. It restarts `itermax` times.

It is important that `search_size/itermax` is high enough. It is generally recommended that this ratio is at least ten times the length of the weight vector. For more details on the use of DE strategy in portfolio allocation, we refer the reader to Ardia et al. (2010).

3.1 Minimum CVaR portfolio under an upper 40% CVaR allocation constraint

The portfolio object and functions needed to obtain the minimum CVaR portfolio under an upper 40% CVaR allocation objective are the following:

```
# Create the portfolio specification object
ObjSpec <- portfolio.spec(assets=colnames(indexes[,1:4]))
# Add box constraints
```

```
ObjSpec <- add.constraint(portfolio=ObjSpec, type='box', min = 0, max=1)
# Add the full investment constraint that specifies the weights must sum to 1.
ObjSpec <- add.constraint(portfolio=ObjSpec, type="full_investment")
# Add objective to minimize CVaR
ObjSpec <- add.objective(portfolio=ObjSpec, type="risk", name="CVaR",
                         arguments=list(p=0.95, clean="boudt"))
# Add objective for an upper 40% CVaR allocation
ObjSpec <- add.objective(portfolio=ObjSpec, type="risk_budget_objective",
                         name="CVaR", max_prisk=0.4,
                         arguments=list(p=0.95, clean="boudt"))
```

After the call to these functions it starts to explore the feasible space iteratively and is shown in the output. Iterations are given as intermediate output and by default every iteration will be printed. We set `traceDE=5` to print every 5 iterations and `itermax=50` for a maximum of 50 iterations.

```
set.seed(1234)
out <- optimize.portfolio(R=indexes, portfolio=ObjSpec,
                          optimize_method="DEoptim", search_size=2000,
                          traceDE=5, itermax=50, trace=TRUE)

##
## DEoptim package
## Differential Evolution algorithm in R
## Authors:  D. Ardia, K. Mullen, B. Peterson and J. Ulrich
##
## Leverage constraint min_sum and max_sum are restrictive,
##               consider relaxing.  e.g.  'full_investment' constraint should be min_sum=0.99
and max_sum=1.01
## Warning:  executing %dopar% sequentially:  no parallel backend registered
## Optimizer was unable to find a solution for target

print(out)

## [1] "Optimizer was unable to find a solution for target"
```

If `trace=TRUE` in `optimize.portfolio`, additional output from the DEoptim solver is included in the `out` object created by `optimize.portfolio`. The additional elements in the output are `DEoptim_objective_results` and `DEoutput`. The `DEoutput` element contains output from the function `DEoptim`. The `DEoptim_objective_results` element contains the weights, value of the objective measures, and other data at each iteration.

```
names(out)

## NULL

# View the DEoptim_objective_results information at the last iteration
out$DEoptim_objective_results[[601]]

## Error:  $ operator is invalid for atomic vectors


# Extract stats from the out object into a matrix
xtract <- extractStats(out)

## Error:  no applicable method for 'extractStats' applied to an object of class "character"

dim(xtract)

## Error:  object 'xtract' not found

head(xtract)

## Error:  error in evaluating the argument 'x' in selecting a method for function
'head':  Error:  object 'xtract' not found
```

It can be seen from the charts that although US Bonds has a higher weight allocation, the percentage contribution to risk is the lowest of all four indexes.

```
chart.Weights(out)

## Error:  no applicable method for 'chart.Weights' applied to an object of class "character"

chart.RiskBudget(out, risk.type = "pct_contrib", col = "blue", pch = 18)

## Error:  no applicable method for 'chart.RiskBudget' applied to an object of class
"character"
```

3.2 Minimum CVaR concentration portfolio

The functions needed to obtain the minimum CVaR concentration portfolio are the following:

```
# Create the portfolio specification object
ObjSpec <- portfolio.spec(assets=colnames(indexes))
# Add box constraints
ObjSpec <- add.constraint(portfolio=ObjSpec, type='box', min = 0, max=1)
# Add the full investment constraint that specifies the weights must sum to 1.
ObjSpec <- add.constraint(portfolio=ObjSpec, type="full_investment")
# Add objective for min CVaR concentration
ObjSpec <- add.objective(portfolio=ObjSpec, type="risk_budget_objective",
                         name="CVaR", arguments=list(p=0.95, clean="boudt"),
                         min_concentration=TRUE)

set.seed(1234)
out <- optimize.portfolio(R=indexes, portfolio=ObjSpec,
                          optimize_method="DEoptim", search_size=5000,
                          itermax=50, traceDE=5, trace=TRUE)

## Leverage constraint min_sum and max_sum are restrictive,
##              consider relaxing.  e.g.  'full_investment' constraint should be min_sum=0.99
and max_sum=1.01
## Optimizer was unable to find a solution for target
```

This portfolio has the near equal risk contribution characteristic:

```
print(out)

## [1] "Optimizer was unable to find a solution for target"


# Verify results with ES function
ES(indexes[,1:4], weights=out$weights, p=0.95, clean="boudt",
   portfolio_method="component")

## Error:  $ operator is invalid for atomic vectors
```

The 95% CVaR percent contribution to risk is near equal for all four indexes. The neighbor portfolios can be plotted to view other near optimal portfolios. Alternatively, the contribution to risk in absolute terms can plotted by setting risk.type=absolute".

```
chart.RiskBudget(out, neighbors=25, risk.type="pct_contrib",
                 col="blue", pch=18)

## Error:  no applicable method for 'chart.RiskBudget' applied to an object of class
"character"
```

3.3 Dynamic optimization

Dynamic rebalancing of the risk budget optimized portfolio is possible through the function optimize.portfolio.rebalancing. Additional arguments are rebalance_on which indicates the rebalancing frequency (years, quarters, months). The estimation is either done from inception (trailing_periods=0) or through moving window estimation, where each window has trailing_periods observations. The minimum number of observations in the estimation sample is specified by training_period. Its default value is 36, which corresponds to three years for monthly data.

As an example, consider the minimum CVaR concentration portfolio, with estimation from inception and monthly rebalancing. Since we require a minimum estimation length of total number of observations -1, we can optimize the portfolio only for the last two months.

```
library(iterators)
set.seed(1234)
out <- optimize.portfolio.rebalancing(R=indexes, portfolio=ObjSpec,
                                      optimize_method="DEoptim", search_size=5000,
                                      rebalance_on="months",
                                      training_period=nrow(indexes)-1,
                                      traceDE=10)

## Leverage constraint min_sum and max_sum are restrictive,
##                consider relaxing.  e.g.  'full_investment' constraint should be min_sum=0.99
and max_sum=1.01
## Optimizer was unable to find a solution for target
## Leverage constraint min_sum and max_sum are restrictive,
##                consider relaxing.  e.g.  'full_investment' constraint should be min_sum=0.99
and max_sum=1.01
```

```
## Optimizer was unable to find a solution for target
## overall elapsed time:0.853953838348389
```

The output of `optimize.portfolio.rebalancing` is a list of objects created by `optimize.portfolio`, one for each rebalancing period.

```
names(out)

## [1] "2009-11-30" "2009-12-31"

names(out[[1]])

## NULL

print(out)

## $`2009-11-30`
## [1] "Optimizer was unable to find a solution for target"
##
## $`2009-12-31`
## [1] "Optimizer was unable to find a solution for target"
##
## attr(,"class")
## [1] "optimize.portfolio.rebalancing"
```

The optimal weights for each rebalancing period can be extracted fron the object with the following function:

```
extractWeights(out)

## Error: $ operator is invalid for atomic vectors
```

Also the value of the objective function at each rebalancing period:

```
out[[1]]$out

## Error: $ operator is invalid for atomic vectors

out[[2]]$out

## Error: $ operator is invalid for atomic vectors
```

The first and last observation from the estimation sample:

```
out[[1]]$data_summary

## Error:  $ operator is invalid for atomic vectors

out[[2]]$data_summary

## Error:  $ operator is invalid for atomic vectors
```

Of course, DE is a stochastic optimizer and typically will only find a near-optimal solution that depends on the seed. The function `optimize.portfolio.parallel` in `PortfolioAnalytics` allows to run an arbitrary number of portfolio sets in parallel in order to develop "confidence bands" around your solution. It is based on Revolution's `foreach` package (Computing, 2009).

References

D. Ardia and K. Mullen. *DEoptim: Differential Evolution Optimization in R*, 2009. URL `http://CRAN.R-project.org/package=DEoptim`. R package version 2.00-04.

D. Ardia, K. Boudt, P. Carl, K. Mullen, and B. Peterson. Differential evolution (deoptim) for non-convex portfolio optimization. *Mimeo*, 2010.

T. Bollerslev. Modeling the coherence in short-run nominal exchange rates: A multivariate generalized ARCH model. *Review of Economics and Statistics*, 72:498–505, 1990.

K. Boudt, P. Carl, and B. G. Peterson. Portfolio optimization with conditional value-at-risk budgets, Jan. 2010.

K. Boudt, P. Carl, and B. G. Peterson. PortfolioAnalytics: Portfolio analysis, including numeric methods for optimization of portfolios, 2012. URL `https://r-forge.r-project.org/projects/returnanalytics/`. R package version 0.8.2.

P. Carl and B. G. Peterson. PerformanceAnalytics: Econometric tools for performance and risk analysis in R, 2009. URL `http://braverock.com/R/`. R package version 1.0.0.

P. Carl, B. G. Peterson, and K. Boudt. Business objectives and complex portfolio optimization. Presentation at R/Finance 2010. Available at: `http://www.rinfinance.com/agenda/2010/Carl+Peterson+Boudt_Tutorial.pdf`, 2010.

R. Computing. *foreach: Foreach looping construct for R*, 2009. URL `http://CRAN.R-project.org/package=foreach`. R package version 1.3.0.

K. M. Mullen, D. Ardia, D. L. Gil, D. Windover, and J. Cline. DEoptim: An R package for global optimization by differential evolution, Dec. 2009.

K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, Berlin, Germany, second edition, Dec. 2006. ISBN 3540209506.

J. A. Ryan and J. M. Ulrich. *xts: Extensible Time Series*, 2010. URL `http://CRAN.R-project.org/package=xts`. R package version 0.7-0.

R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997. ISSN 0925-5001.

Wuertz, Diethelm, Chalabi, Yohan, Chen, William, Ellis, and Andrew. *Portfolio Optimization with R/Rmetrics*. Rmetrics Association & Finance Online, www.rmetrics.org, April 2010. R package version 2110.79.

G. Yollin. R tools for portfolio optimization. In *Presentation at R/Finance conference 2009*, 2009.