Differential Evolution with DEoptim

An Application to Non-Convex Portfolio Optimization

by David Ardia, Kris Boudt, Peter Carl, Katharine M. Mullen and Brian G. Peterson

Abstract The R package **DEoptim** implements the Differential Evolution algorithm. This algorithm is an evolutionary technique similar to classic genetic algorithms that is useful for the solution of global optimization problems. In this note we provide an introduction to the package and demonstrate its utility for financial applications by solving a non-convex portfolio optimization problem.

Introduction

Differential Evolution (DE) is a search heuristic introduced by Storn and Price (1997). Its remarkable performance as a global optimization algorithm on continuous numerical minimization problems has been extensively explored (Price et al., 2006). DE has also become a powerful tool for solving optimization problems that arise in financial applications: for fitting sophisticated models (Gilli and Schumann, 2009), for performing model selection (Maringer and Meyer, 2008), and for optimizing portfolios under non-convex settings (Krink and Paterlini, 2011). DE is available in R with the package **DEoptim**.

In what follows, we briefly sketch the DE algorithm and discuss the content of the package **DEoptim**. The utility of the package for financial applications is then explored by solving a non-convex portfolio optimization problem.

Differential Evolution

DE belongs to the class of genetic algorithms (GAs) which use biology-inspired operations of crossover, mutation, and selection on a population in order to minimize an objective function over the course of successive generations (Holland, 1975). As with other evolutionary algorithms, DE solves optimization problems by evolving a population of candidate solutions using alteration and selection operators. DE uses floating-point instead of bit-string encoding of population members, and arithmetic operations instead of logical operations in mutation, in contrast to classic GAs.

Let NP denote the number of parameter vectors (members) $x \in \mathbb{R}^d$ in the population, where d denotes dimension. In order to create the initial generation, NP guesses for the optimal value of the parameter vector are made, either using random values between upper and lower bounds (defined by the user) or using values given by the user. Each generation involves creation of a new population from the current population members $\{x_i \,|\, i = 1,\ldots,NP\}$, where i indexes the vectors that make up the population. This is accomplished using *differential mutation* of the population members. An initial mutant parameter vector v_i is created by choosing three members of the population, x_{i_1}, x_{i_2} and x_{i_3}, at random. Then v_i is generated as

$$v_i \doteq x_{i_1} + F \cdot (x_{i_2} - x_{i_3}),$$

where F is a positive scale factor, effective values for which are typically less than one. After the first mutation operation, mutation is continued until d mutations have been made, with a crossover probability $CR \in [0,1]$. The crossover probability CR controls the fraction of the parameter values that are copied from the mutant. Mutation is applied in this way to each member of the population. If an element of the trial parameter vector is found to violate the bounds after mutation and crossover, it is reset in such a way that the bounds are respected (with the specific protocol depending on the implementation). Then, the objective function values associated with the children are determined. If a trial vector has equal or lower objective function value than the previous vector it replaces the previous vector in the population; otherwise the previous vector remains. Variations of this scheme have also been proposed; see Price et al. (2006).

Intuitively, the effect of the scheme is that the shape of the distribution of the population in the search space is converging with respect to size and direction towards areas with high fitness. The closer the population gets to the global optimum, the more the distribution will shrink and therefore reinforce the generation of smaller difference vectors.

For more details on the DE strategy, we refer the reader to Price et al. (2006) and Storn and Price (1997).

The package DEoptim

DEoptim (Ardia et al., 2011) was first published on CRAN in 2005. Since becoming publicly available, it has been used by several authors to solve optimization problems arising in diverse domains. We refer the reader to Mullen et al. (2011) for a detailed description of the package.

DEoptim consists of the core function `DEoptim` whose arguments are:

- `fn`: the function to be optimized (minimized).
- `lower`, `upper`: two vectors specifying scalar real lower and upper bounds on each parameter to

Electronic copy available at: http://ssrn.com/abstract=1584905

be optimized. The implementation searches between `lower` and `upper` for the global optimum of `fn`.

- `control`: a list of tuning parameters, among which: `NP` (default: $10 \cdot d$), the number of population members and `itermax` (default: 200), the maximum number of iterations (i.e., population generations) allowed. For details on the other `control` parameters, the reader is referred to the documentation manual (by typing `?DEoptim`). For convenience, the function `DEoptim.control()` returns a list with default elements of `control`.

- `...`: allows the user to pass additional arguments to the function `fn`.

The output of the function `DEoptim` is a member of the S3 class `DEoptim`. Members of this class have a `plot` and a `summary` method that allow to analyze the optimizer's output.

Let us quickly illustrate the package's usage with the minimization of the Rastrigin function in $\mathbb{R}^2$, which is a common test for global optimization:

```
> Rastrigin <- function(x) {
+   sum(x^2 - 10 * cos(2 * pi * x)) + 20
}
```

The global minimum is zero at point $\boldsymbol{x} = (0,0)'$. A perspective plot of the function is shown in Figure 1.

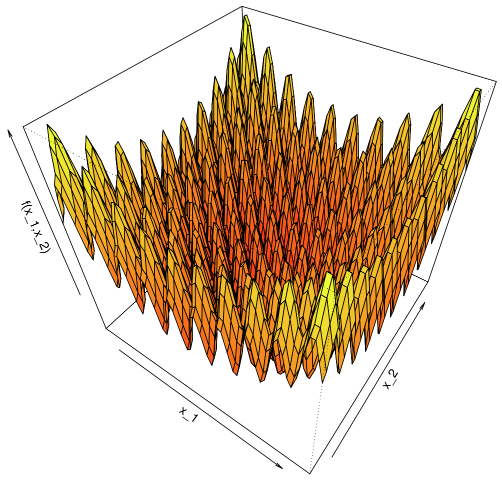

Figure 1: Perspective plot of the `Rastrigin` function.

The function `DEoptim` searches for a minimum of the objective function between `lower` and `upper` bounds. A call to `DEoptim` can be made as follows:

```
> set.seed(1234)
> DEoptim(fn = Rastrigin,
+   lower = c(-5, -5),
+   upper = c(5, 5),
+   control = list(storepopfrom = 1))
```

The above call specifies the objective function to minimize, `Rastrigin`, the lower and upper bounds on the parameters, and, via the `control` argument, that we want to store intermediate populations from the first generation onwards (`storepopfrom = 1`). Storing intermediate populations allows us to examine the progress of the optimization in detail. Upon initialization, the population is comprised of 50 random values drawn uniformly within the `lower` and `upper` bounds. The members of the population generated by the above call are plotted at the end of different generations in Figure 2. `DEoptim` consistently finds the minimum of the function (market with an open white circle) within 200 generations using the default settings. We have observed that **DEoptim** solves the Rastrigin problem more efficiently than the simulated annealing method available in the R function `optim` (for all annealing schedules tried). Note that users interested in exact reproduction of results should set the seed of their random number generator before calling `DEoptim` (using `set.seed`). DE is a randomized algorithm, and the results may vary between runs.

Finally, note that **DEoptim** relies on repeated evaluation of the objective function in order to move the population toward a global minimum. Therefore, users interested in making **DEoptim** run as fast as possible should ensure that evaluation of the objective function is as efficient as possible. Using pure R code, this may often be accomplished using vectorization. Writing parts of the objective function in a lower-level language like C or Fortran may also increase speed.

Risk allocation portfolios

Mean-risk models were developed in early fifties for the portfolio selection problem. Initially, variance was used as a risk measure. Since then, many alternative risk measures have been proposed. The question of which risk measure is most appropriate is still the subject of much debate. Value-at-Risk (VaR) and Conditional Value-at-Risk (CVaR) are the most popular measures of downside risk. VaR is the negative value of the portfolio return such that lower returns will only occur with at most a preset probability level, which typically is between one and five percent. CVaR is the negative value of the mean of all return realizations that are below the VaR. There is a voluminous literature on portfolio optimization problems with VaR and CVaR risk measures, see, e.g., Fábián and Veszprémi (2008) and the references therein.

Electronic copy available at: http://ssrn.com/abstract=1584905

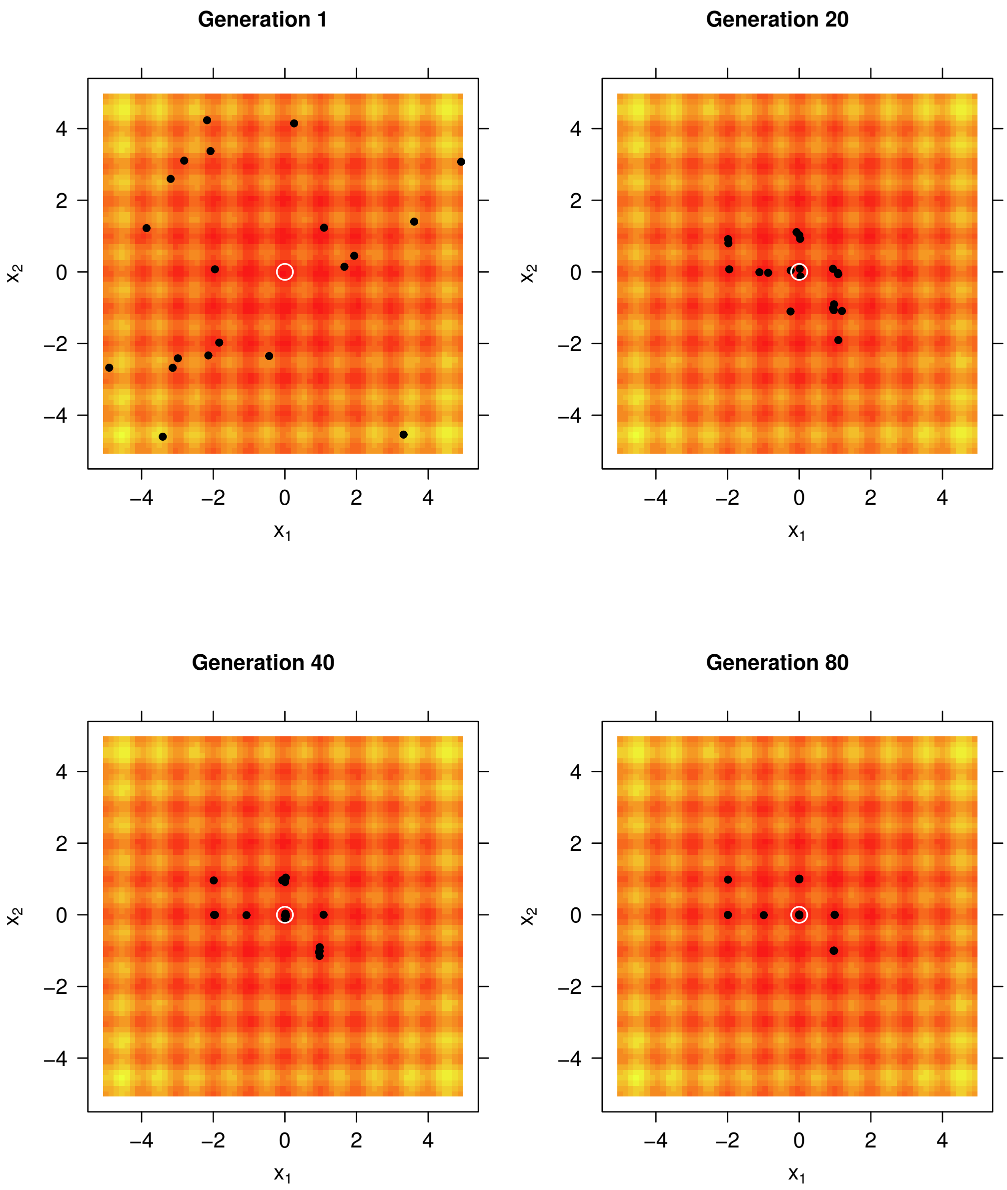

Figure 2: The population (market with black dots) associated with various generations of a call to `DEoptim` as it searches for the minimum of the `Rastrigin` function at point $x = (0,0)'$ (market with an open white circle). The minimum is consistently determined within 200 generations using the default settings of `DEoptim`.

Modern portfolio selection considers additional criteria such as to maximize upper-tail skewness and liquidity or minimize the number of securities in the portfolio. In the recent portfolio literature, it has been advocated by various authors to incorporate risk contributions in the portfolio allocation problem. Qian's (2005) *Risk Parity Portfolio* allocates portfolio variance equally across the portfolio components. Maillard et al. (2010) call this the *Equally-Weighted Risk Contribution (ERC) Portfolio*. They derive the theoretical properties of the ERC portfolio and show that its volatility is located between those of the minimum variance and equal-weight portfolio. Zhu et al. (2010) study optimal mean-variance portfolio selection under a direct constraint on the contributions to portfolio variance. Because the resulting optimization model is a non-convex quadratically constrained quadratic programming problem, they develop a branch-and-bound algorithm to solve it.

Boudt et al. (2010a) propose to use the contributions to portfolio CVaR as an input in the portfolio optimization problem to create portfolios whose percentage CVaR contributions are aligned with the desired level of CVaR risk diversification. Under the assumption of normality, the percentage CVaR contribution of asset i is given by the following explicit function of the vector of weights $\boldsymbol{w} \doteq (w_1, \ldots, w_d)'$, mean vector $\boldsymbol{\mu} \doteq (\mu_1, \ldots, \mu_d)'$ and covariance matrix $\boldsymbol{\Sigma}$:

$$\frac{w_i \left[-\mu_i + \frac{(\boldsymbol{\Sigma w})_i}{\sqrt{\boldsymbol{w}'\boldsymbol{\Sigma w}}} \frac{\phi(z_\alpha)}{\alpha} \right]}{-\boldsymbol{w}'\boldsymbol{\mu} + \sqrt{\boldsymbol{w}'\boldsymbol{\Sigma w}} \frac{\phi(z_\alpha)}{\alpha}},$$

with z_α the α-quantile of the standard normal distribution and $\phi(\cdot)$ the standard normal density function. Throughout the paper we set $\alpha = 5\%$. As we show here, the package **DEoptim** is well suited to solve these problems. Note that it is also the evolutionary optimization strategy used in the package **PortfolioAnalytics** (Boudt et al., 2010b).

To illustrate this, consider as a stylized example a five-asset portfolio invested in the stocks with tickers GE, IBM, JPM, MSFT and WMT. We keep the dimension of the problem low in order to allow interested users to reproduce the results using a personal computer. More realistic portfolio applications can be obtained in a straightforward manner from the code below, by expanding the number of parameters optimized. Interested readers can also see the **DEoptim** package vignette for a 100-parameter portfolio optimization problem that is typical of those encountered in practice.

We first download ten years of monthly data using the function `getSymbols` of the package **quantmod** (Ryan, 2010). Then we compute the log-return series and the mean and covariance matrix estimators. For on overview of alternative estimators of the covariance matrix, such as outlier robust or shrinkage estimators, and their implementation in portfolio allocation, we refer the reader to Würtz et al. (2009, Chapter 4). These estimators might yield better performance in the case of small samples, outliers or departures from normality.

```
> library("quantmod")
> tickers <- c("GE", "IBM", "JPM", "MSFT", "WMT")
> getSymbols(tickers,
+   from = "2000-12-01",
+   to   = "2010-12-31")
> P <- NULL
> for(ticker in tickers) {
+   tmp <- Cl(to.monthly(eval(parse(text = ticker))))
+   P <- cbind(P, tmp)
+ }
> colnames(P) <- tickers
> R <- diff(log(P))
> R <- R[-1,]
> mu <- colMeans(R)
> sigma <- cov(R)
```

We first compute the equal-weight portfolio. This is the portfolio with the highest weight diversification and often used as a benchmark. But is the risk exposure of this portfolio effectively well diversified across the different assets? This question can be answered by computing the percentage CVaR contributions with function `ES` in the package **PerformanceAnalytics** (Carl and Peterson, 2011). These percentage CVaR contributions indicate how much each asset contributes to the total portfolio CVaR.

```
> library("PerformanceAnalytics")
> pContribCVaR <- ES(weights = rep(0.2, 5),
+   method = "gaussian",
+   portfolio_method = "component",
+   mu = mu,
+   sigma = sigma)$pct_contrib_ES
> rbind(tickers, round(100 * pContribCVaR, 2))
        [,1]    [,2]    [,3]    [,4]    [,5]
tickers "GE"    "IBM"   "JPM"   "MSFT"  "WMT"
        "21.61" "18.6"  "25.1"  "25.39" "9.3"
```

We see that in the equal-weight portfolio, 25.39% of the portfolio CVaR risk is caused by the 20% investment in MSFT, while the 20% investment in WMT only causes 9.3% of total portfolio CVaR. The high risk contribution of MSFT is due to its high standard deviation and low average return (reported in percent):

```
> round(100 * mu , 2)
   GE   IBM   JPM  MSFT   WMT
-0.80  0.46 -0.06 -0.37  0.0
> round(100 * diag(sigma)^(1/2), 2)
   GE   IBM   JPM  MSFT   WMT
8.90  7.95  9.65 10.47  5.33
```

We now use the function `DEoptim` of the package **DEoptim** to find the portfolio weights for which the portfolio has the lowest CVaR and each investment can contribute at most 22.5% to total portfolio CVaR risk. For this, we first define our objective function to minimize. The current implementation of **DEoptim** allows for constraints on the domain space. To include the risk budget constraints, we add them to the objective function through a penalty function. As such, we allow the search algorithm to consider infeasible solutions. A portfolio which is unacceptable for the investor must be penalized enough to be rejected by the minimization process and the larger the violation of the constraint, the larger the increase in the value of the objective function. A standard expression of these violations is $\alpha \times |\text{violation}|^p$. Crama and Schyns (2003) describe several ways to calibrate the scaling factors α and p. If these values are too small, then the penalties do not play their expected role and the final solution may be infeasible. On the other hand, if α and p are too large, then the CVaR term becomes negligible with respect to the penalty; thus, small variations of $\boldsymbol{w}$ can lead to large variations of the penalty term, which mask the effect on the portfolio CVaR. We set $\alpha = 10^3$ and $p = 1$, but recognize that better choices may be possible and depend on the problem at hand.

```
> obj <- function(w) {
+   if (sum(w) == 0) { w <- w + 1e-2 }
+   w <- w / sum(w)
+   CVaR <- ES(weights = w,
+     method = "gaussian",
+     portfolio_method = "component",
+     mu = mu,
+     sigma = sigma)
+   tmp1 <- CVaR$ES
+   tmp2 <- max(CVaR$pct_contrib_ES - 0.225, 0)
+   out <- tmp1 + 1e3 * tmp2
+ }
```

The penalty introduced in the objective function is non-differentiable and therefore standard gradient-based optimization routines cannot be used. In contrast, `DEoptim` is designed to consistently find a good approximation to the global minimum of the optimization problem:

```
> set.seed(1234)
> out <- DEoptim(fn = obj,
+   lower = rep(0, 5),
+   upper = rep(1, 5))
> out$optim$bestval
[1] 0.1143538
> wstar <- out$optim$bestmem
> wstar <- wstar / sum(wstar)
> rbind(tickers, round(100 * wstar, 2))
        par1    par2    par3    par4    par5
tickers "GE"    "IBM"   "JPM"   "MSFT"  "WMT"
        "18.53" "21.19" "11.61" "13.37" "35.3"
> 100 * (sum(wstar * mu) - mean(mu))
[1] 0.04827935
```

Note that the main differences with the equal-weight portfolio is the low weights given to JPM and MSFT and the high weight to WMT. As can be seen from the last two lines, this *minimum risk* portfolio has a higher expected return than the equal weight portfolio. The following code illustrates that DEoptim yields superior results than the gradient-based optimization routines available in R.

```
> out <- optim(par = rep(0.2, 5),
+  fn = obj,
+  method = "L-BFGS-B",
+  lower = rep(0, 5),
+  upper = rep(1, 5))
> out$value
[1] 0.1255093
> out <- nlminb(start = rep(0.2, 5),
+   objective = obj,
+   lower = rep(0, 5),
+   upper = rep(1, 5))
> out$objective
[1] 0.1158250
```

Even on this relatively simple stylized example, the `optim` and `nlminb` routines converged to local minima.

Suppose now the investor is interested in the most risk diversified portfolio whose expected return is higher than the equal-weight portfolio. This amounts to minimizing the largest CVaR contribution subject to a return target and can be implemented as follows:

```
> obj <- function(w) {
+   if(sum(w) == 0) { w <- w + 1e-2 }
+   w <- w / sum(w)
+   contribCVaR <- ES(weights = w,
+     method = "gaussian",
+     portfolio_method = "component",
+     mu = mu,
+     sigma = sigma)$contribution
+   tmp1 <- max(contribCVaR)
+   tmp2 <- max(mean(mu) - sum(w * mu), 0)
+   out <- tmp1 + 1e3 * tmp2
+ }
> set.seed(1234)
> out <- DEoptim(fn = obj,
+   lower = rep(0, 5),
+   upper = rep(1, 5))
> wstar <- out$optim$bestmem
> wstar <- wstar / sum(wstar)
> rbind(tickers, round(100 * wstar, 2))
        par1    par2    par3    par4    par5
tickers "GE"    "IBM"   "JPM"   "MSFT"  "WMT"
        "17.38" "19.61" "14.85" "15.19" "32.98"
> 100 * (sum(wstar * mu) - mean(mu))
[1] 0.04150506
```

This portfolio invests more in the JPM stock and less in the GE (which has the lowest average return) compared to the portfolio with the upper 22.5% percentage CVaR constraint. We refer to Boudt et al. (2010a) for a more elaborate study on using CVaR allocations as an objective function or constraint in the portfolio optimization problem.

A classic risk/return (i.e., CVaR/mean) scatter chart showing the results for portfolios tested by

DEoptim is displayed in Figure 3. Gray elements depict the results for all tested portfolios (using hexagon binning which is a form of bivariate histogram, see Carr et al. (2011); higher density regions are darker). The yellow-red line shows the path of the best member of the population over time, with the darkest solution at the end being the *optimal* portfolio. We can notice how DEoptim does not spend much time computing solutions in the scatter space that are suboptimal, but concentrates the bulk of the calculation time in the vicinity of the final *best* portfolio. Note that we are not looking for the tangency portfolio; simple mean/CVaR optimization can be achieved with standard optimizers. We are looking here for the best balance between return and risk concentration.

We have utilized the mean return/CVaR concentration examples here as more realistic, but still stylized, examples of non-convex objectives and constraints in portfolio optimization; other non-convex objectives, such as drawdown minimization, are also common in real portfolios, and are likewise suitable to application of Differential Evolution. One of the key issues in practice with real portfolios is that a portfolio manager rarely has only a single objective or only a few simple objectives combined. For many combinations of objectives, there is no unique global optimum, and the constraints and objectives formed lead to a non-convex search space. It may take several hours on very fast machines to get the best answers, and the best answers may not be a true global optimum, they are just *as close as feasible* given potentially competing and contradictory objectives.

When the constraints and objectives are relatively simple, and may be reduced to quadratic, linear, or conical forms, a simpler optimization solver will produce answers more quickly. When the objectives are more layered, complex, and potentially contradictory, as those in real portfolios tend to be, **DEoptim** or other global optimization algorithms such as those integrated into **PortfolioAnalytics** provide a portfolio manager with a feasible option for optimizing their portfolio under real-world non-convex constraints and objectives.

The **PortfolioAnalytics** framework allows any arbitrary R function to be part of the objective set, and allows the user to set the relative weighting that they want on any specific objective, and use the appropriately tuned optimization solver algorithm to locate portfolios that most closely match those objectives.

Summary

In this note we have introduced DE and **DEoptim**. The package **DEoptim** provides a means of applying the DE algorithm in the R language and environment for statistical computing. DE and the package **DEoptim** have proven themselves to be powerful tools for the solution of global optimization problems in a wide variety of fields. We have referred interested users to Price et al. (2006) and Mullen et al. (2011) for a more extensive introduction, and further pointers to the literature on DE. The utility of using **DEoptim** was further demonstrated with a simple example of a stylized non-convex portfolio risk contribution allocation, with users referred to **PortfolioAnalytics** for portfolio optimization using DE with real portfolios under non-convex constraints and objectives.

The latest version of the package includes the adaptive Differential Evolution framework by Zhang and Sanderson (2009). Future work will also be directed at parallelization of the implementation. The **DEoptim** project is hosted on R-forge at https://r-forge.r-project.org/projects/deoptim/.

Acknowledgements

The authors would like to thank Dirk Eddelbuettel, Juan Ospina, and Enrico Schumann for useful comments and Rainer Storn for his advocacy of DE and making his code publicly available. The authors are also grateful to two anonymous reviewers for helpful comments that have led to improvements of this note. Kris Boudt gratefully acknowledges financial support from the National Bank of Belgium.

Disclaimer

The views expressed in this note are the sole responsibility of the authors and do not necessarily reflect those of aeris CAPITAL AG, Guidance Capital Management, Cheiron Trading, or any of their affiliates. Any remaining errors or shortcomings are the authors' responsibility.

Bibliography

D. Ardia, K. M. Mullen, B. G. Peterson and J. Ulrich. *DEoptim: Differential Evolution Optimization in R*, 2011. URL http://CRAN.R-project.org/package=DEoptim. R package version 2.1-0.

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

K. Boudt, P. Carl, and B. G. Peterson. *PortfolioAnalytics: Portfolio Analysis, including Numeric Methods for Optimization of Portfolios*, 2010b. URL http://r-forge.r-project.org/projects/returnanalytics/. R package version 0.6.

P. Carl and B. G. Peterson. *PerformanceAnalytics: Econometric tools for performance and risk analysis.*, 2011. URL http://r-forge.r-project.org/

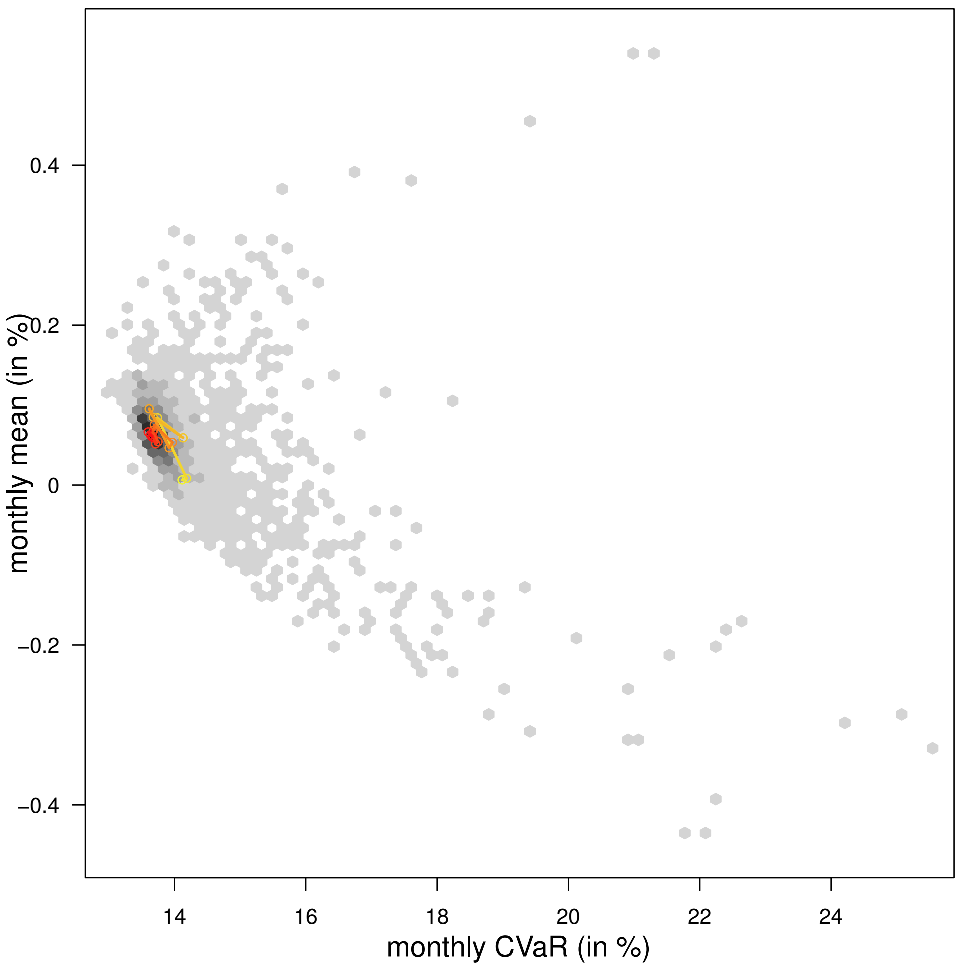

Figure 3: Risk/return scatter chart showing the results for portfolios tested by `DEoptim`.

projects/returnanalytics/. R package version 1.0.3.3.

D. Carr, N. Lewin-Koh and M. Maechler. *hexbin: Hexagonal binning routines*, 2011. URL http://CRAN.R-project.org/package=hexbin. R package version 1.26.0

Y. Crama and M. Schyns. Simulated annealing for complex portfolio selection problems. *European Journal of Operations Research*, 150(3):546–571, 2003.

C. Fábián and A. Veszprémi. Algorithms for handling CVaR constraints in dynamic stochastic programming models with applications to finance. *Journal of Risk*, 10(3):111–131, 2008.

M. Gilli and E. Schumann. Heuristic optimisation in financial modelling. COMISEF wps-007 09/02/2009, 2009.

J. H. Holland. *Adaptation in Natural Artificial Systems*. University of Michigan Press, 1975.

T. Krink and S. Paterlini. Multiobjective optimization using Differential Evolution for real-world portfolio optimization. *Computational Management Science*, 8:157–179, 2011.

S. Maillard, T. Roncalli, and J. Teiletche. On the properties of equally-weighted risk contributions portfolios. *Journal of Portfolio Management*, 36(4):60–70, 2010.

D. G. Maringer and M. Meyer. Smooth transition autoregressive models: New approaches to the model selection problem. *Studies in Nonlinear Dynamics & Econometrics*, 12(1):1–19, 2008.

K. M. Mullen, D. Ardia, D. L. Gil, D. Windover, and J. Cline. DEoptim: An R package for global optimization by Differential Evolution. *Journal of Statistical Software*, 40(6):1–26, 2011.

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

E. Qian. Risk parity portfolios: Efficient portfolios through true diversification of risk. *PanAgora Asset Management working paper*, 2005.

J. A. Ryan. *quantmod: Quantitative Financial Modelling Framework*, 2010. URL http://CRAN.R-project.org/package=quantmod. R package version 0.3-16.

O. Scaillet. Nonparametric estimation and sensitivity analysis of expected shortfall. *Mathematical Finance*, 14(1):74–86, 2002.

R. Storn and K. Price. Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

D. Würtz, Y. Chalabi, W. Chen and A. Ellis *Portfolio Optimization with R/Rmetrics*. Rmetrics Association and Finance Online, 2009.

J. Zhang and A. C. Sanderson. JADE: Adaptive Differential Evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, 2009.

S. Zhu, D. Li, and X. Sun. Portfolio selection with marginal risk control. *Journal of Computational Finance*, 14(1):1–26, 2010.

David Ardia
aeris CAPITAL AG, Switzerland
da@aeris-capital.com

Kris Boudt
Lessius and K.U.Leuven, Belgium
kris.boudt@econ.kuleuven.be

Peter Carl
Guidance Capital Management, Chicago, IL
pcarl@gsb.uchicago.edu

Katharine M. Mullen
National Institute of Standards and Technology
Gaithersburg, MD
katharine.mullen@nist.gov

Brian G. Peterson
Cheiron Trading, Chicago, IL
brian@braverock.com