Fast Design of Risk Parity Portfolios

Zé Vinícius and Daniel P. Palomar
Hong Kong University of Science and Technology (HKUST)
2018-11-22

Contents

1 Vanilla risk parity portfolio **1**

2 Modern risk parity portfolio **4**

3 Comparison with other packages **5**

4 Appendix I: Risk concentration formulations **7**

5 Appendix II: Computational time **8**

References **13**

This vignette illustrates the design of risk-parity portfolios, widely used by practitioners in the financial industry, with the package `riskParityPortfolio`, gives a description of the algorithms used, and compares the performance against existing packages such as `cccp` and `FinCovRegularization`.

1 Vanilla risk parity portfolio

A risk parity portfolio denotes a class of portfolios whose assets verify the following equalities:

$$w_i \frac{\partial f(\mathbf{w})}{\partial w_i} = w_j \frac{\partial f(\mathbf{w})}{\partial w_j}, \forall i, j,$$

where f is a positively homogeneous function of degree one that measures the total risk of the portfolio and $\mathbf{w}$ is the portfolio weight vector. In other words, the marginal risk contributions for every asset in a risk parity portfolio are equal. A common choice for f, for instance, is the standard deviation of the portfolio, which is usually called volatility, i.e., $f(\mathbf{w}) = \sqrt{\mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w}}$, where $\boldsymbol{\Sigma}$ is the covariance matrix of the assets.

With that particular choice of f, the risk parity requirements become

$$w_i (\Sigma \mathbf{w})_i = w_j (\Sigma \mathbf{w})_j, \forall i, j.$$

A natural extension of the risk parity portfolio is the so called risk budget portfolio, in which the marginal risk contributions match preassigned quantities. Mathematically,

$$(\Sigma \mathbf{w})_i w_i = b_i \mathbf{w}^T \Sigma \mathbf{w}, \forall i,$$

where $\mathbf{b} \triangleq (b_1, b_2, ..., b_N)$ is the vector of desired marginal risk contributions.

In the case that $\boldsymbol{\Sigma}$ is diagonal and with the constraints $\mathbf{1}^T \mathbf{w} = 1$ and $\mathbf{w} \geq \mathbf{0}$, the risk budgeting portfolio is

$$w_i = \frac{\sqrt{b_i} / \sqrt{\Sigma_{ii}}}{\sum_{k=1}^{N} \sqrt{b_k} / \sqrt{\Sigma_{kk}}}, \qquad i = 1, \ldots, N.$$

However, for non-diagonal $\boldsymbol{\Sigma}$ or with other additional constraints or objective function terms, a closed-form solution does not exist and some optimization procedures have to be constructed. The previous diagonal solution can always be used and is called *naive risk budgeting portfolio*.

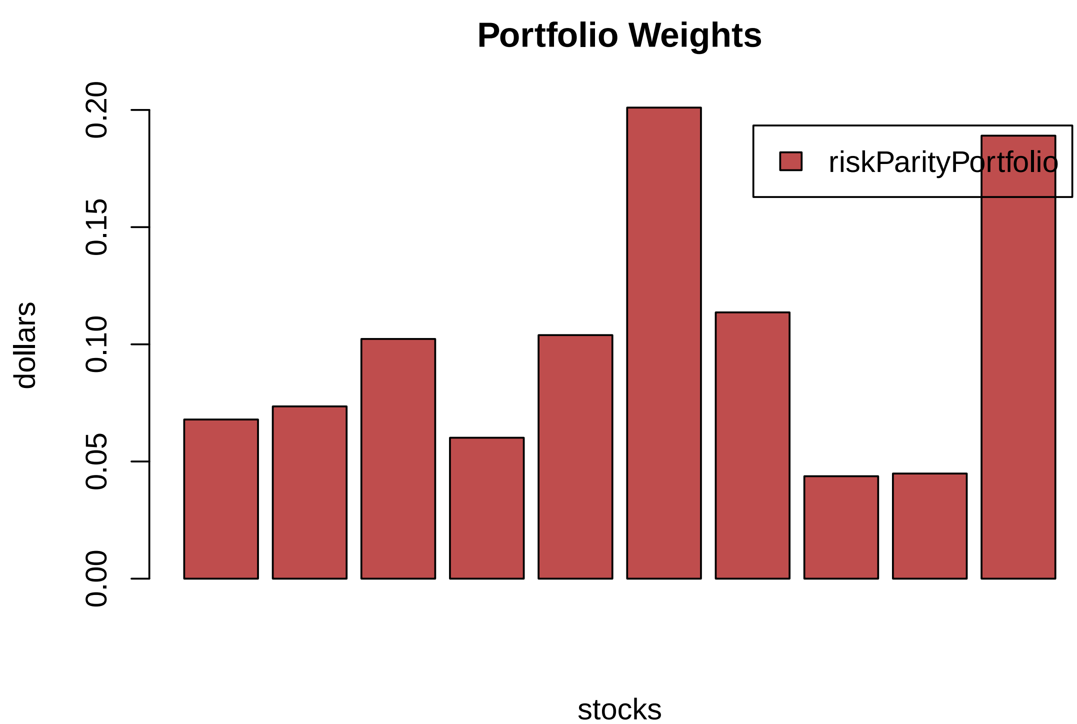
With the goal of designing risk budget portfolios, Spinu proposed in [1] to solve the following convex optimization problem:

$$\begin{array}{ll} \underset{\mathbf{w}}{\text{minimize}} & \frac{1}{2}\mathbf{w}^T\Sigma\mathbf{w} - \sum_{i=1}^{N} b_i \log(w_i) \\ \text{subject to} & \mathbf{1}^T\mathbf{w} = 1 \\ & \mathbf{w} \geq \mathbf{0}. \end{array}$$

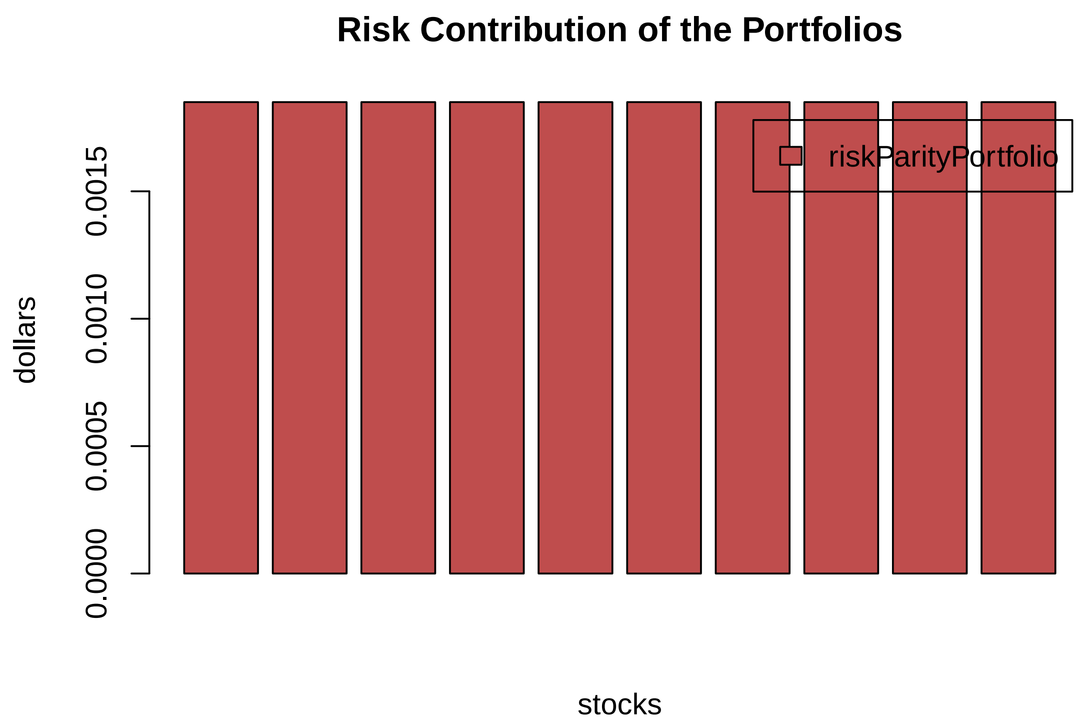
It turns out, as shown in [1], that the unique solution for the optimization problem stated above attains the risk budget requirements in an exact fashion. Such solution can be computed using convex optimization packages, such as CVXR, but a faster implementation of a Newton algorithm, also proposed by [1], is implemented in this package.

A simple code example on how to design a risk parity portfolio is as follows:

```r
library(riskParityPortfolio)
library(PerformanceAnalytics)

# generate synthetic data
set.seed(123)
N <- 10
V <- matrix(rnorm(N^2), nrow = N)
Sigma <- cov(V)
# compute risk parity portfolio
portfolio <- riskParityPortfolio(Sigma)
# plot the portfolio designed by each method
barplot(portfolio$w, main = "Portfolio Weights", xlab = "stocks", ylab = "dollars",
        beside = TRUE, col = rainbow8equal[1], legend = c("riskParityPortfolio"))
```




```r
# plot the risk contributions
barplot(portfolio$risk_contribution,
        main = "Risk Contribution of the Portfolios", xlab = "stocks", ylab = "dollars",
        beside = TRUE, col = rainbow8equal[1], legend = c("riskParityPortfolio"))
```

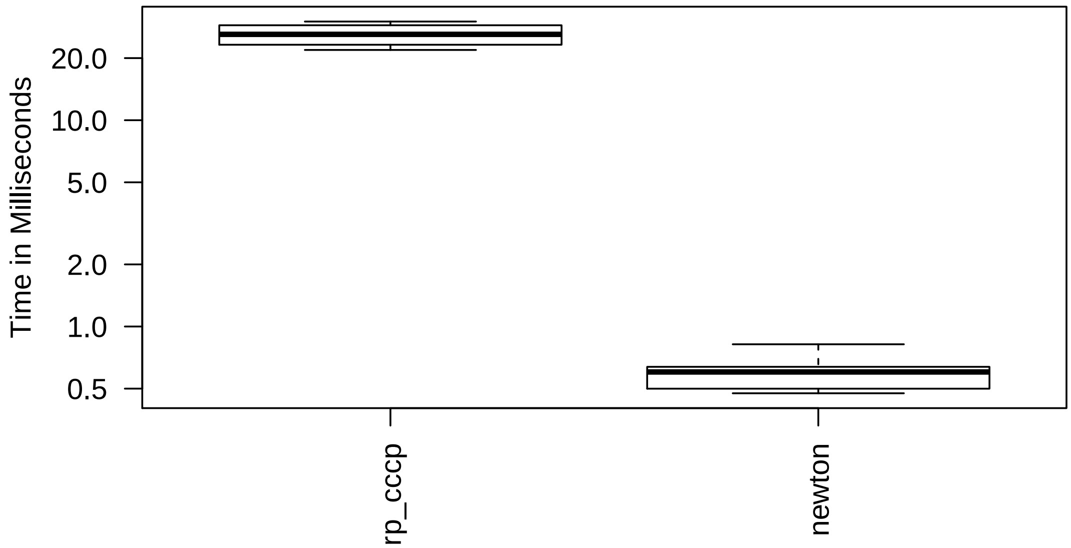


As presented earlier, the risk parity portfolios are designed in such a way as to ensure equal risk contribution from the assests, which can be noted in the chart above.

Now, let's see a comparison, in terms of computational time, of our Newton implementation against the `rp()` function from the `cccp` package. (For a fair comparison, instead of calling our function `riskParityPortfolio()`, we call directly the core internal function `risk_parity_portfolio_nn()`, which only computes the risk parity weights, just like `rp()`.)

```
library(microbenchmark)
library(cccp)
library(riskParityPortfolio)

N <- 100
V <- matrix(rnorm(N^2), nrow = N)
Sigma <- V %*% t(V)
b <- rep(1/N, N)

# use risk_parity_portfolio_nn with default values of tolerance and number of iterations
op <- microbenchmark(rp_cccp = rp(b, Sigma, b, optctrl = ctrl(trace = FALSE)),
                     newton = riskParityPortfolio:::risk_parity_portfolio_nn(Sigma, b,
                                                                              1e-6, 50),
                     times = 10L)
print(op)
#> Unit: microseconds
#>     expr       min        lq       mean     median        uq       max
#>  rp_cccp 21905.720 23217.269 26101.3075 26074.5450 28859.526 30075.774
#>   newton   474.669   499.681   604.6747   602.0945   637.633   820.085
#>  neval
#>     10
#>     10
par(mar = c(7, 4, 4, 2))
boxplot(op, ylab = "Time in Milliseconds",
```

```
        xlab = NULL, unit = "ms", outline = FALSE, las = 2)
```



As it can be observed, our implementation is quite faster (a factor of ~6x) than the interior-point method used by `cccp`. We suggest the interested reader to check out Chapter 11 of reference [2] for a thorough explanation on interior-point methods.

2 Modern risk parity portfolio

The design of risk parity portfolios as solved by [1] is of paramount importance both for academia and industry. However, practitioners would like the ability to include additional constraints and objective terms desired in practice, such as the mean return, box constraints, etc. In such cases, the risk-contribution constraints cannot be met with equality and gives rise to nonconvex formulations.

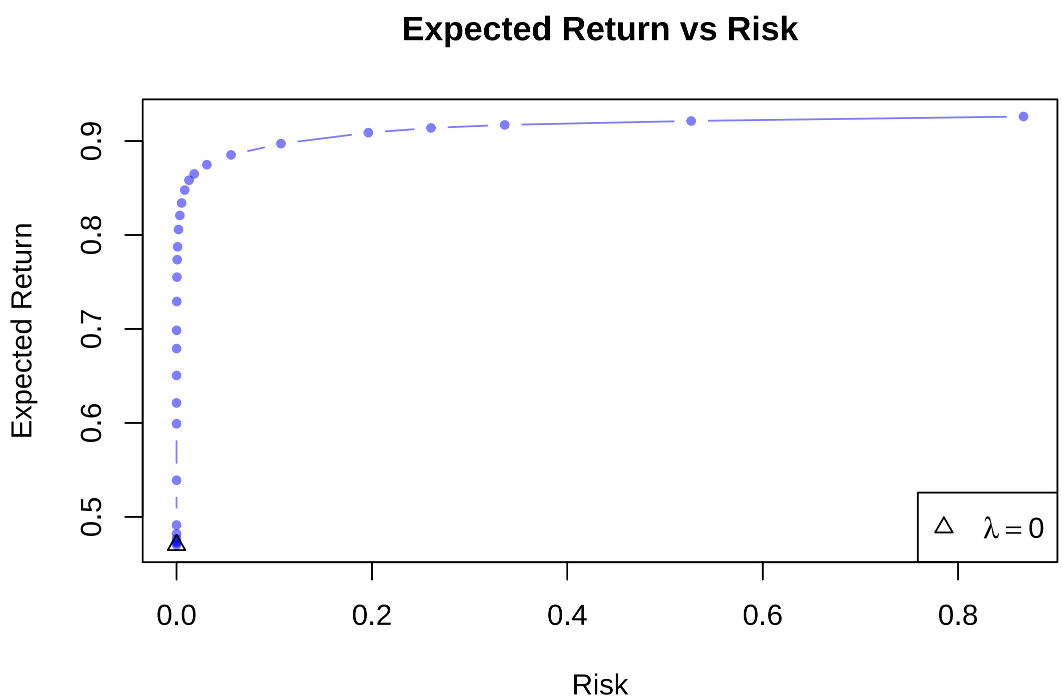
Let's explore, for instance, the effect of including the expected return as an additional objective in the optimization problem. The problem can be formulated as

$$\begin{array}{ll} \underset{\mathbf{w}}{\textsf{minimize}} & R(\mathbf{w}) - \lambda \mathbf{w}^T \boldsymbol{\mu} \\ \textsf{subject to} & \mathbf{1}^T \mathbf{w} = 1, \mathbf{w} \geq \mathbf{0}, \end{array}$$

where $R(\mathbf{w}) = \sum_{i=1}^{N} \left(w_i \left(\boldsymbol{\Sigma}\mathbf{w}\right)_i - b_i \mathbf{w}^T \boldsymbol{\Sigma}\mathbf{w}\right)^2$ is the risk concentration function, $\mathbf{w}^T \boldsymbol{\mu}$ is the expected return, and λ is a trade-off parameter.

```
library(scales)
library(latex2exp)

N <- 100
V <- matrix(rnorm(N^2), nrow = N)
Sigma <- cov(V)
mu <- runif(N)
w0 <- riskParityPortfolio(Sigma)$w  # vanilla solution as starting point

lmd_sweep <- c(0, 10 ^ (seq(-5, 2, .25)))
mean_return <- c()
risk_parity <- c()

for (lmd in lmd_sweep) {
```

```r
  rpp <- riskParityPortfolio(Sigma, w0 = w0, mu = mu, lambda = lmd,
                             formulation = "rc-double-index")
  mean_return <- c(mean_return, rpp$mean_return)
  risk_parity <- c(risk_parity, rpp$risk)
}

plot(risk_parity, mean_return, type = "b", pch=19, cex=.6, col=alpha("blue", .5),
     xlab = "Risk", ylab = "Expected Return",
     ylim = c(min(mean_return), max(mean_return)),
     xlim = c(min(risk_parity), max(risk_parity)),
     main = "Expected Return vs Risk")
points(risk_parity[1], mean_return[1], pch=24)
legend("bottomright", legend=TeX("$\\lambda = 0$"), col="black", pch = 24)
```

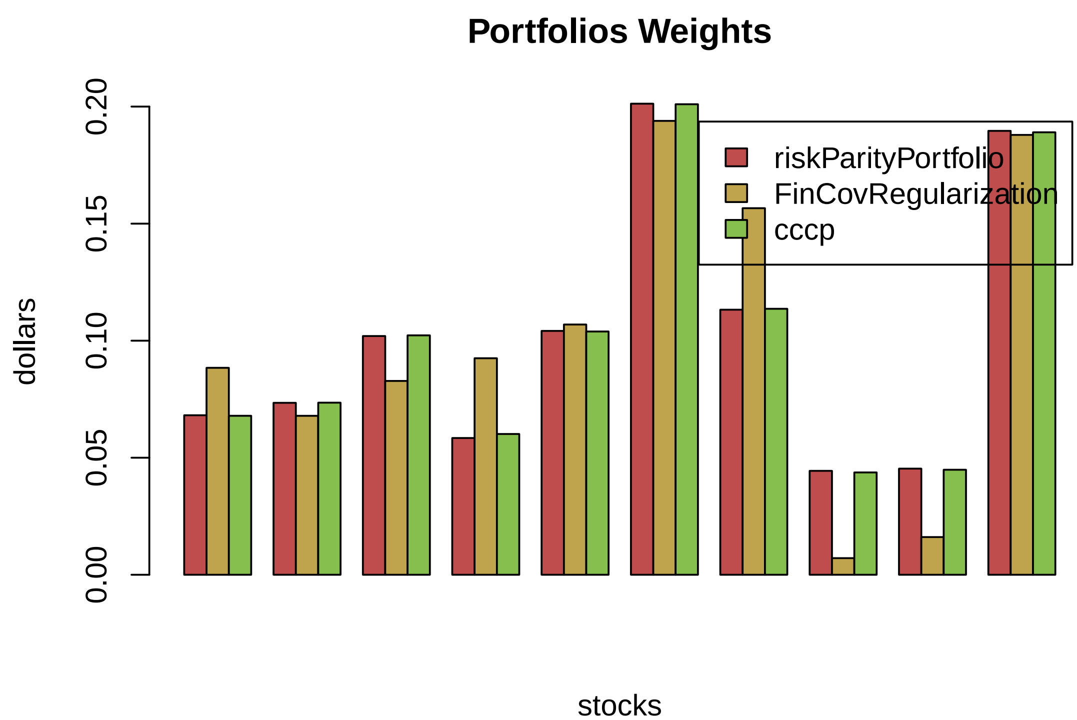


3 Comparison with other packages

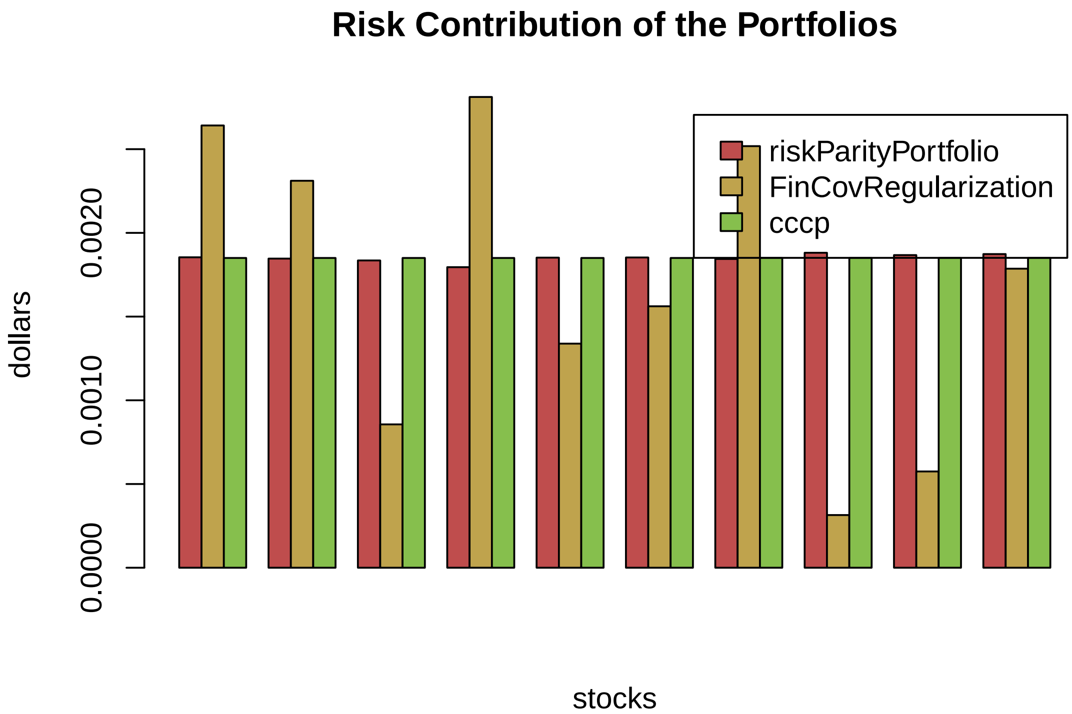
Others R packages with the goal of designing risk parity portfolios do exist, such as `FinCovRegularization` and `cccp`. Let's check how do they perform against `riskParityPortfolio`.

```r
library(FinCovRegularization)
library(cccp)

# generate synthetic data
set.seed(123)
N <- 10
V <- matrix(rnorm(N^2), nrow = N)
Sigma <- cov(V)

# uniform initial guess for the portfolio weights
w0 <- rep(1/N, N)
```

```r
# compute risk parity portfolios using different methods
rpp <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-double-index")
fincov_w <- RiskParity(Sigma)
fincov_risk_contribution <- c(fincov_w * (Sigma %*% fincov_w))
# The list of arguments is the following: 1) the initial guess for the portfolio weights,
# 2) the covariance matrix of the asset returns, 3) the marginal risk contributions
cccp_w <- c(getx(rp(w0, Sigma, mrc = w0, optctrl = ctrl(trace = FALSE))))
cccp_risk_contribution <- c(cccp_w * (Sigma %*% cccp_w))

barplot(rbind(rpp$w, fincov_w, cccp_w),
        main = "Portfolios Weights", xlab = "stocks", ylab = "dollars",
        beside = TRUE, col = rainbow8equal[1:3],
        legend = c("riskParityPortfolio", "FinCovRegularization", "cccp"))
```




```r
barplot(rbind(rpp$risk_contribution, fincov_risk_contribution, cccp_risk_contribution),
        main = "Risk Contribution of the Portfolios", xlab = "stocks", ylab = "dollars",
        beside = TRUE, col = rainbow8equal[1:3],
        legend = c("riskParityPortfolio", "FinCovRegularization", "cccp"))
```



Apart from the `RiskParity()` function from the `FinCovRegularization` package, the other functions perform the same. `FinCovRegularization` uses a general solver with a hard coded initial guess for the portfolio weights. We conjecture that its poor initialization might be the reason for the convergence to a local solution.

4 Appendix I: Risk concentration formulations

In general, with different constraints and objective functions, exact parity cannot be achieved and one needs to define a risk term to be minimized: $R(\mathbf{w}) = \sum_{i=1}^{N} (g_i(\mathbf{w}))^2$, where the g_i's denote the different risk contribution errors, e.g., $g_i = w_i(\boldsymbol{\Sigma}\mathbf{w})_i - b_i\mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}$. A double-index summation can also be used: $R(\mathbf{w}) = \sum_{i,j=1}^{N} (g_{ij}(\mathbf{w}))^2$.

We consider the risk formulations as presented in [3]. They can be passed through the keyword argument `formulation` in the function `riskParityPortfolio`.

The name of the formulations and their mathematical expressions are presented as follows.

Formulation "rc-double-index":

$$R(\mathbf{w}) = \sum_{i,j=1}^{N} \left(w_i(\boldsymbol{\Sigma}\mathbf{w})_i - w_j(\boldsymbol{\Sigma}\mathbf{w})_j\right)^2$$

Formulation "rc-vs-theta":

$$R(\mathbf{w}, \theta) = \sum_{i=1}^{N} \left(w_i(\boldsymbol{\Sigma}\mathbf{w})_i - \theta\right)^2$$

Formulation "rc-over-var-vs-b":

$$R(\mathbf{w}) = \sum_{i=1}^{N} \left(\frac{w_i(\boldsymbol{\Sigma}\mathbf{w})_i}{\mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}} - b_i\right)^2$$

Formulation "rc-over-b double-index":

$$R(\mathbf{w}) = \sum_{i,j=1}^{N} \left(\frac{w_i(\boldsymbol{\Sigma}\mathbf{w})_i}{b_i} - \frac{w_j(\boldsymbol{\Sigma}\mathbf{w})_j}{b_j}\right)^2$$

Formulation "rc-vs-b-times-var":

$$R(\mathbf{w}) = \sum_{i=1}^{N} \left(w_i \left(\boldsymbol{\Sigma}\mathbf{w}\right)_i - b_i \mathbf{w}^T \boldsymbol{\Sigma}\mathbf{w}\right)^2$$

Formulation "rc-over-sd vs b-times-sd":

$$R(\mathbf{w}) = \sum_{i=1}^{N} \left(\frac{w_i \left(\boldsymbol{\Sigma}\mathbf{w}\right)_i}{\sqrt{\mathbf{w}^T \boldsymbol{\Sigma}\mathbf{w}}} - b_i \sqrt{\mathbf{w}^T \boldsymbol{\Sigma}\mathbf{w}}\right)^2$$

Formulation "rc-over-b vs theta":

$$R(\mathbf{w}, \theta) = \sum_{i=1}^{N} \left(\frac{w_i \left(\boldsymbol{\Sigma}\mathbf{w}\right)_i}{b_i} - \theta\right)^2$$

Formulation "rc-over-var":

$$R(\mathbf{w}) = \sum_{i=1}^{N} \left(\frac{w_i \left(\boldsymbol{\Sigma}\mathbf{w}\right)_i}{\mathbf{w}^T \boldsymbol{\Sigma}\mathbf{w}}\right)^2$$

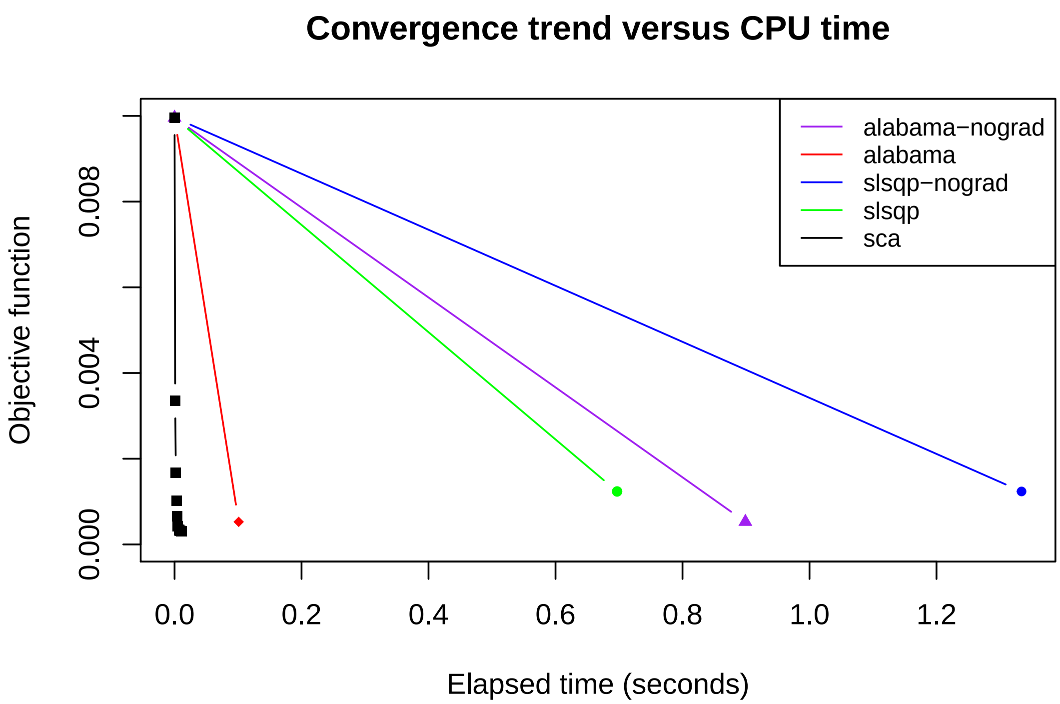
5 Appendix II: Computational time

In the subsections that follows we explore the computational time required by `method="sca"`, `method="alabama"`, and `method="slsqp"` for some of the formulations presented above. Additionally, we compare `method="alabama"` and `method="slsqp"` without using the gradient of the objective function.

5.1 Experiment: formulation "rc-over-var vs b"

```
set.seed(123)
N <- 100
V <- matrix(rnorm(N^2), nrow = N)
Sigma <- V %*% t(V)
w0 <- riskParityPortfolio(Sigma, formulation = "diag")$w

res_slsqp <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-var vs b", method = "slsqp")
#> For consistency with the rest of the package the inequality sign may be switched from >= to <= in a 
res_slsqp_nograd <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-var vs b",
                                        method = "slsqp", use_gradient = FALSE)
#> For consistency with the rest of the package the inequality sign may be switched from >= to <= in a 
res_alabama <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-var vs b", method = "alabama")
res_alabama_nograd <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-var vs b",
                                          method = "alabama", use_gradient = FALSE)
res_sca <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-var vs b", method = "sca")

plot(res_slsqp_nograd$elapsed_time, res_slsqp_nograd$obj_fun, type = "b",
     pch=19, cex=.6, col = "blue", xlab = "Elapsed time (seconds)",
     ylab = "Objective function", main = "Convergence trend versus CPU time",
     ylim = c(0, 0.01))
lines(res_alabama$elapsed_time, res_alabama$obj_fun, type = "b", pch=18, cex=.8,
      col = "red")
lines(res_alabama_nograd$elapsed_time, res_alabama_nograd$obj_fun, type = "b", pch=17,
      cex=.8, col = "purple")
lines(res_slsqp$elapsed_time, res_slsqp$obj_fun, type = "b", pch=16, cex=.8,
```

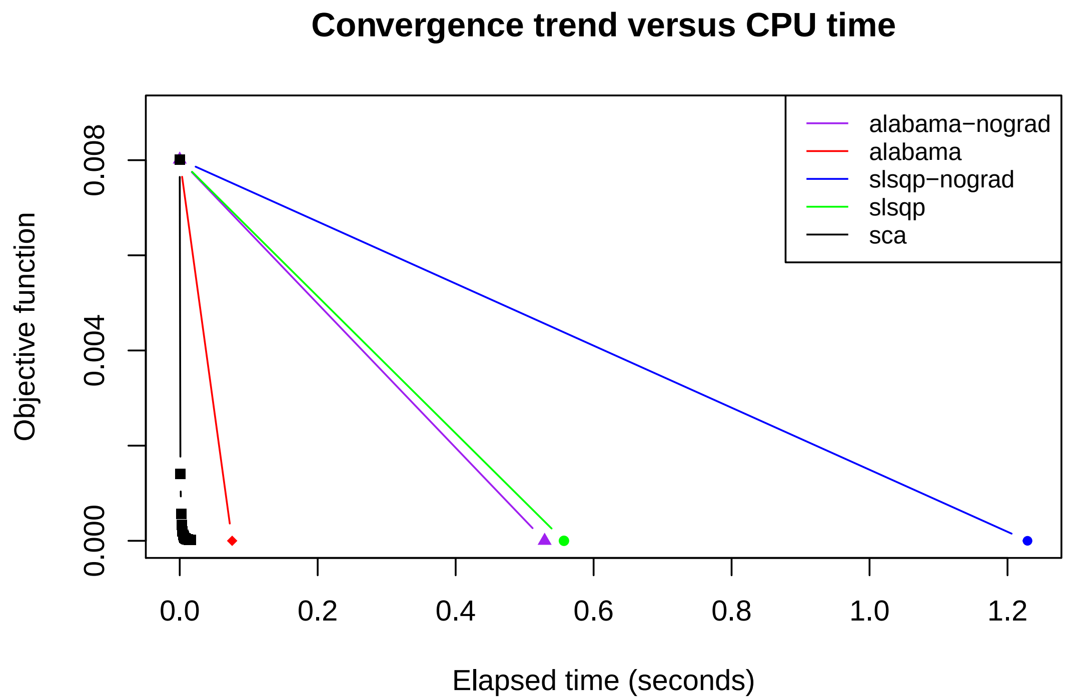
```
      col = "green")
lines(res_sca$elapsed_time, res_sca$obj_fun, type = "b", pch=15, cex=.8,
      col = "black")

legend("topright", legend=c("alabama-nograd",
                            "alabama",
                            "slsqp-nograd",
                            "slsqp",
                            "sca"),
       col=c("purple", "red", "blue", "green", "black"), lty=c(1, 1, 1), cex=0.8)
```



5.2 Experiment: formulation "rc vs b-times-var"

```
res_slsqp <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc vs b-times-var", method = "slsqp")
#> For consistency with the rest of the package the inequality sign may be switched from >= to <= in a
res_slsqp_nograd <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc vs b-times-var",
                                        method = "slsqp", use_gradient = FALSE)
#> For consistency with the rest of the package the inequality sign may be switched from >= to <= in a
res_alabama <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc vs b-times-var", method = "alabama
res_alabama_nograd <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc vs b-times-var",
                                          method = "alabama", use_gradient = FALSE)
res_sca <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc vs b-times-var", method = "sca")

plot(res_slsqp_nograd$elapsed_time, res_slsqp_nograd$obj_fun, type = "b",
     pch=19, cex=.6, col = "blue", xlab = "Elapsed time (seconds)",
     ylab = "Objective function", main = "Convergence trend versus CPU time",
     ylim = c(0, 0.009))
lines(res_alabama$elapsed_time, res_alabama$obj_fun, type = "b", pch=18, cex=.8,
      col = "red")
lines(res_alabama_nograd$elapsed_time, res_alabama_nograd$obj_fun, type = "b", pch=17,
```

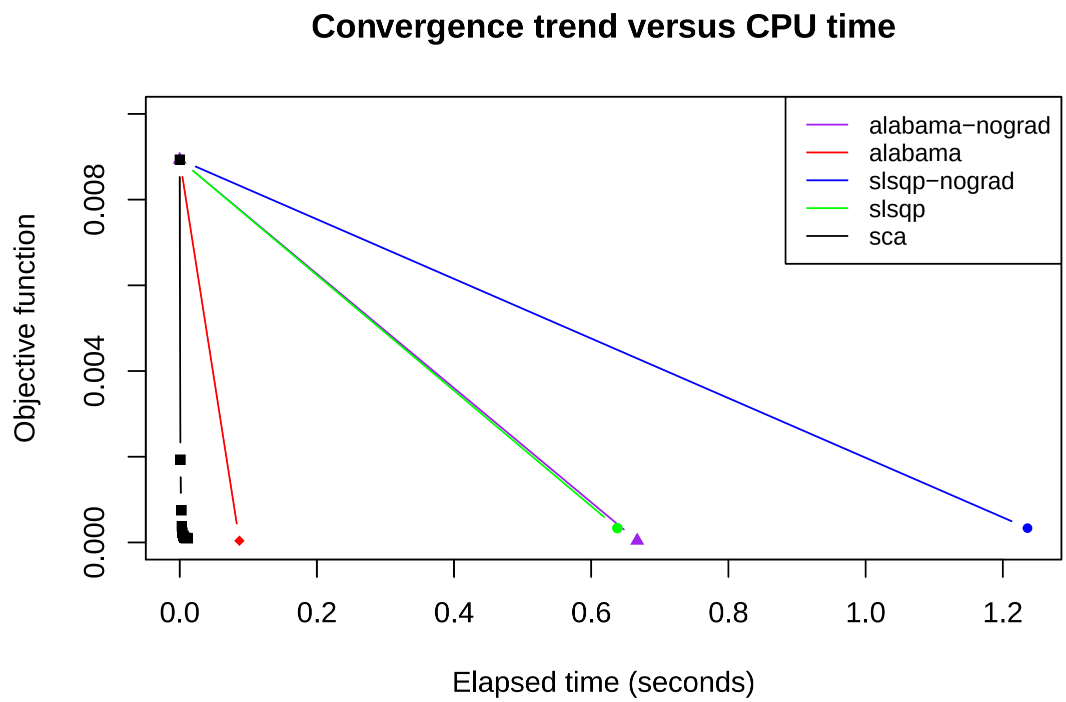
```
      cex=.8, col = "purple")
lines(res_slsqp$elapsed_time, res_slsqp$obj_fun, type = "b", pch=16, cex=.8,
      col = "green")
lines(res_sca$elapsed_time, res_sca$obj_fun, type = "b", pch=15, cex=.8,
      col = "black")

legend("topright", legend=c("alabama-nograd",
                            "alabama",
                            "slsqp-nograd",
                            "slsqp",
                            "sca"),
       col=c("purple", "red", "blue", "green", "black"), lty=c(1, 1, 1), cex=0.8)
```



5.3 Experiment: formulation "rc-over-sd vs b-times-sd"

```
res_slsqp <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-sd vs b-times-sd", method = "sl
#> For consistency with the rest of the package the inequality sign may be switched from >= to <= in a
res_slsqp_nograd <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-sd vs b-times-sd",
                                        method = "slsqp", use_gradient = FALSE)
#> For consistency with the rest of the package the inequality sign may be switched from >= to <= in a
res_alabama <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-sd vs b-times-sd", method = "
res_alabama_nograd <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-sd vs b-times-sd",
                                          method = "alabama", use_gradient = FALSE)
res_sca <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-sd vs b-times-sd", method = "sca"

plot(res_slsqp_nograd$elapsed_time, res_slsqp_nograd$obj_fun, type = "b",
     pch=19, cex=.6, col = "blue", xlab = "Elapsed time (seconds)",
     ylab = "Objective function", main = "Convergence trend versus CPU time",
     ylim = c(0, 0.01))
lines(res_alabama$elapsed_time, res_alabama$obj_fun, type = "b", pch=18, cex=.8,
```

```
      col = "red")
lines(res_alabama_nograd$elapsed_time, res_alabama_nograd$obj_fun, type = "b", pch=17,
      cex=.8, col = "purple")
lines(res_slsqp$elapsed_time, res_slsqp$obj_fun, type = "b", pch=16, cex=.8,
      col = "green")
lines(res_sca$elapsed_time, res_sca$obj_fun, type = "b", pch=15, cex=.8,
      col = "black")

legend("topright", legend=c("alabama-nograd",
                            "alabama",
                            "slsqp-nograd",
                            "slsqp",
                            "sca"),
       col=c("purple", "red", "blue", "green", "black"), lty=c(1, 1, 1), cex=0.8)
```

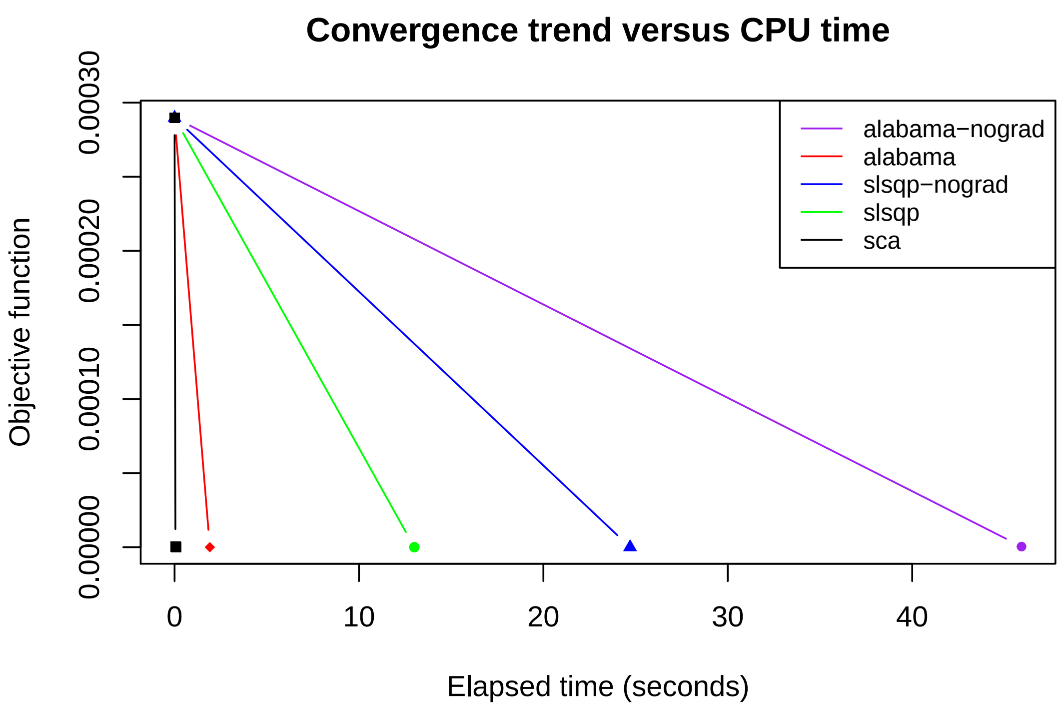


5.4 Experiment with real market data

Now, let's query some real market data using the package `sparseIndexTracking` and check the performance of the different methods.

```
library(sparseIndexTracking)
library(xts)
data(INDEX_2010)
Sigma <- cov(INDEX_2010$X)
N <- nrow(Sigma)
w0 <- rep(1/N, N)

res_slsqp <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-var vs b", method = "slsqp")
#> For consistency with the rest of the package the inequality sign may be switched from >= to <= in a
res_slsqp_nograd <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-var vs b",
                                        method = "slsqp", use_gradient = FALSE)
```

```
#> For consistency with the rest of the package the inequality sign may be switched from >= to <= in a
res_alabama <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-var vs b", method = "alabama")
res_alabama_nograd <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-var vs b",
                                          method = "alabama", use_gradient = FALSE)
res_sca <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-var vs b", method = "sca")

plot(res_alabama_nograd$elapsed_time, res_alabama_nograd$obj_fun, type = "b",
     pch=19, cex=.6, col = "purple", xlab = "Elapsed time (seconds)",
     ylab = "Objective function", main = "Convergence trend versus CPU time")
lines(res_alabama$elapsed_time, res_alabama$obj_fun, type = "b", pch=18, cex=.8,
      col = "red")
lines(res_slsqp_nograd$elapsed_time, res_slsqp_nograd$obj_fun, type = "b", pch=17,
      cex=.8, col = "blue")
lines(res_slsqp$elapsed_time, res_slsqp$obj_fun, type = "b", pch=16, cex=.8,
      col = "green")
lines(res_sca$elapsed_time, res_sca$obj_fun, type = "b", pch=15, cex=.8,
      col = "black")

legend("topright", legend=c("alabama-nograd",
                            "alabama",
                            "slsqp-nograd",
                            "slsqp",
                            "sca"),
       col=c("purple", "red", "blue", "green", "black"), lty=c(1, 1, 1), cex=0.8)
```



It can be noted that the "alabama" and "slsqp" greatly benefit from the additional gradient information. Despite that fact, the "sca" method still performs faster. Additionally, in some cases, the "sca" method attains a better solution than the other methods.

References

[1] F. Spinu, "An algorithm for computing risk parity weights," *SSRN*, 2013.

[2] Boyd S. and L. Vandenberghe, *Convex optimization.* Cambridge University Press, 2009.

[3] Y. Feng and D. P. Palomar, "SCRIP: Successive convex optimization methods for risk parity portfolios design," *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5285–5300, Oct. 2015.