Fast Design of Risk Parity Portfolios

Zé Vinícius and Daniel P. Palomar
Hong Kong University of Science and Technology (HKUST)
2018-12-15

Contents

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.

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 $\mathbf{b} \triangleq (b_1, b_2, ..., b_N)$ (with $\mathbf{1}^T\mathbf{b} = 1$ and $\mathbf{b} \geq \mathbf{0}$) 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

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

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*.

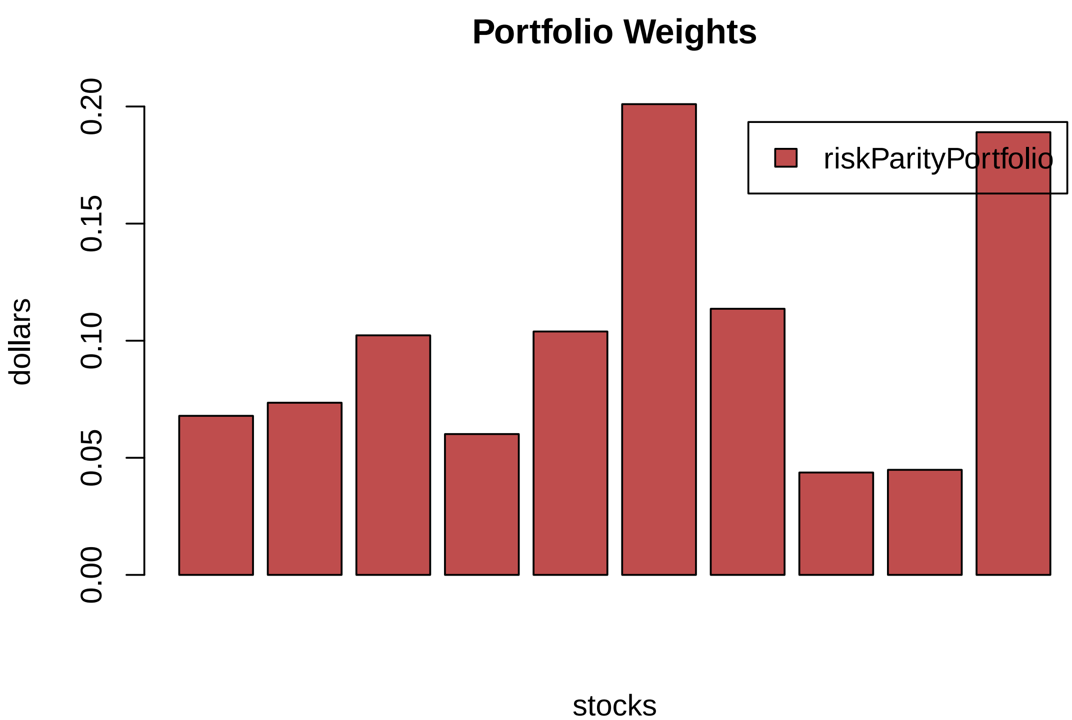
With the goal of designing risk budget portfolios, Spinu proposed in [1] to solve the following convex optimization problem:

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

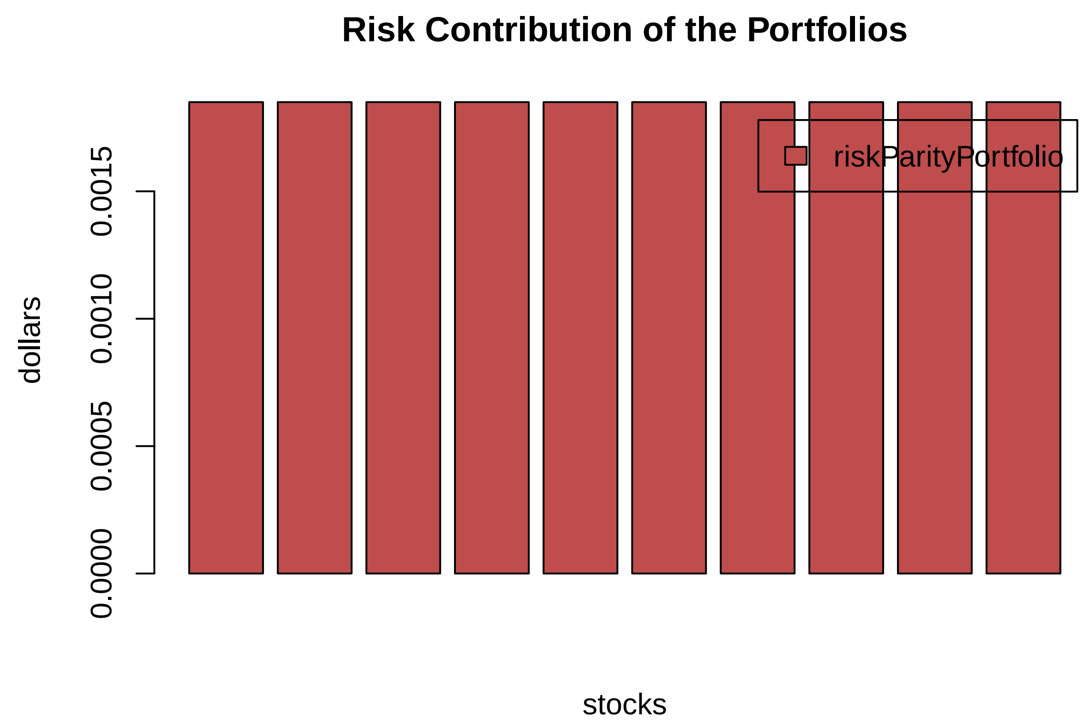
It turns out, as shown in [1], that the unique solution for the optimization problem stated above attains the risk budget requirements in an exact fashion. Such solution can be computed using convex optimization packages, such as CVXR, but faster algorithms such as Newton and cyclical coordinate descent, proposed by [1] and [2], are implemented in this package.

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

```
library(riskParityPortfolio)

# 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"))
```




```
# 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"))
```

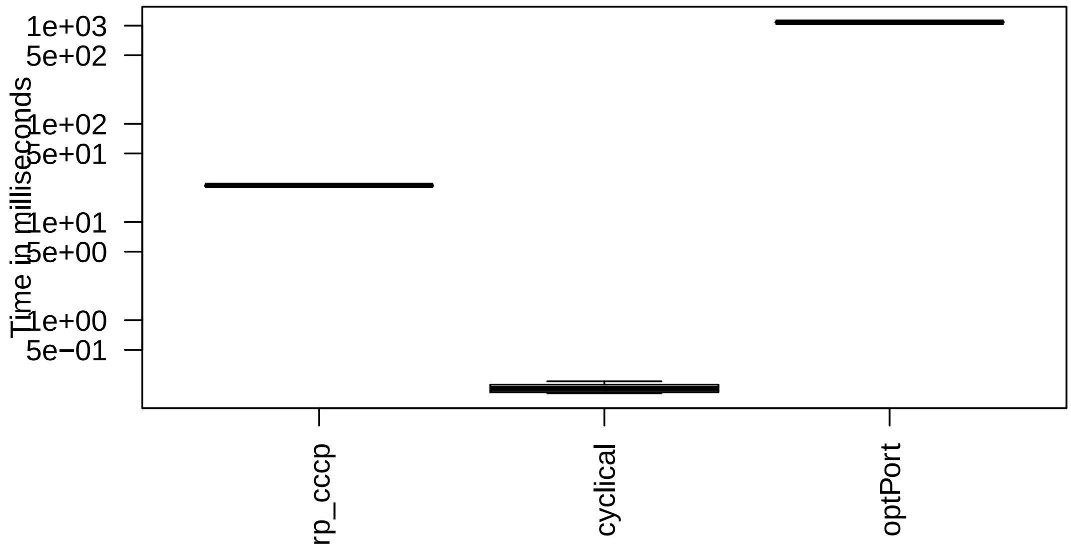


As presented earlier, the risk parity portfolios are designed in such a way as to ensure equal risk contribution from the assests, which can be noted in the chart above.

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

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

N <- 100
V <- matrix(rnorm(N^2), nrow = N)
Sigma <- cov(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)),
          cyclical = riskParityPortfolio:::risk_parity_portfolio_ccd_spinu(Sigma, b, 1e-6, 50),
          optPort = optimalPortfolio(Sigma = Sigma,control = list(type = 'erc', constraint = 'lo')),
          times = 10L)
print(op)
#> Unit: microseconds
#>      expr         min          lq         mean       median          uq
#>   rp_cccp   23232.094   23371.028   23580.2552   23645.6880   23738.529
#>  cyclical     180.348     184.665     203.7327     201.3385     221.015
#>   optPort 1072392.506 1073522.575 1092057.4457 1082183.7920 1096715.886
#>         max neval
```

```
#>    23890.015    10
#>      238.569    10
#>  1181832.693    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 orders of maginitude faster than the interior-point method used by `cccp` and the formulation used by `RiskPortfolios`. We suggest the interested reader to check out Chapter 11 of reference [3] for a thorough explanation on interior-point methods.

2 Modern risk parity portfolio

The design of risk parity portfolios as solved by [1] and [2] 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, mainly because they give rise to nonconvex formulations.

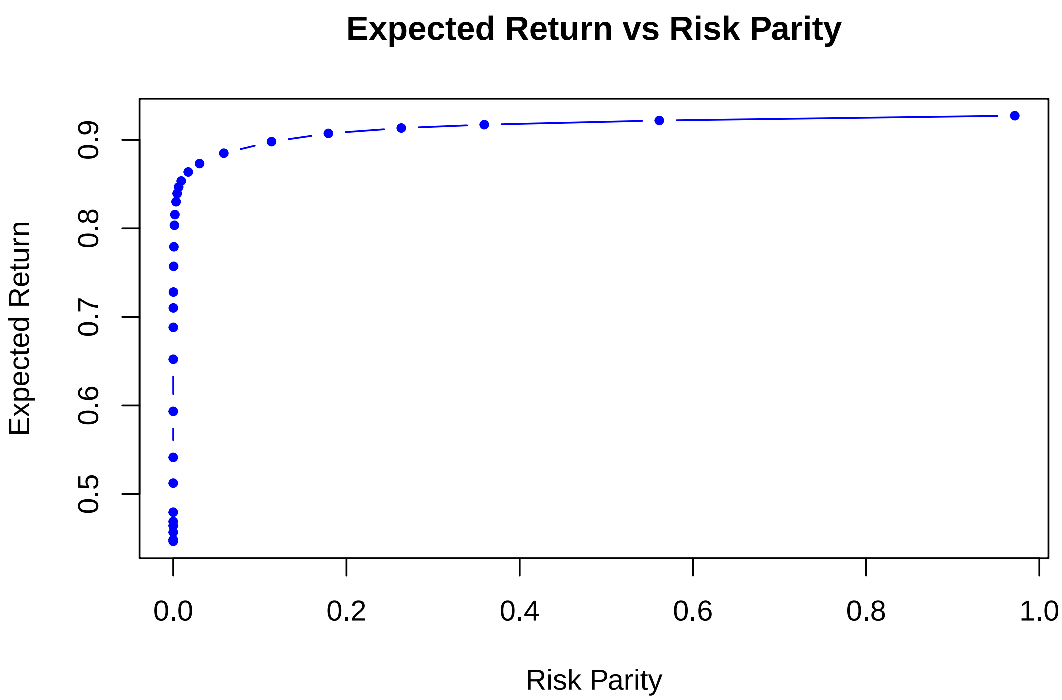
Let's explore, for instance, the effect of including the expected return as an additional objective in the optimization problem. The problem can be formulated as

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

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 or risk parity function, $\mathbf{w}^T\boldsymbol{\mu}$ is the expected return, and λ is a trade-off parameter.

```
N <- 100
V <- matrix(rnorm(N^2), nrow = N)
Sigma <- cov(V)
mu <- runif(N)

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

```r
for (lmd_mu in lmd_sweep) {
  rpp <- riskParityPortfolio(Sigma, mu = mu, lmd_mu = lmd_mu,
                             formulation = "rc-double-index")
  mean_return <- c(mean_return, rpp$mean_return)
  risk_parity <- c(risk_parity, rpp$risk_parity)
}

plot(risk_parity, mean_return, type = "b", pch = 19, cex = .6, col = "blue",
     xlab = "Risk Parity", ylab = "Expected Return",
     ylim = c(min(mean_return), max(mean_return)),
     xlim = c(min(risk_parity), max(risk_parity)),
     main = "Expected Return vs Risk Parity")
```



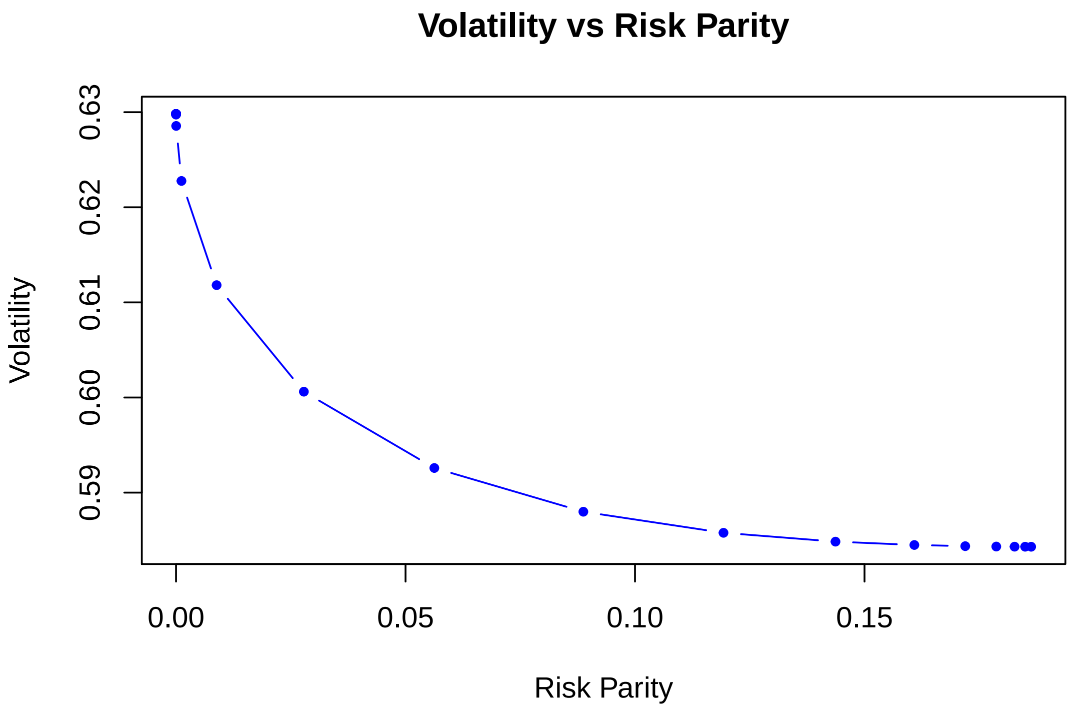
Similarly, the `riskParityPortfolio` package let us include the variance as an objective term, so that the actual optimization problem can be expressed as

$$\begin{array}{ll} \underset{\mathbf{w}}{\text{minimize}} & R(\mathbf{w}) + \lambda_{\text{var}} \mathbf{w}^T \mathbf{\Sigma} \mathbf{w} \\ \text{subject to} & \mathbf{1}^T \mathbf{w} = 1, \mathbf{w} \geq \mathbf{0}, \end{array}$$

In the code, that can be done by passing a positive value to the parameter `lmd_var`. Let's check the following illustrative example that depicts the depence between volatility and risk parity:

```r
N <- 10
Sigma <- diag(c(1:N))
lmd_sweep <- c(10 ^ (seq(-5, 5, .25)))
variance <- c()
risk_parity <- c()

for (lmd_var in lmd_sweep) {
  rpp <- riskParityPortfolio(Sigma, w0 = (1/diag(Sigma)) / sum(1/diag(Sigma)),
                             lmd_var = lmd_var)
```

```
  variance <- c(variance, rpp$variance)
  risk_parity <- c(risk_parity, rpp$risk_parity)
}

volatility <- sqrt(variance)
plot(risk_parity, volatility, type = "b", pch = 19, cex = .6, col = "blue",
     xlab = "Risk Parity", ylab = "Volatility",
     ylim = c(min(volatility), max(volatility)),
     xlim = c(min(risk_parity), max(risk_parity)),
     main = "Volatility vs Risk Parity")
```

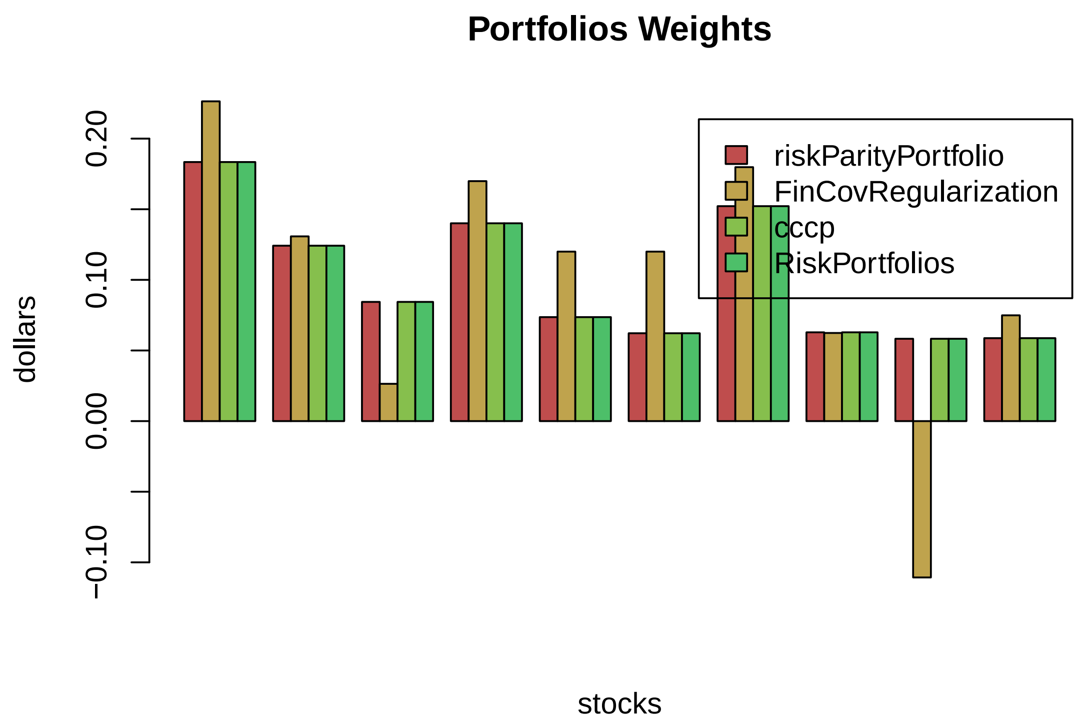


3 Comparison with other packages

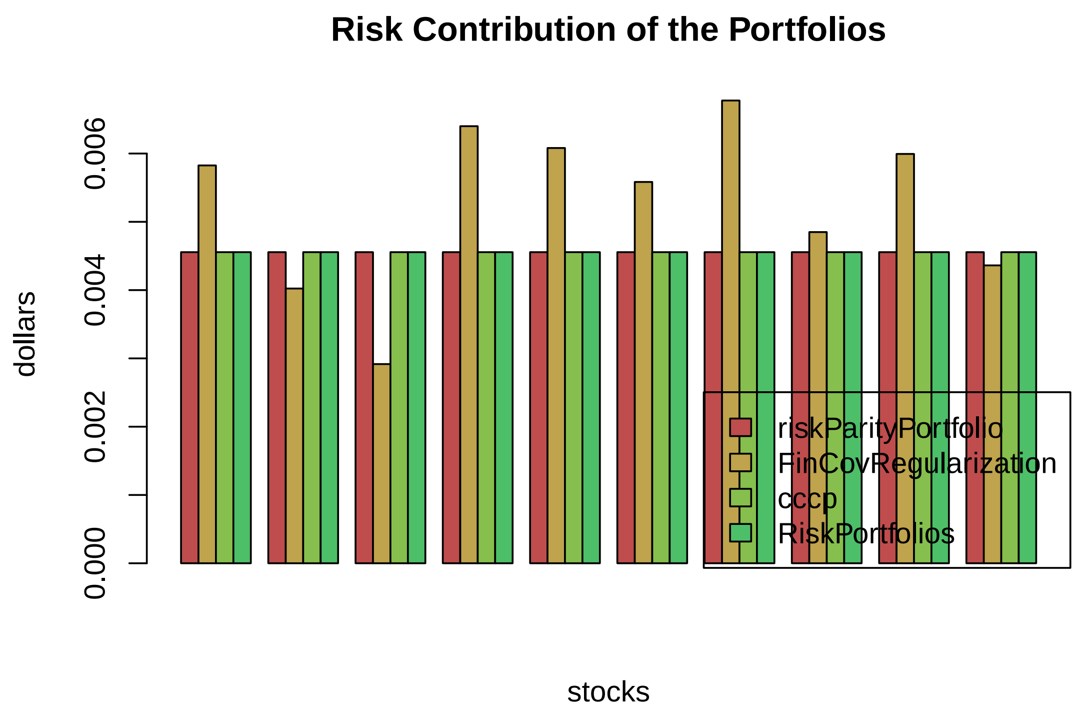
Others R packages with the goal of designing risk parity portfolios do exist, such as `FinCovRegularization`, `cccp`, and `RiskPortfolios`. Let's check how do they perform against `riskParityPortfolio`. (Note that other packages like `FRAPO` use `cccp` under the hood.)

```
library(FinCovRegularization)
library(cccp)
library(RiskPortfolios)

# generate synthetic data
set.seed(123)
N <- 10
#V <- matrix(rnorm(N^2), nrow = N)  # with this, RiskPortfolios::optimalPortfolio() fails
V <- matrix(rnorm(N*(N+N/5)), N+N/5, N)  # with this, FinCovRegularization::RiskParity() fails
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")
riskport_w <- optimalPortfolio(Sigma = Sigma, control = list(type = 'erc',
                                                         constraint = 'lo'))
riskport_risk_contribution <- c(riskport_w * (Sigma %*% riskport_w))
fincov_w <- RiskParity(Sigma)
fincov_risk_contribution <- c(fincov_w * (Sigma %*% fincov_w))
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, riskport_w),
        main = "Portfolios Weights", xlab = "stocks", ylab = "dollars",
        beside = TRUE, col = rainbow8equal[1:4],
        legend = c("riskParityPortfolio", "FinCovRegularization", "cccp",
                   "RiskPortfolios"))
```




```r
barplot(rbind(rpp$risk_contribution, fincov_risk_contribution, cccp_risk_contribution,
              riskport_risk_contribution),
        main = "Risk Contribution of the Portfolios", xlab = "stocks", ylab = "dollars",
        beside = TRUE, col = rainbow8equal[1:4],
        legend = c("riskParityPortfolio", "FinCovRegularization", "cccp",
                   "RiskPortfolios"),
        args.legend = list(x = "bottomright"))
```



Depending on the condition number of the covariance matrix, we found that the packages `FinCovRegularization` and `RiskPortfolios` may fail unexpectedly. Apart from that, the other functions perform the same.

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 [4]. 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":

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

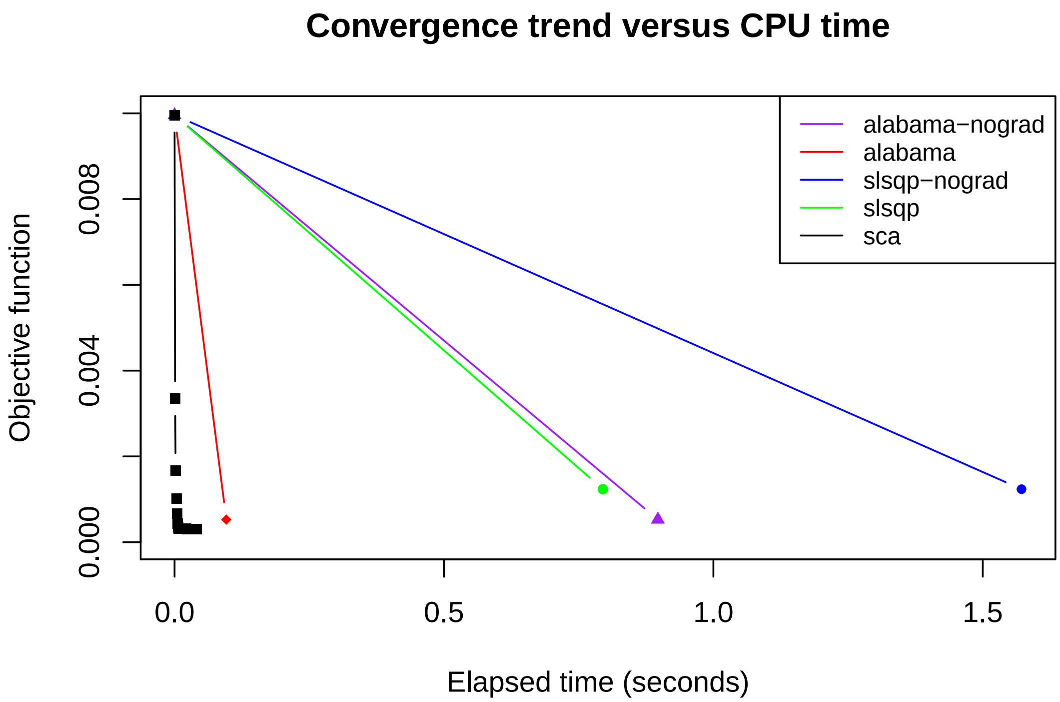
5 Appendix II: Computational time

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")
res_slsqp_nograd <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-var vs b",
                                        method = "slsqp", use_gradient = FALSE)
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,
```

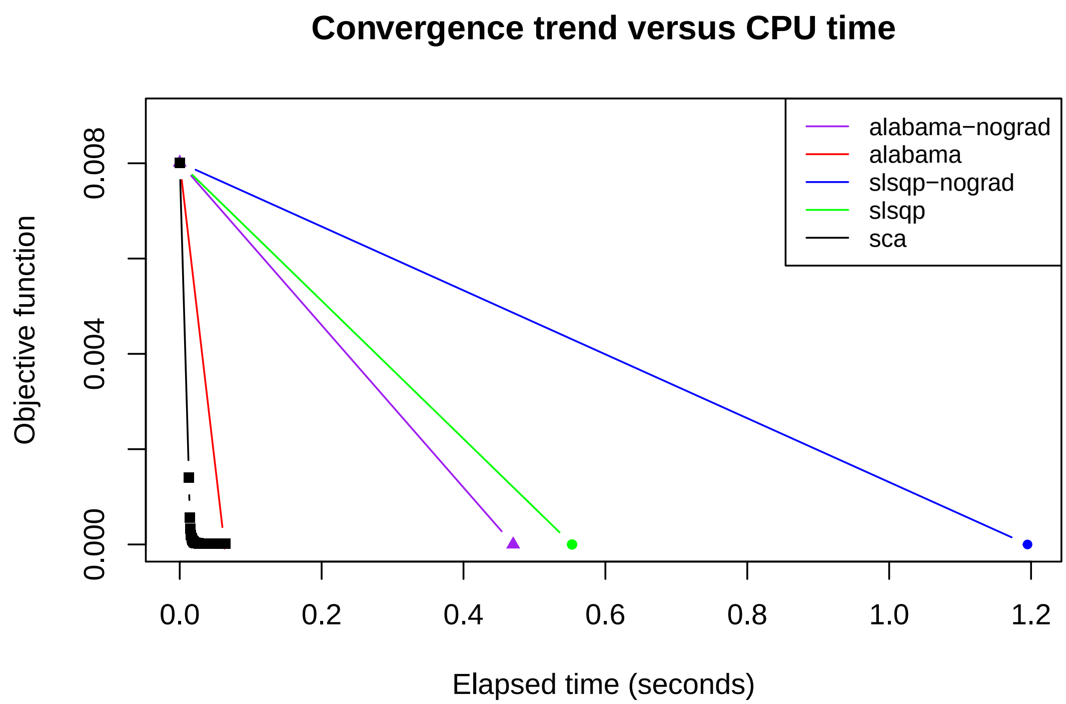
```
      col = "green")
lines(res_sca$elapsed_time, res_sca$obj_fun, type = "b", pch=15, cex=.8,
      col = "black")

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, bg = "white")
```



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

```
res_slsqp <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc vs b-times-var",
                                 method = "slsqp")
res_slsqp_nograd <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc vs b-times-var",
                                        method = "slsqp", use_gradient = FALSE)
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")
```

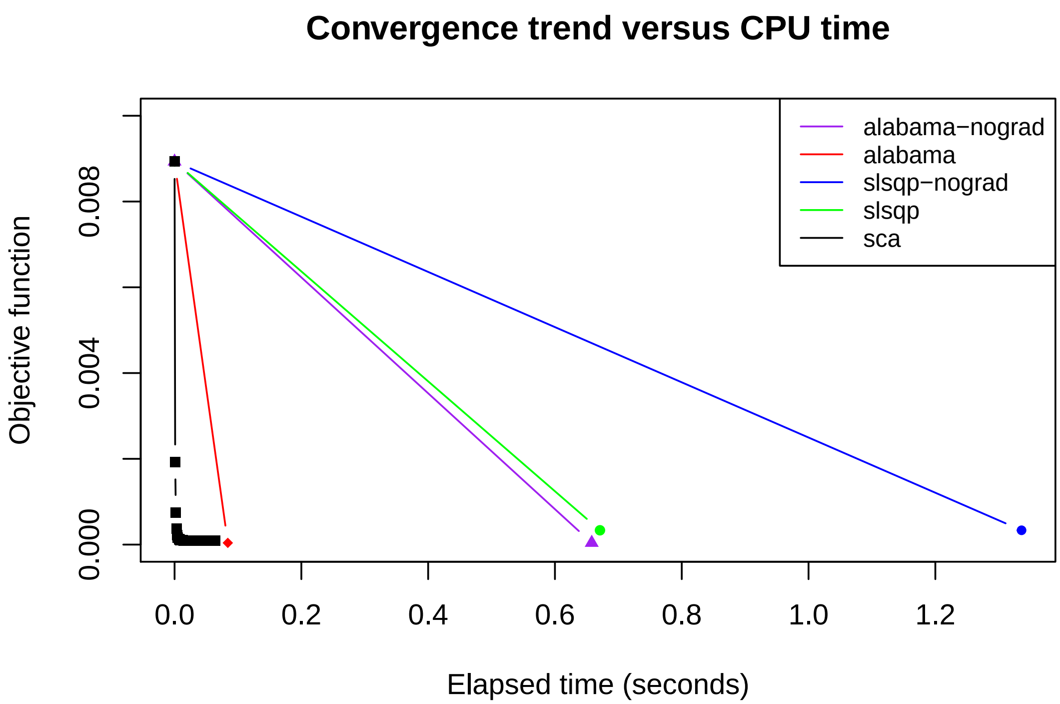
```
lines(res_alabama_nograd$elapsed_time, res_alabama_nograd$obj_fun, type = "b", pch=17,
      cex=.8, col = "purple")
lines(res_slsqp$elapsed_time, res_slsqp$obj_fun, type = "b", pch=16, cex=.8,
      col = "green")
lines(res_sca$elapsed_time, res_sca$obj_fun, type = "b", pch=15, cex=.8,
      col = "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 = "slsqp")
res_slsqp_nograd <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-sd vs b-times-sd",
                                        method = "slsqp", use_gradient = FALSE)
res_alabama <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-sd vs b-times-sd",
                                   method = "alabama")
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",
```

```r
     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, bg = "white")
```

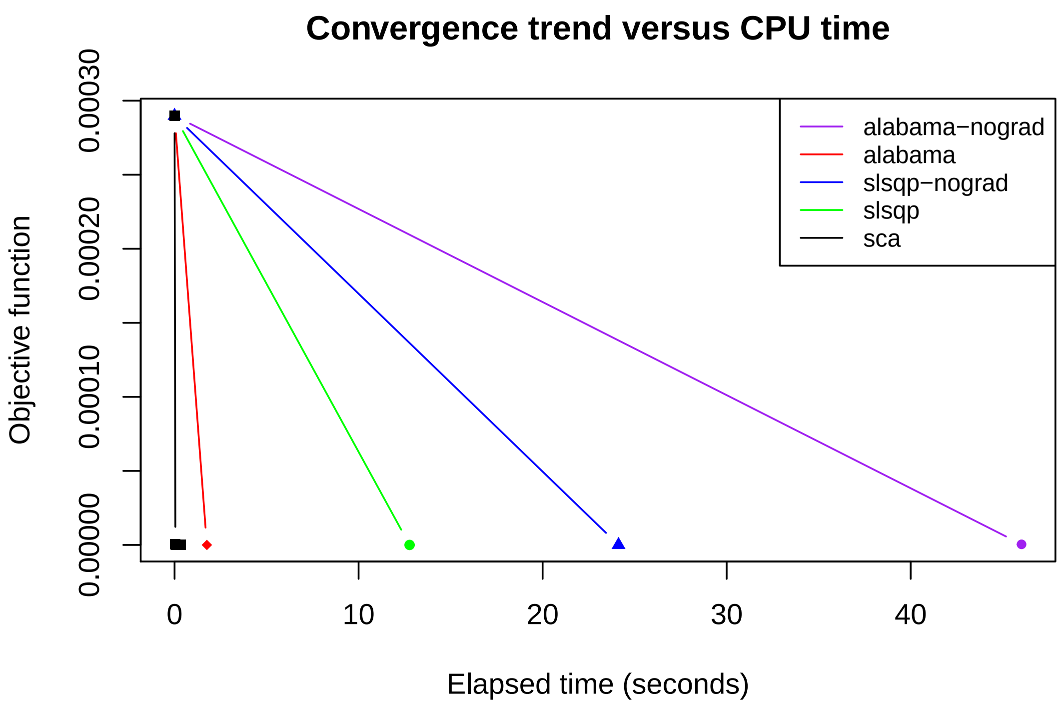


5.4 Experiment with real market data

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

```r
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")
```

```r
res_slsqp_nograd <- riskParityPortfolio(Sigma, w0 = w0, formulation = "rc-over-var vs b",
                                        method = "slsqp", use_gradient = FALSE)
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,
       bg = "white")
```



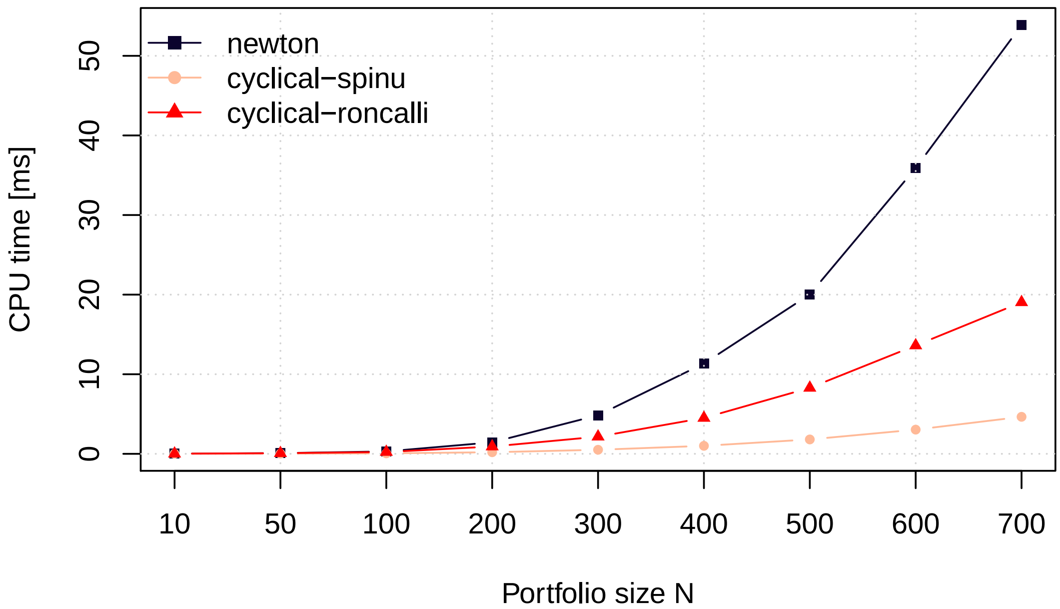
It can be noted that the "alabama" and "slsqp" greatly benefit from the additional gradient information. Despite that fact, the "sca" method still performs faster. Additionally, in some cases, the "sca" method attains a better solution than the other methods.

6 Appendix III: Design of high dimensional risk parity portfolio

In order to efficiently design high dimensional portfolios that follows the risk parity criterion, we implement the cyclical coordinate descent algorithm proposed by [2]. In brief, this algorithm optimizes for one portfolio weight at a time while leaving the rest fixed. By iteratively applying this procedure it is possible to show that the sequence of estimations reaches the global minimum of the convex function in roughly $O(N_{iter} \times N^2)$.

The plot below illustrates the computational scaling of both Newton and cyclical algorithms. Note that the cyclical algorithm is implemented for two different formulations used by [1] and [2], respectively. Nonetheless, they output the same solution, as they should.

```
library(microbenchmark)
library(riskParityPortfolio)

sizes <- c(10, 50, 100, 200, 300, 400, 500, 600, 700)
size_seq <- c(1:length(sizes))
times <- matrix(0, 3, length(sizes))
for (i in size_seq) {
  V <- matrix(rnorm(1000 * sizes[i]), nrow = sizes[i])
  Sigma <- V %*% t(V)
  bench <- microbenchmark(
            newton = riskParityPortfolio(Sigma, method_init="newton"),
            cyclical_spinu = riskParityPortfolio(Sigma, method_init="cyclical-spinu"),
            cyclical_roncalli = riskParityPortfolio(Sigma, method_init="cyclical-roncalli"),
            times = 10L, unit = "ms", control = list(order = "inorder", warmup = 4))
  times[1, i] <- median(bench$time[c(TRUE, FALSE, FALSE)] / 10 ^ 6)
  times[2, i] <- median(bench$time[c(FALSE, TRUE, FALSE)] / 10 ^ 6)
  times[3, i] <- median(bench$time[c(FALSE, FALSE, TRUE)] / 10 ^ 6)
}

colors <- c("#0B032D", "#FFB997", "red")
plot(size_seq, times[1,], type = "b", pch=15, cex=.75, col = colors[1],
     xlab = "Portfolio size N", ylab = "CPU time [ms]", xaxt = "n")
grid()
lines(size_seq, times[2,], type = "b", pch=16, cex=.75, col = colors[2])
lines(size_seq, times[3,], type = "b", pch=17, cex=.75, col = colors[3])
axis(side = 1, at = size_seq, labels = sizes)
legend("topleft", legend = c("newton", "cyclical-spinu", "cyclical-roncalli"),
       col=colors, pch=c(15, 16, 17), lty=c(1, 1, 1), bty="n")
```



References

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

[2] T. Griveau-Billion, J. Richard, and T. Roncalli, “A fast algorithm for computing high-dimensional risk parity portfolios,” *ArXiv preprint*, 2013.

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

[4] 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.