Risk-Parity Portfolio with R

MAFS6010R - Portfolio Optimization with R MSc in Financial Mathematics Fall 2018-19, HKUST, Hong Kong

Prof. Daniel P. Palomar

Hong Kong University of Science and Technology (HKUST)

- Loading market data
- Warm-up: Markowitz portfolio
    - Lack of diversification
    - Sensitivity to parameters
- Risk-parity portfolio
    - Formulations
    - Algorithms
        - Closed-form solution for diagonal case
        - Convex formulation
        - General formulations with nonlinear solvers
        - R packages
        - SCA method
    - Formulation with expected return
    - Sensitivity to parameters
- Conclusion

This R session will introduce the risk-parity portfolio and compare with Markowitz portfolio.

The R package **riskParityPortfolio**, available in GitHub (https://github.com/dppalomar/riskParityPortfolio), is recommended.

(Useful R links: Cookbook R (http://www.cookbook-r.com/), Quick-R (https://www.statmethods.net/), R documentation (https://www.rdocumentation.org/), CRAN (https://cran.r-project.org/), METACRAN (https://r-pkg.org/).)

Loading market data

In this section, we will divide the stock market data into a training part (for the estimation of the expected return $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, and subsequent portfolio design) and a test part (for the out-of-sample performance evaluation).

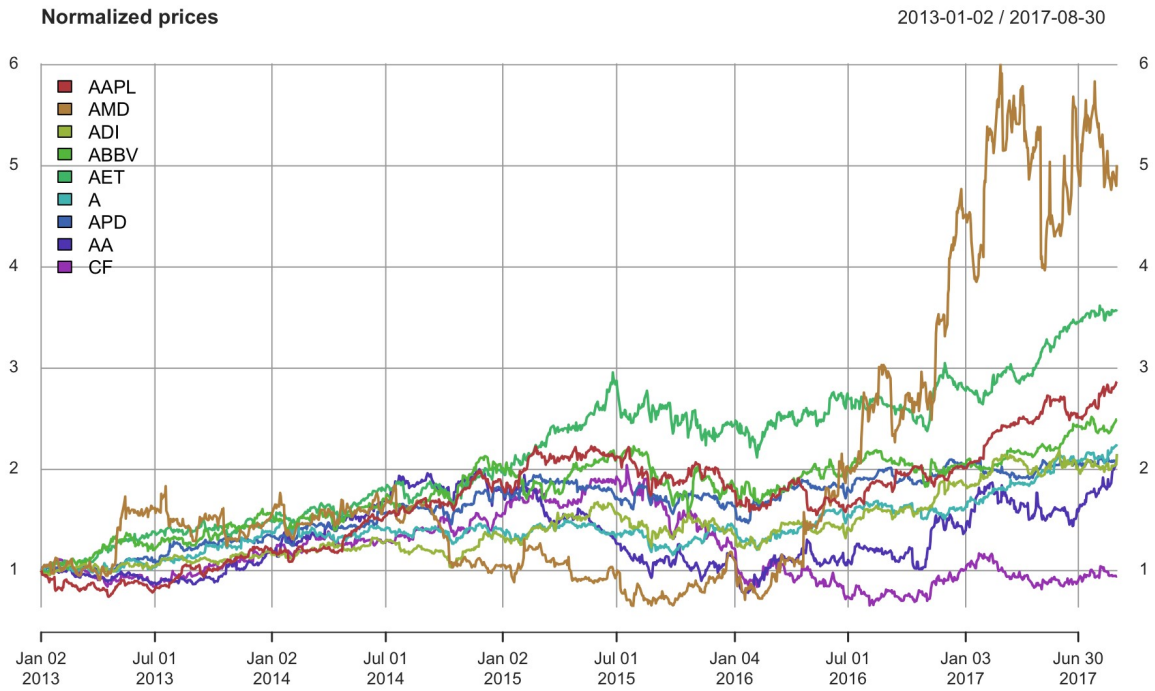
We start by loading some stock market data:

```
library(xts)
library(quantmod)
library(PerformanceAnalytics)

# set begin-end date and stock namelist
begin_date <- "2013-01-01"
end_date <- "2017-08-31"
stock_namelist <- c("AAPL", "AMD", "ADI",  "ABBV", "AET", "A",  "APD", "AA","CF")

# download data from YahooFinance
prices <- xts()
for (stock_index in 1:length(stock_namelist))
  prices <- cbind(prices, Ad(getSymbols(stock_namelist[stock_index],
                                        from = begin_date, to = end_date, auto.assign = FALSE)))
colnames(prices) <- stock_namelist
indexClass(prices) <- "Date"
str(prices)
#> An 'xts' object on 2013-01-02/2017-08-30 containing:
#>   Data: num [1:1175, 1:9] 56.1 55.4 53.9 53.6 53.7 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : NULL
#>   ..$ : chr [1:9] "AAPL" "AMD" "ADI" "ABBV" ...
#>   Indexed by objects of class: [Date] TZ: UTC
#>   xts Attributes:
#>  NULL
head(prices)
#>                AAPL  AMD      ADI     ABBV      AET        A      APD
#> 2013-01-02 56.11887 2.53 37.90372 28.62027 43.32114 28.16434 67.74826
#> 2013-01-03 55.41053 2.49 37.29209 28.38393 42.40319 28.26521 67.51154
#> 2013-01-04 53.86707 2.59 36.62878 28.02536 42.59989 28.82338 68.41895
#> 2013-01-07 53.55021 2.67 36.74078 28.08241 43.23684 28.61492 68.35583
#> 2013-01-08 53.69434 2.67 36.36173 27.47122 41.75045 28.38627 68.48207
#> 2013-01-09 52.85514 2.63 36.26697 27.62606 42.31490 29.15291 69.40527
#>                  AA       CF
#> 2013-01-02 20.62187 29.72212
#> 2013-01-03 20.80537 29.58158
#> 2013-01-04 21.24121 30.24417
#> 2013-01-07 20.87419 30.13087
#> 2013-01-08 20.87419 29.68914
#> 2013-01-09 20.82831 30.72749
tail(prices)
#>                AAPL   AMD      ADI     ABBV      AET        A      APD
#> 2017-08-23 157.5971 12.48 77.08375 69.09010 154.8036 62.21310 140.5500
#> 2017-08-24 156.8977 12.50 77.27874 69.66007 154.3686 62.12389 140.6178
#> 2017-08-25 157.4789 12.43 76.98628 70.01749 154.3192 62.34195 141.3730
#> 2017-08-28 159.0649 12.23 77.44447 70.82896 154.7443 62.89698 141.1504
#> 2017-08-29 160.4835 12.15 77.55172 71.37959 154.6356 62.92671 140.7534
#> 2017-08-30 160.9169 12.67 81.61696 71.40857 155.1101 63.33307 140.8889
#>                AA       CF
#> 2017-08-23 41.06 28.08939
#> 2017-08-24 41.34 28.18649
#> 2017-08-25 41.21 28.13794
#> 2017-08-28 42.17 28.18649
#> 2017-08-29 43.00 27.99230
#> 2017-08-30 43.09 28.09910

# compute log-returns and linear returns
X_log <- diff(log(prices))[-1]
X_lin <- (prices/lag(prices) - 1)[-1]

# or alternatively...
```

```
X_log <- CalculateReturns(prices, "log")[-1]
X_lin <- CalculateReturns(prices)[-1]

N <- ncol(X_log)  # number of stocks
T <- nrow(X_log)  # number of days
```

We can take a look at the prices of the selected stocks:

```
plot(prices/rep(prices[1, ], each = nrow(prices)), col = rainbow10equal, legend.loc = "topleft",
     main = "Normalized prices")
```



We now divide the data into a training set and test set:

```
# split data into training and set data
T_trn <- round(0.7*T)  # 70% of data
X_log_trn <- X_log[1:T_trn, ]
X_log_tst <- X_log[(T_trn+1):T, ]
X_lin_trn <- X_lin[1:T_trn, ]
X_lin_tst <- X_lin[(T_trn+1):T, ]
```

We can now use the training set to obtain the sample estimates from the returns $\mathbf{x}_t$ (i.e., sample means and sample covariance matrix) as

$$\hat{\boldsymbol{\mu}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_t$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{T-1}\sum_{t=1}^{T}(\mathbf{x}_t - \hat{\boldsymbol{\mu}})(\mathbf{x}_t - \hat{\boldsymbol{\mu}})^T$$

```
mu <- colMeans(X_log_trn)
Sigma <- cov(X_log_trn)
```

Warm-up: Markowitz portfolio

The mean-variance Markowitz portfolio with no shorting is

$$\begin{aligned} \underset{\mathbf{w}}{\text{maximize}} \quad & \boldsymbol{\mu}^T\mathbf{w} - \lambda\mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w} \\ \text{subject to} \quad & \mathbf{1}^T\mathbf{w} = 1 \\ & \mathbf{w} \geq \mathbf{0}. \end{aligned}$$

For completeness, we can also consider the Global Minimum Variance Portfolio (GMVP), which doesn't make use of $\boldsymbol{\mu}$:

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w} \\ \text{subject to} \quad & \mathbf{1}^T\mathbf{w} = 1 \\ & \mathbf{w} \geq \mathbf{0}. \end{aligned}$$

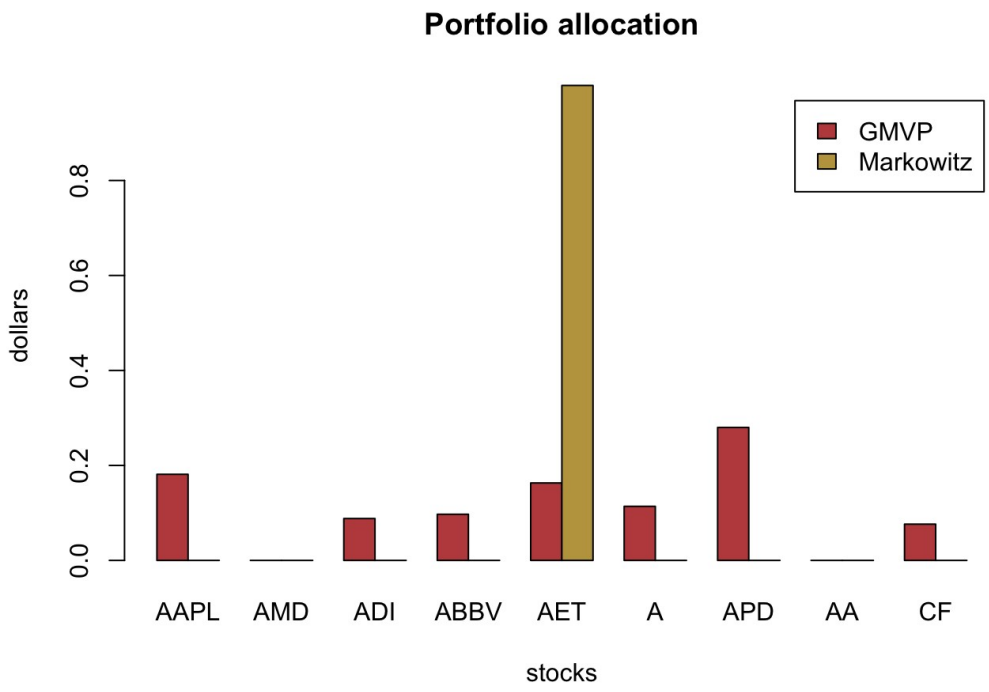
We can compute the optimal $\mathbf{w}$ with a solver (the closed-form solution does not exist for $\mathbf{w} \geq \mathbf{0}$):

```
library(CVXR)

portolioMarkowitz <- function(mu, Sigma, lmd = 0.5) {
  w <- Variable(nrow(Sigma))
  prob <- Problem(Maximize(t(mu) %*% w - lmd*quad_form(w, Sigma)),
                  constraints = list(w >= 0, sum(w) == 1))
  result <- solve(prob)
  return(as.vector(result$getValue(w)))
}

portolioGMVP <- function(Sigma) {
  w <- Variable(nrow(Sigma))
  prob <- Problem(Minimize(quad_form(w, Sigma)),
                  constraints = list(w >= 0, sum(w) == 1))
  result <- solve(prob)
  return(as.vector(result$getValue(w)))
}

w_Markowitz <- portolioMarkowitz(mu, Sigma)
w_GMVP <- portolioGMVP(Sigma)
```

Lack of diversification

We can now observe the allocations of the portfolios:

```r
# put together all portfolios
w_all <- cbind(w_GMVP, w_Markowitz)
rownames(w_all) <- colnames(X_lin)
colnames(w_all) <- c("GMVP", "Markowitz")
round(w_all, digits = 2)
#>      GMVP Markowitz
#> AAPL 0.18         0
#> AMD  0.00         0
#> ADI  0.09         0
#> ABBV 0.10         0
#> AET  0.16         1
#> A    0.11         0
#> APD  0.28         0
#> AA   0.00         0
#> CF   0.08         0
barplot(t(w_all),
        main = "Portfolio allocation", xlab = "stocks", ylab = "dollars", beside = TRUE,
        legend = colnames(w_all), col = rainbow8equal[1:2])
```



Indeed, we can clearly see that the mean-variance Markowitz portfolio concentrates all the budget in one single asset! The GMVP is much more diversified.

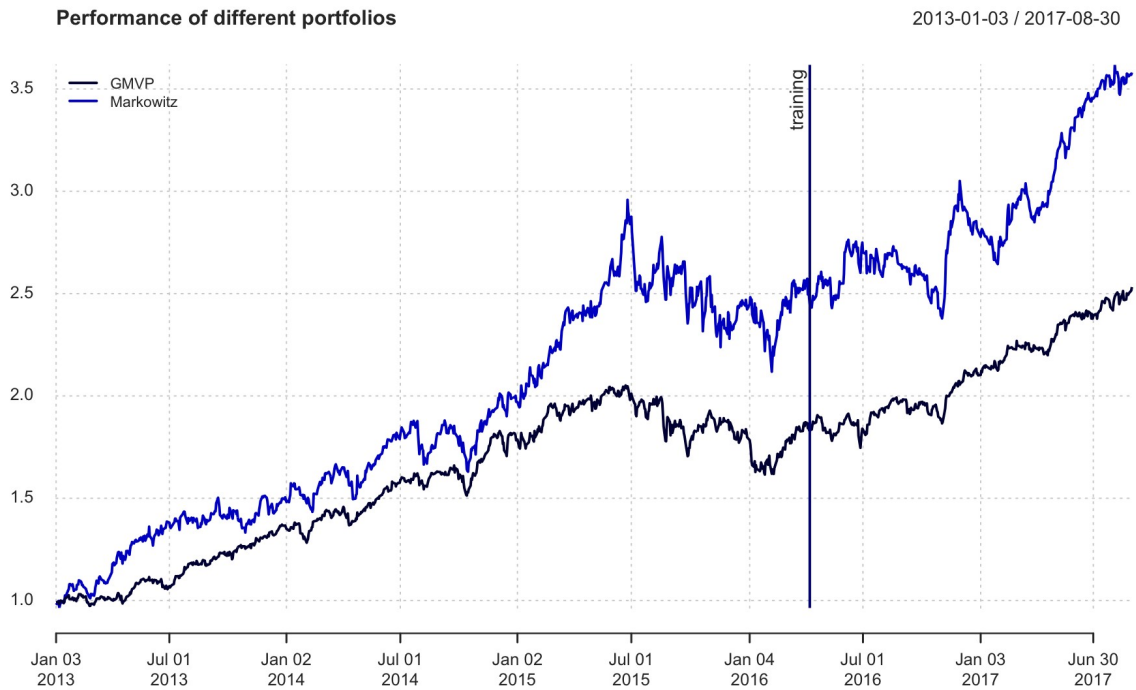
Then we can compare the performance (in sample vs out-of-sample):

```
# compute returns of all portfolios
ret_all <- xts(X_lin %*% w_all, index(X_lin))
ret_all_trn <- ret_all[1:T_trn, ]
ret_all_tst <- ret_all[-c(1:T_trn), ]

# performance
table.AnnualizedReturns(ret_all_trn)
#>                            GMVP Markowitz
#> Annualized Return         0.2041    0.3154
#> Annualized Std Dev        0.1622    0.2503
#> Annualized Sharpe (Rf=0%) 1.2584    1.2599
table.AnnualizedReturns(ret_all_tst)
#>                            GMVP Markowitz
#> Annualized Return         0.2604    0.3138
#> Annualized Std Dev        0.1234    0.1860
#> Annualized Sharpe (Rf=0%) 2.1092    1.6875
```

We can see that the mean-variance Markowitz portfolio performs even worse than the GMVP in the out-of-sample (the in-sample Sharpe ratio is approximately the same though).

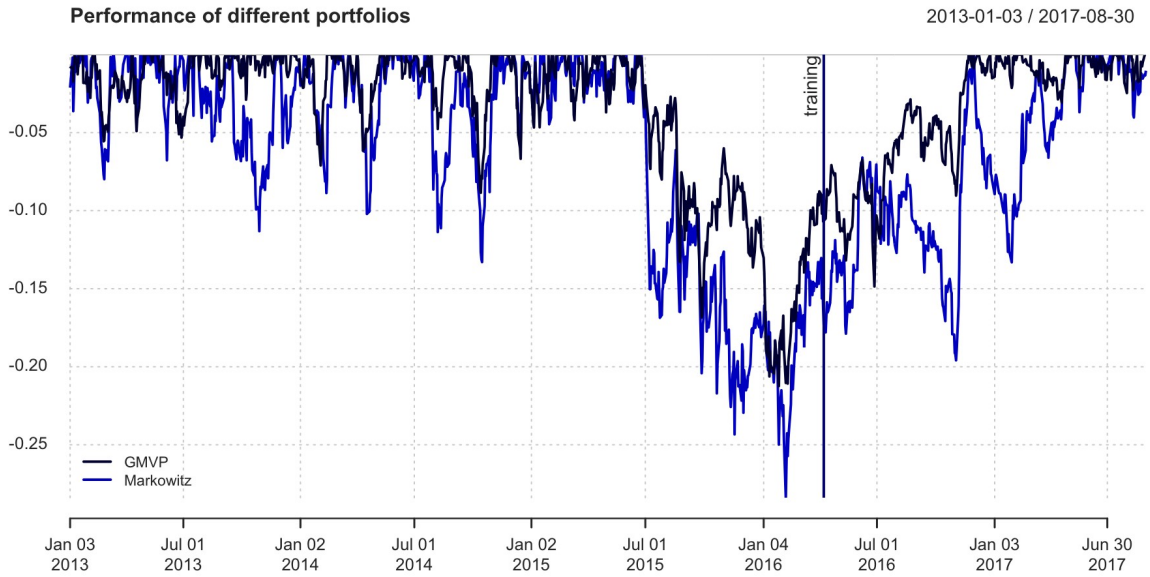
Let us plot the cumulative PnL over time:

```
{ chart.CumReturns(ret_all, main = "Performance of different portfolios",
                   wealth.index = TRUE, legend.loc = "topleft", colorset = rich8equal)
  addEventLines(xts("training", index(X_lin[T_trn])), srt=90, pos=2, lwd = 2, col = "darkblue") }
```



The cum PnL may seem contradictory at first because the mean-variance portfolio seems to be doing much better than the GMVP. This is however a visual effect. The drawdown is very instructive:

```
{ chart.Drawdown(ret_all, main = "Performance of different portfolios",
                 legend.loc = "bottomleft", colorset = rich8equal)
  addEventLines(xts("training", index(X_lin[T_trn])), srt=90, pos=2, lwd = 2, col = "darkblue") }
```



We can see that the drawdown of the mean-variance portfolio is indeed much worse than that of the GMVP.

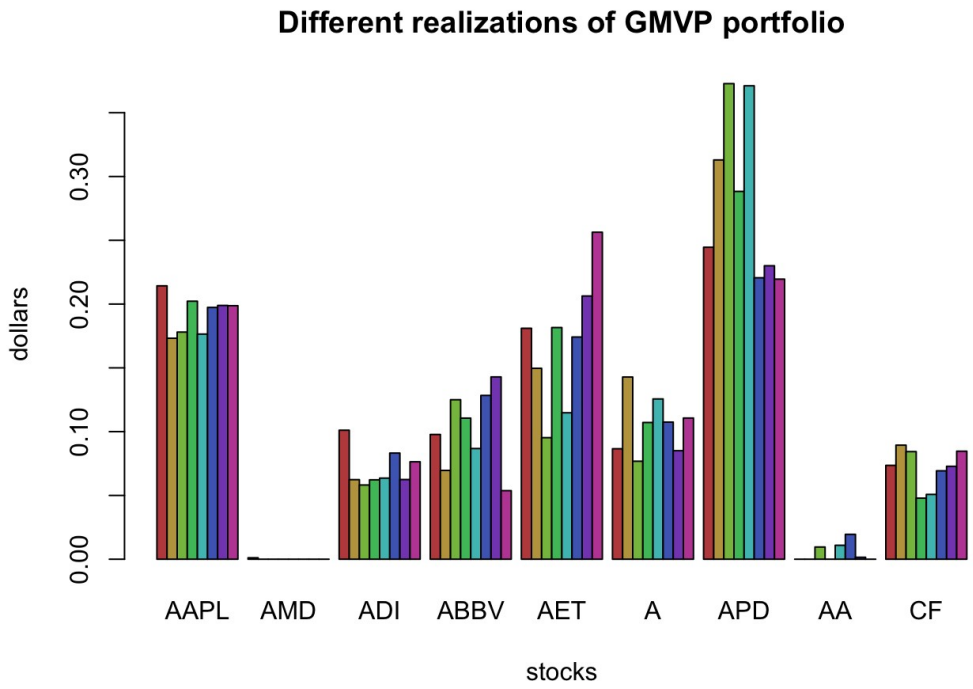
Sensitivity to parameters

To study the sensitivity with respect to the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we will design multiple portfolios based on different samples from the training set:

```
w_GMVP_acc <- w_Markowitz_acc <- NULL
for (i in 1:8) {
  # sample means with random samples
  idx <- sample(1:T_trn, T_trn/2)
  mu_ <- colMeans(X_log_trn[idx, ])
  Sigma_ <- cov(X_log_trn[idx, ])

  # design portfolios
  w_Markowitz_acc <- cbind(w_Markowitz_acc, portolioMarkowitz(mu_, Sigma_))
  w_GMVP_acc <- cbind(w_GMVP_acc, portolioGMVP(Sigma_))
}
rownames(w_GMVP_acc) <- rownames(w_Markowitz_acc) <- colnames(X_lin)
```

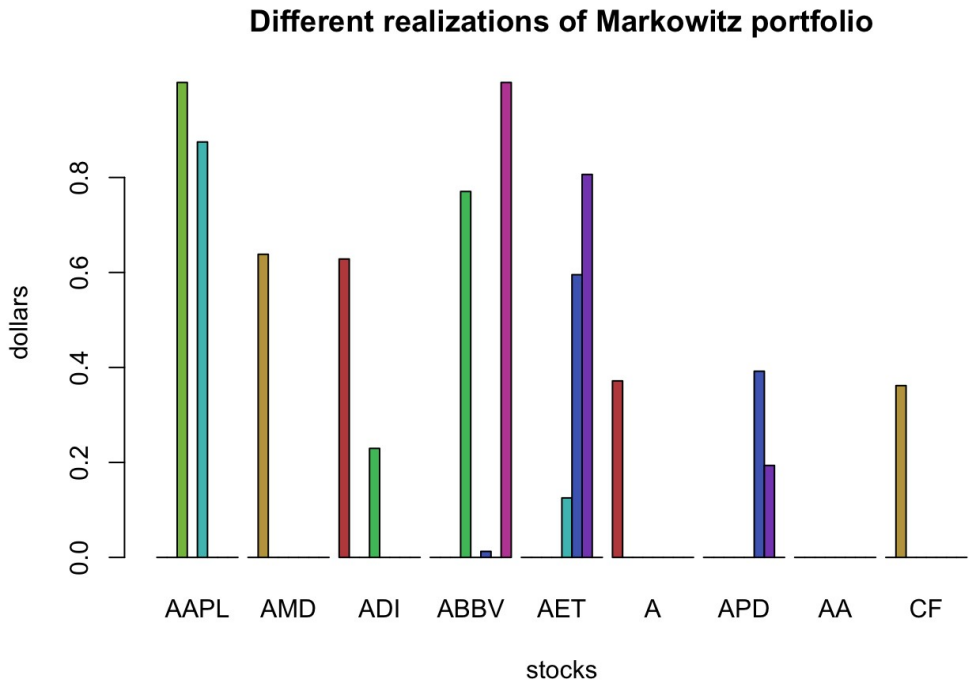
The GMVP is not very sensitive since all the realizations have a similar allocation:

```
barplot(t(w_GMVP_acc),
        main = "Different realizations of GMVP portfolio", xlab = "stocks", ylab = "dollars",
        beside = TRUE, col = rainbow8equal)
```



On the other hand, the mean-variance portfolio is highly sensitive. So sensitive that the portfolio at each realization is totally different!:

```
barplot(t(w_Markowitz_acc),
        main = "Different realizations of Markowitz portfolio", xlab = "stocks", ylab = "dollars",
        beside = TRUE, col = rainbow8equal)
```



Of course the reason for this very distinct behavior is that the sensitivity w.r.t. $\boldsymbol{\mu}$ is much greater than w.r.t. $\boldsymbol{\Sigma}$ (also, the estimation error in the former is larger than in the latter).

Risk-parity portfolio

As we have seen, the Markowitz porftolio, while it started the field of modern portfolio theory in 1952, has not been embraced by practitioners because it does not diversify the risk and concentrates in a few assets, also it is too sensitive to the parameters (particularly to $\boldsymbol{\mu}$).

We now consider the risk-parity portfolio that precisely aims at diversifying the risk contribution and it also happens to be less sensitive to the parameters (because in its original formulation $\boldsymbol{\mu}$ is not used).

Detailed information on the risk-parity portfolio can be found in the following references:

T. Roncalli, *Introduction to Risk Parity and Budgeting*. CRC, 2013.

F. Spinu, "An algorithm for computing risk parity weights," *SSRN*, 2013. [Online]. Available: https://ssrn.com/abstract=2297383 (https://ssrn.com/abstract=2297383).

T. Griveau-Billion, J.-C. Richard, and T. Roncalli, "A fast algorithm for computing high-dimensional risk parity portfolios," *SSRN*, 2013. [Online]. Available: https://ssrn.com/abstract=2325255 (https://ssrn.com/abstract=2325255).

Yiyong Feng and Daniel P. Palomar, *A Signal Processing Perspective on Financial Engineering* (http://www.ece.ust.hk/~palomar/Publications_files/2016/Feng&Palomar-FnT2016.pdf). Foundations and Trends in Signal Processing, Now Publishers, 2016.

Yiyong Feng and Daniel P. Palomar, "SCRIP: Successive Convex Optimization Methods for Risk Parity Portfolio Design (http://www.ece.ust.hk/~palomar/Publications_files/2015/FengPalomar-TSP2015%20-%20risk_parity_portfolio.pdf)," *IEEE Trans. on Signal Processing*, vol. 63, no. 19, pp. 5285-5300, Oct. 2015.

Formulations

In its simplest version, the risk-parity portfolio aims at achieving parity of the risk contributions from the different assets:

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

Assuming 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

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

When $\boldsymbol{\Sigma}$ is not diagonal, the previous expression can still be used (albeit being suboptimal) and it is called *naive risk budgeting portfolio*.

For nondiagonal $\boldsymbol{\Sigma}$, with the risk budgeting equations

$$w_i(\boldsymbol{\Sigma}\mathbf{w})_i = b_i\ \mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w}, \qquad i = 1,\ldots,N,$$

and $\mathbf{1}^T\mathbf{w} = 1$ and $\mathbf{w} \geq \mathbf{0}$, the solution can be characterized as

$$\boldsymbol{\Sigma}\mathbf{x} = \mathbf{b}/\mathbf{x}$$

with $\mathbf{x} \geq \mathbf{0}$ and we can always recover the portfolio by normalizing: . As shown by Spinu, this solution can be obtained by solving the following convex optimization problem: $\mathbf{w} = \mathbf{x}/(\quad \mathbf{x})$

$$\underset{\mathbf{x} \geq \mathbf{0}}{\text{minimize}} \quad \frac{1}{2}\mathbf{x}^T \boldsymbol{\Sigma} \mathbf{x} - \mathbf{b}^T \log(\mathbf{x}).^{\mathbf{1}^T}$$

For more general formulations with other constraints (like allowing shortselling or box constraints) and additional objectives (like maximizing the expected return $\mathbf{w}^T \boldsymbol{\mu}$ or minimizing the overall variance $\mathbf{w}^T \boldsymbol{\Sigma} \mathbf{w}$), we need more sophisticated formulations, which unfortunately are nonconvex.

In general, exact parity may not be achieved and one needs to define a risk term to be minimized:

$$R(\mathbf{w}) = \sum_{i,j=1}^{N} \left(g_{ij}\left(\mathbf{w}\right)\right)^2$$

or simply

$$R(\mathbf{w}) = \sum_{i=1}^{N} \left(g_{i}\left(\mathbf{w}\right)\right)^2.$$

One of the earliest risk-parity portfolio formulations is

$$\begin{array}{ll} \underset{\mathbf{w}}{\text{minimize}} & \sum_{i,j=1}^{N} \left(w_i (\boldsymbol{\Sigma} \mathbf{w})_i - w_j (\boldsymbol{\Sigma} \mathbf{w})_j \right)^2 \\ \text{subject to} & \mathbf{1}^T \mathbf{w} = 1, \quad \mathbf{w} \in \mathcal{W}, \end{array}$$

which corresponds to $g_{i,j}(\mathbf{w}) = \mathbf{w}^T (\mathbf{M}_i - \mathbf{M}_j) \mathbf{w}$ (with $\mathcal{W}$ denoting some other constraints).

Another similar formulation is

$$\begin{array}{ll} \underset{\mathbf{w}, \theta}{\text{minimize}} & \sum_{i=1}^{N} \left(w_i (\boldsymbol{\Sigma} \mathbf{w})_i - \theta \right)^2 \\ \text{subject to} & \mathbf{1}^T \mathbf{w} = 1, \quad \mathbf{w} \in \mathcal{W}. \end{array}$$

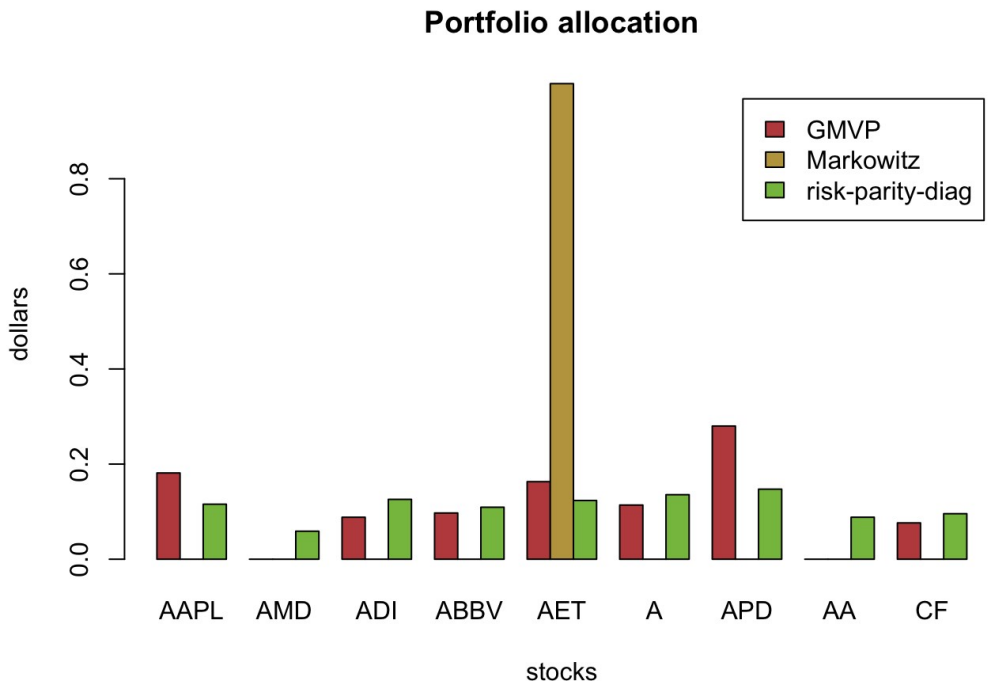
which corresponds to $g_i(\mathbf{w}) = \mathbf{w}^T \mathbf{M}_i \mathbf{w} - \theta$.

Algorithms

Closed-form solution for diagonal case

We can try the closed-form solution for the case of diagonal $\boldsymbol{\Sigma}$ even if it is not diagonal:

```
w_diag <- 1/sqrt(diag(Sigma))
w_diag <- w_diag/sum(w_diag)

# plot
w_all <- cbind(w_all, "risk-parity-diag" = w_diag)
barplot(t(w_all),
        main = "Portfolio allocation", xlab = "stocks", ylab = "dollars", beside = TRUE,
        legend = colnames(w_all), col = rainbow8equal[1:3])
```

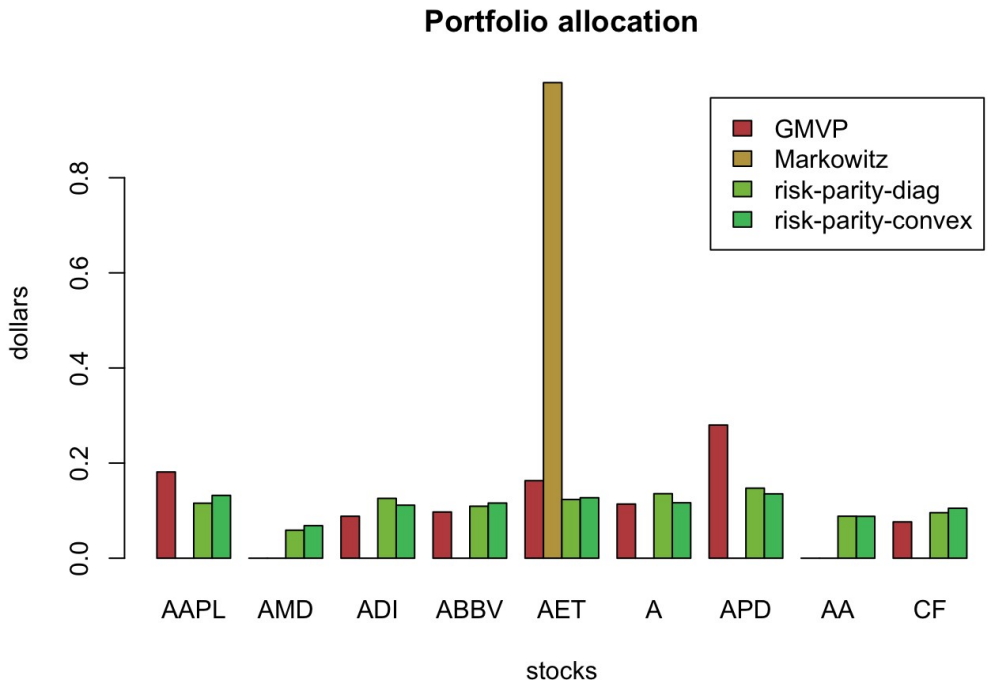


Convex formulation

Let's start with the convex formulation

$$\underset{\mathbf{x}\geq\mathbf{0}}{\text{minimize}} \quad \frac{1}{2}\mathbf{x}^T\mathbf{\Sigma}\mathbf{x} - \mathbf{b}^T\log(\mathbf{x}).$$

```r
x0 <- rep(1/N, N)  # initial point
fn_convex <- function(x, Sigma) {
  N <- nrow(Sigma)
  0.5 * t(x) %*% Sigma %*% x - (1/N)*sum(log(x))
}

# general solver
result <- optim(par = x0, fn = fn_convex, Sigma = Sigma,
                method = "BFGS")
x_convex <- result$par
w_convex <- x_convex/sum(x_convex)

# sanity check of the solution
b <- rep(1/N, N)
Sigma %*% x_convex - b/x_convex
#>               [,1]
#> AAPL  3.182503e-05
#> AMD   1.386190e-05
#> ADI   2.802476e-05
#> ABBV  2.508116e-05
#> AET   3.457522e-05
#> A     4.084369e-05
#> APD  -2.151470e-05
#> AA    9.498874e-06
#> CF   -1.350343e-05

# plot
w_all <- cbind(w_all, "risk-parity-convex" = w_convex)
barplot(t(w_all),
        main = "Portfolio allocation", xlab = "stocks", ylab = "dollars", beside = TRUE,
        legend = colnames(w_all), col = rainbow8equal[1:4])
```

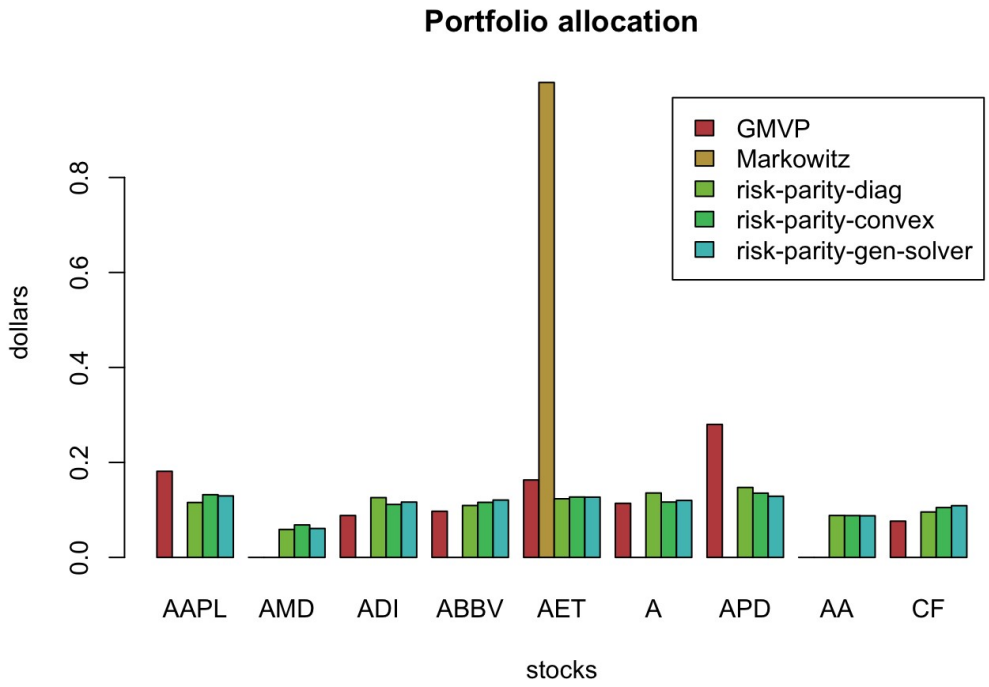


General formulations with nonlinear solvers

The general formulations are nonconvex and general-purpose nonlinear solvers may be slow. Let's consider the formulation

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \sum_{i,j=1}^{N} \left(w_i (\mathbf{\Sigma} \mathbf{w})_i - w_j (\mathbf{\Sigma} \mathbf{w})_j \right)^2 \\ \text{subject to} \quad & \mathbf{1}^T \mathbf{w} = 1, \quad \mathbf{w} \geq \mathbf{0}. \end{aligned}$$

```
w0 <- rep(1/N, N)  # initial point
fn <- function(w, Sigma) {
  N <- length(w)
  risks <-  w * (Sigma %*% w)
  g <- rep(risks, times = N) - rep(risks, each = N)
  return(sum(g^2))
}

# general solver
result <- optim(par = w0, fn = fn, Sigma = Sigma,
                method = "BFGS")
w_gen_solver <- result$par
w_gen_solver <- w_gen_solver/sum(w_gen_solver)

# plot
w_all <- cbind(w_all, "risk-parity-gen-solver" = w_gen_solver)
barplot(t(w_all),
        main = "Portfolio allocation", xlab = "stocks", ylab = "dollars", beside = TRUE,
        legend = colnames(w_all), col = rainbow8equal[1:5])
```

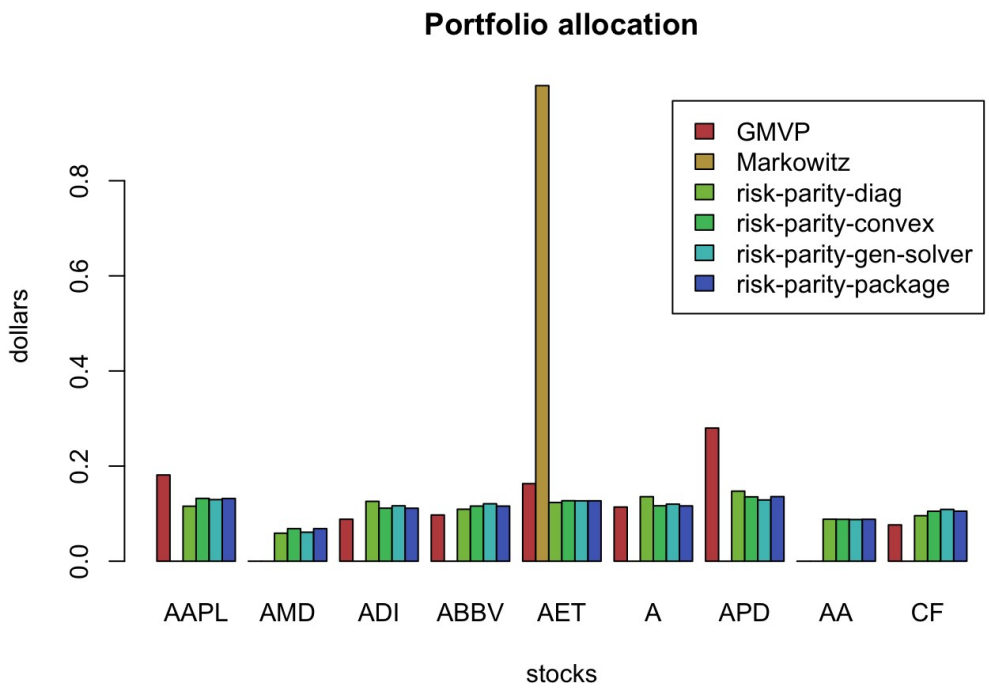


R packages

There are very few R packages for risk-parity portfolio, namely, **riskParityPortfolio** (https://github.com/dppalomar/riskParityPortfolio), **cccp**, and **FinCovRegularization**.

Let's illustrate the use of the package **riskParityPortfolio** (https://github.com/dppalomar/riskParityPortfolio) (it is much faster and reliable than the others and it allows for combinations of the risk-parity minimization with expected return maximization and other variations):

```r
#devtools::install_github("dppalomar/riskParityPortfolio")
library(riskParityPortfolio)

rpp <- riskParityPortfolio(Sigma)
names(rpp)
#> [1] "w"                "risk_contribution"
rpp$w
#> [1] 0.13185209 0.06844814 0.11141372 0.11590295 0.12696512 0.11633224
#> [7] 0.13577685 0.08810002 0.10520888
rpp$risk_contribution
#> [1] 1.305173e-05 1.305173e-05 1.305173e-05 1.305173e-05 1.305173e-05
#> [6] 1.305173e-05 1.305173e-05 1.305173e-05 1.305173e-05
c(rpp$w * (Sigma %*% rpp$w))
#> [1] 1.305173e-05 1.305173e-05 1.305173e-05 1.305173e-05 1.305173e-05
#> [6] 1.305173e-05 1.305173e-05 1.305173e-05 1.305173e-05

# plot
w_all <- cbind(w_all, "risk-parity-package" = rpp$w)
barplot(t(w_all),
        main = "Portfolio allocation", xlab = "stocks", ylab = "dollars", beside = TRUE,
        legend = colnames(w_all), col = rainbow8equal[1:6])
```



SCA method

And finally we will devise our own algorithm based on the successive convex approximation (SCA) method. Basically, at the k-th iteration, one approximates the problem by a convex one around the current point $\mathbf{w}^{(k)}$. In our case, the approximated problem is a QP:

$$\begin{array}{ll} \underset{\mathbf{w}}{\text{minimize}} & \tilde{U}\left(\mathbf{w}, \mathbf{w}^k\right) = \frac{1}{2}\mathbf{w}^T\mathbf{Q}^k\mathbf{w} + \mathbf{w}^T\mathbf{q}^k + \lambda F\left(\mathbf{w}\right) \\ \text{subject to} & \mathbf{1}^T\mathbf{w} = 1, \quad \mathbf{w} \geq \mathbf{0}, \end{array}$$

where

$$\mathbf{Q}^k \triangleq 2\left(\mathbf{A}^k\right)^T \mathbf{A}^k + \tau \mathbf{I},$$
$$\mathbf{q}^k \triangleq 2\left(\mathbf{A}^k\right)^T \mathbf{g}\left(\mathbf{w}^k\right) - \mathbf{Q}^k \mathbf{w}^k,$$
$$\mathbf{A}^k \triangleq \left[\nabla g_1\left(\mathbf{w}^k\right), \ldots, \nabla g_N\left(\mathbf{w}^k\right)\right]^T,$$
$$\mathbf{g}\left(\mathbf{w}^k\right) \triangleq \left[g_1\left(\mathbf{w}^k\right), \ldots, g_N\left(\mathbf{w}^k\right)\right]^T.$$

This QP can be solved by a solver. In the particular case of only having equality constraints $\mathbf{C}\mathbf{w} = \mathbf{c}$, then from the KKT optimality conditions the optimal solution is found as

$$\hat{\mathbf{w}}^k = -(\mathbf{Q}^k)^{-1}(\mathbf{q}^k + \mathbf{C}^T \boldsymbol{\lambda}^k),$$

where $\boldsymbol{\lambda}^k = -\left(\mathbf{C}(\mathbf{Q}^k)^{-1}\mathbf{C}^T\right)^{-1}\left(\mathbf{C}(\mathbf{Q}^k)^{-1}\mathbf{q}^k + \mathbf{c}\right)$.

Finally, the next iterate is formed by introducing some smoothing:

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \gamma^k\left(\hat{\mathbf{w}}^k - \mathbf{w}^k\right),$$

where γ^k can be chosen in practice as

$$\gamma^k = \gamma^{k-1}(1 - \zeta\gamma^{k-1})$$

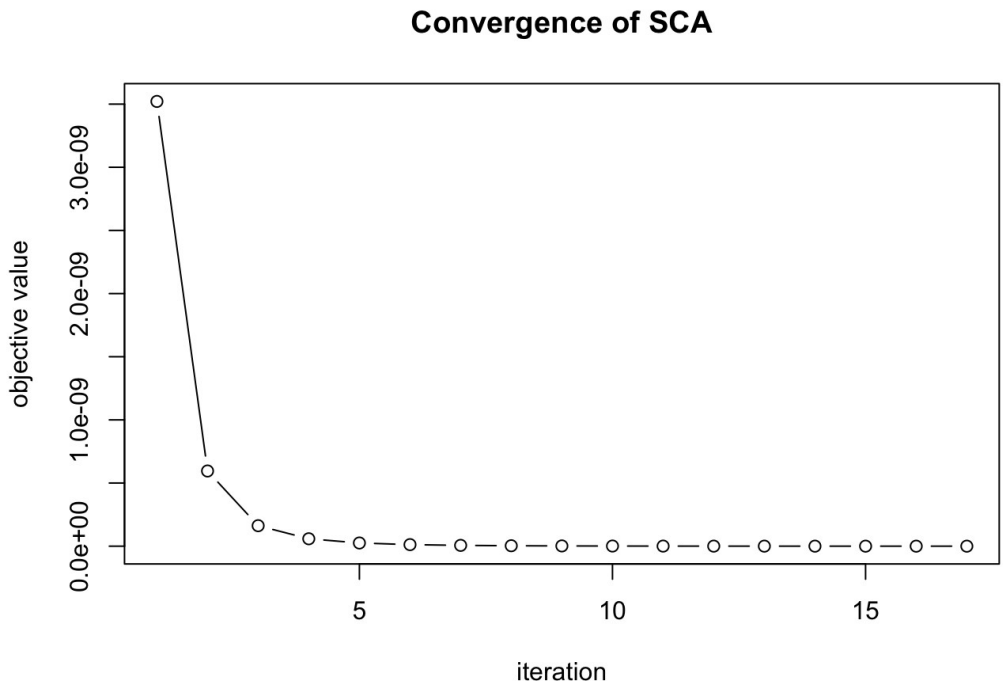
with $\gamma_0 \in (0, 1]$ and $\zeta \in (0, 1)$.

We are ready to implement our own algorithm based on the SCA approach. First we define a function that computes the functions $g_{i,j}$ and their gradients (which are needed for $\mathbf{g}\left(\mathbf{w}^k\right)$ and $\mathbf{A}^k$):

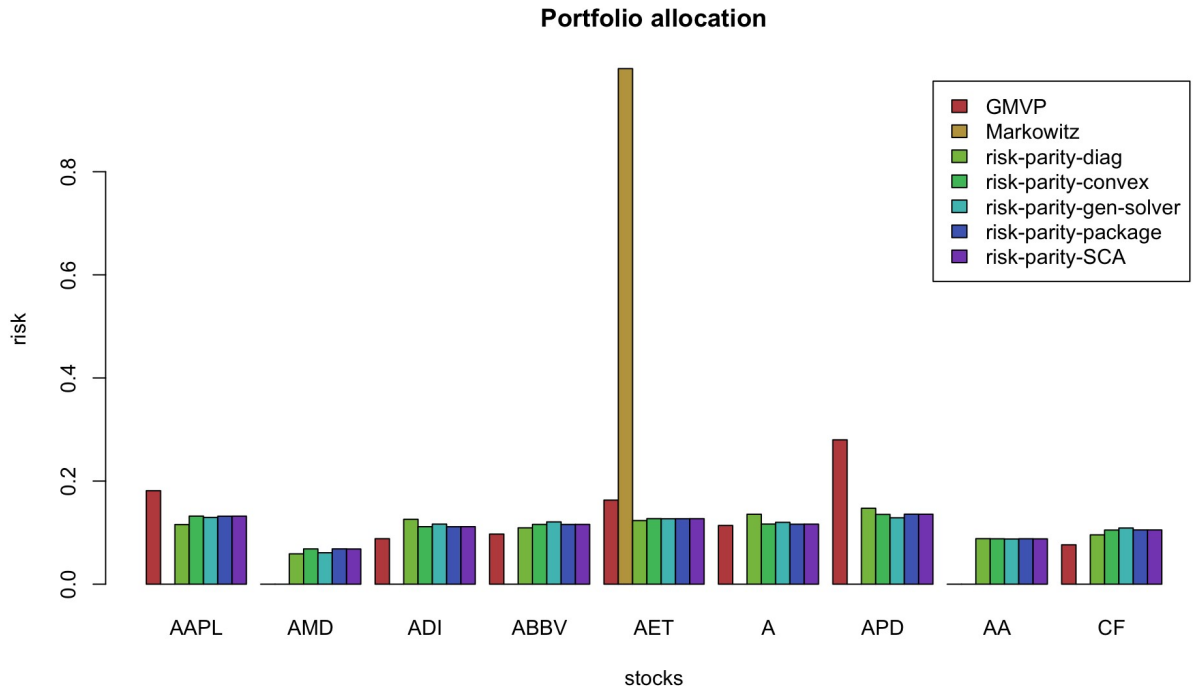
```
compute_gA <- function(w, Sigma) {
  N <- length(w)
  g <- rep(NA, N^2)
  A <- matrix(NA, N^2, N)
  for (i in 1:N) {
    Mi <- matrix(0, N, N)
    Mi[i, ] <- Sigma[i, ]
    for (j in 1:N) {
      Mj <- matrix(0, N, N)
      Mj[j, ] <- Sigma[j, ]
      #g[i + (j-1)*N]   <- t(w) %*% (Mi - Mj) %*% w
      g[i + (j-1)*N]   <- w[i]*(Sigma[i, ] %*% w) - w[j]*(Sigma[j, ] %*% w)
      A[i + (j-1)*N, ] <- (Mi + t(Mi) - Mj - t(Mj)) %*% w
      #A[i + (j-1)*N, ] <- (Sigma[i, ] %*% w
    }
  }
  # # this is much faster:
  # wSw <- w * (Sigma %*% w)
  # g <- rep(wSw, times = N) - rep(wSw, each = N)  # N^2 different g_{i,j}
  return(list(g = g, A = A))
}
```

Now we can implement the main loop of the SCA algorithm:

```r
library(quadprog)  # install.packages("quadprog")

# parameters
max_iter <- 40
tau <- 1e-6
zeta <- 0.1
gamma <- 0.99
# initial point
obj_value <- NULL
w_SCA <- rep(1/N, N)
for (k in 1:max_iter) {
  # compute parameters
  gA <- compute_gA(w_SCA, Sigma)
  g <- gA$g
  A <- gA$A
  Q <- 2 * t(A) %*% A + tau*diag(N)  # crossprod(A) = t(A) %*% A
  q <- 2 * t(A) %*% g - Q %*% w_SCA
  obj_value <- c(obj_value, sum(g^2))

  # solve problem with CVXR
  w_ <- Variable(N)
  prob <- Problem(Minimize(0.5*quad_form(w_, Q) + t(q) %*% w_),
                  constraints = list(sum(w_) == 1))
  result <- solve(prob)
  w_ <- as.vector(result$getValue(w_))

  # solve the problem with solve.QP()
  w__ <- solve.QP(Q, -q, matrix(1, N, 1), 1, meq = 1)$solution

  # solve problem in closed form
  C <- matrix(1, 1, N)
  c <- 1
  CinvQ <- C %*% solve(Q)
  lmd <- solve(CinvQ %*% t(C), -(CinvQ %*% q + c))
  w___ <- solve(Q, -(q + t(C) %*% lmd))

  #sanity checks for different solvers
  if ((err <- norm(w_ - w__, "2")/norm(w_, "2")) > 1e-2)
    cat("CVXR and solve.QP do not match:", err, "\n")
  if ((err <- norm(w_ - w___, "2")/norm(w_, "2")) > 1e-2)
    cat("Closed-form solution and CVXR do not match:", err, "\n")

  # next w
  gamma <- gamma*(1 - zeta*gamma)
  w_SCA_prev <- w_SCA
  w_SCA <- w_SCA + gamma*(w_ - w_SCA)

  # stopping criterion
  #if ((norm(w-w_prev, "2")/norm(w_prev, "2")) < 1e-6)
  #  break
  if (k>1 && abs((obj_value[k]-obj_value[k-1])/obj_value[k-1]) < 1e-1)
    break
}
cat("Number of iterations:", k)
#> Number of iterations: 17
plot(obj_value, type = "b",
     main = "Convergence of SCA", xlab = "iteration", ylab = "objective value")
```

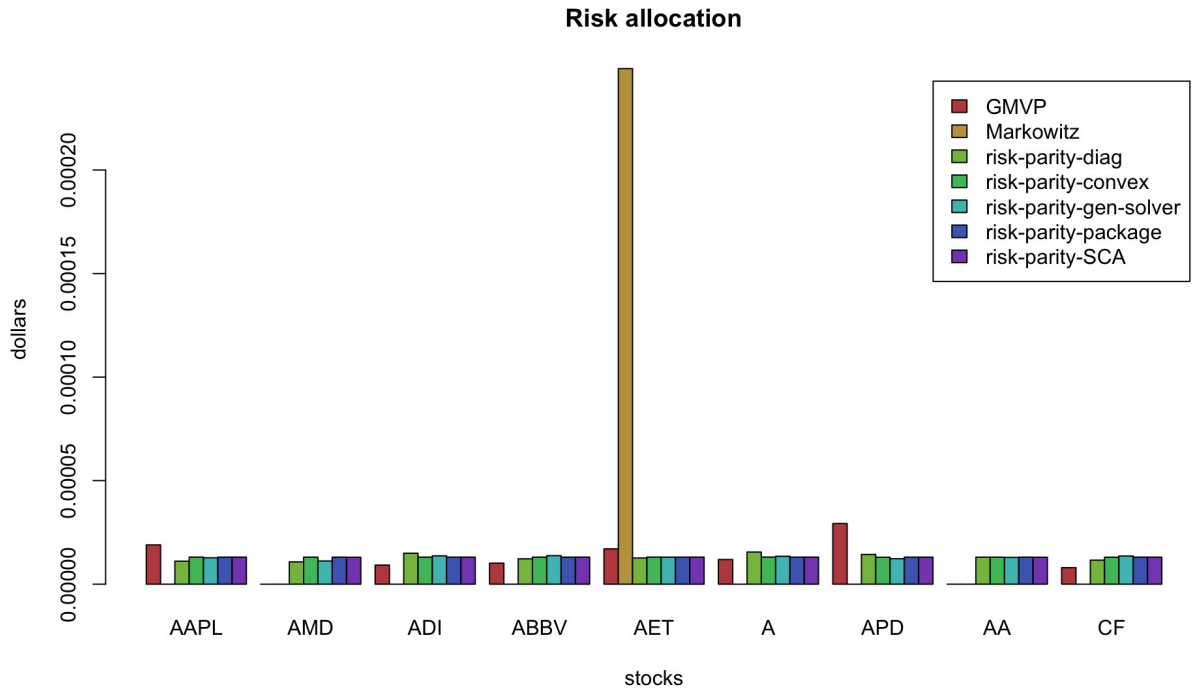



```
w_all <- cbind(w_all, "risk-parity-SCA" = w_SCA)
barplot(t(w_all),
        main = "Portfolio allocation", xlab = "stocks", ylab = "risk", beside = TRUE,
        legend = colnames(w_all), col = rainbow8equal[1:7])
```



We can see that the risk-parity portfolio is more diversified in terms of dollar allocation than Markowitz and even GMVP. And we can also compute the risk allocation:

```
risk_contribution_all <- cbind("GMVP" = w_GMVP * (Sigma %*% w_GMVP),
                               "Markowitz" = w_Markowitz * (Sigma %*% w_Markowitz),
                               "risk-parity-diag" = w_diag * (Sigma %*% w_diag),
                               "risk-parity-convex" = w_convex * (Sigma %*% w_convex),
                               "risk-parity-gen-solver" = w_gen_solver * (Sigma %*% w_gen_solver),
                               "risk-parity-package" = rpp$w * (Sigma %*% rpp$w),
                               "risk-parity-SCA" = w_SCA * (Sigma %*% w_SCA))
barplot(t(risk_contribution_all),
        main = "Risk allocation", xlab = "stocks", ylab = "dollars", beside = TRUE,
        legend = colnames(w_all), col = rainbow8equal[1:7])
```



Indeed, the risk-parity portfolio has an equal risk allocation (no surprise since it was designed for that purpose).

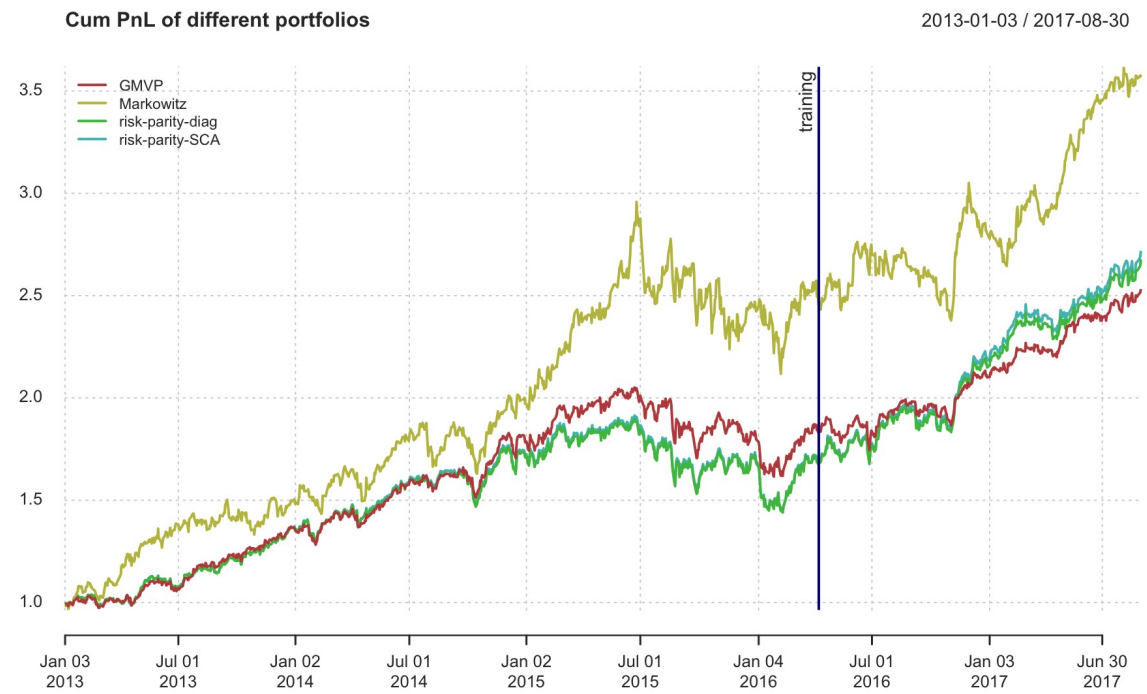
It is instructive now to compare the performance (in sample vs out-of-sample):

```
# compute returns of all portfolios
ret_all <- xts(X_lin %*% w_all[, c("GMVP", "Markowitz", "risk-parity-diag", "risk-parity-SCA")],
               order.by = index(X_lin))
ret_all_trn <- ret_all[1:T_trn, ]
ret_all_tst <- ret_all[-c(1:T_trn), ]

# performance
t(table.AnnualizedReturns(ret_all_trn))
#>                  Annualized Return Annualized Std Dev
#> GMVP                        0.2041             0.1622
#> Markowitz                   0.3154             0.2503
#> risk-parity-diag            0.1721             0.1709
#> risk-parity-SCA             0.1733             0.1718
#>                  Annualized Sharpe (Rf=0%)
#> GMVP                              1.2584
#> Markowitz                         1.2599
#> risk-parity-diag                  1.0067
#> risk-parity-SCA                   1.0086
t(table.AnnualizedReturns(ret_all_tst))
#>                  Annualized Return Annualized Std Dev
#> GMVP                        0.2604             0.1234
#> Markowitz                   0.3138             0.1860
#> risk-parity-diag            0.3968             0.1488
#> risk-parity-SCA             0.4090             0.1520
#>                  Annualized Sharpe (Rf=0%)
#> GMVP                              2.1092
#> Markowitz                         1.6875
#> risk-parity-diag                  2.6669
#> risk-parity-SCA                   2.6910
```

We can see that the in the in sample the risk-parity portfolio is the worst, but actually in the out-of-sample it is the best by far!

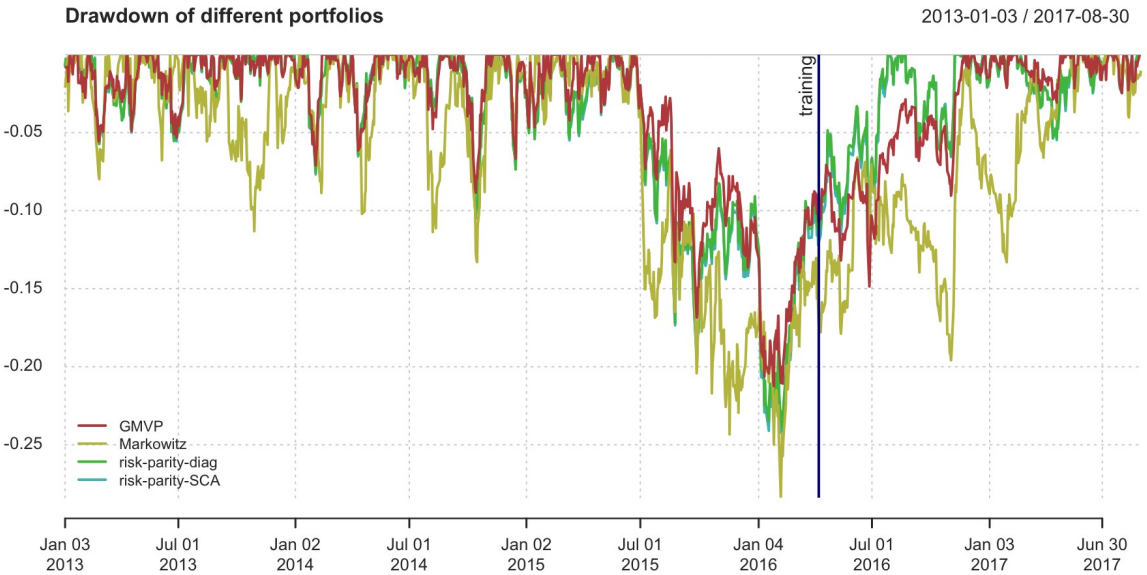
Let us plot the cumulative PnL over time (recall that it looks deceptive to the untrained eye):

```
{ chart.CumReturns(ret_all, main = "Cum PnL of different portfolios",
                   wealth.index = TRUE, legend.loc = "topleft", colorset = rainbow6equal)
  addEventLines(xts("training", index(X_lin[T_trn])), srt=90, pos=2, lwd = 2, col = "darkblue") }
```



The drawdown, on the other hand, clearly shows the superior performance of the risk-parity portfolio:

```
{ chart.Drawdown(ret_all, main = "Drawdown of different portfolios",
                 legend.loc = "bottomleft", colorset = rainbow6equal)
  addEventLines(xts("training", index(X_lin[T_trn])), srt=90, pos=2, lwd = 2, col = "darkblue") }
```



Formulation with expected return

Thus far, the risk-parity formulation was dealing only with the risk contributions while ignoring the expected return. We can now design a risk-parity portfolio that also takes into account the mean return $\mathbf{w}^T\boldsymbol{\mu}$ as

$$\begin{array}{ll} \underset{\mathbf{w}}{\text{minimize}} & \sum_{i,j=1}^{N}\left(w_i(\boldsymbol{\Sigma}\mathbf{w})_i - w_j(\boldsymbol{\Sigma}\mathbf{w})_j\right)^2 - \lambda\mathbf{w}^T\boldsymbol{\mu} \\ \text{subject to} & \mathbf{1}^T\mathbf{w} = 1, \quad \mathbf{w} \in \mathcal{W}, \end{array}$$

The existing R package **riskParityPortfolio** (https://github.com/dppalomar/riskParityPortfolio) can deal with such

formulation:

```
rpp_mu <- riskParityPortfolio(Sigma, mu = mu, lambda = 8e-5, formulation = "rc-double-index")

w_all <- cbind(w_all[, c("GMVP", "Markowitz", "risk-parity-diag", "risk-parity-SCA")],
               "risk-parity-mu" = rpp_mu$w)
```

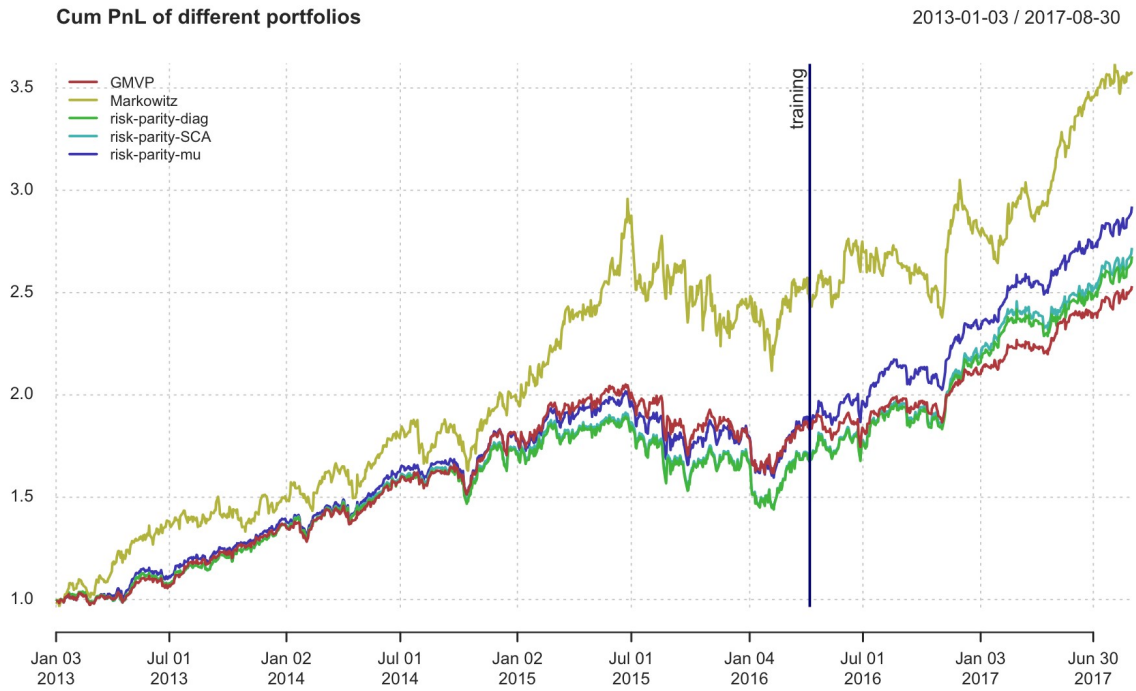
Let's compare the performance (in sample vs out-of-sample):

```
# compute returns of all portfolios
ret_all <- xts(X_lin %*% w_all, index(X_lin))
ret_all_trn <- ret_all[1:T_trn, ]
ret_all_tst <- ret_all[-c(1:T_trn), ]

# performance
t(table.AnnualizedReturns(ret_all_trn))
#>                  Annualized Return Annualized Std Dev
#> GMVP                        0.2041             0.1622
#> Markowitz                   0.3154             0.2503
#> risk-parity-diag            0.1721             0.1709
#> risk-parity-SCA             0.1733             0.1718
#> risk-parity-mu              0.2091             0.1679
#>                  Annualized Sharpe (Rf=0%)
#> GMVP                              1.2584
#> Markowitz                         1.2599
#> risk-parity-diag                  1.0067
#> risk-parity-SCA                   1.0086
#> risk-parity-mu                    1.2452
t(table.AnnualizedReturns(ret_all_tst))
#>                  Annualized Return Annualized Std Dev
#> GMVP                        0.2604             0.1234
#> Markowitz                   0.3138             0.1860
#> risk-parity-diag            0.3968             0.1488
#> risk-parity-SCA             0.4090             0.1520
#> risk-parity-mu              0.3827             0.1319
#>                  Annualized Sharpe (Rf=0%)
#> GMVP                              2.1092
#> Markowitz                         1.6875
#> risk-parity-diag                  2.6669
#> risk-parity-SCA                   2.6910
#> risk-parity-mu                    2.9015
```

The new risk-parity portfolio that takes into account the expected return has a very good Sharpe ratio!

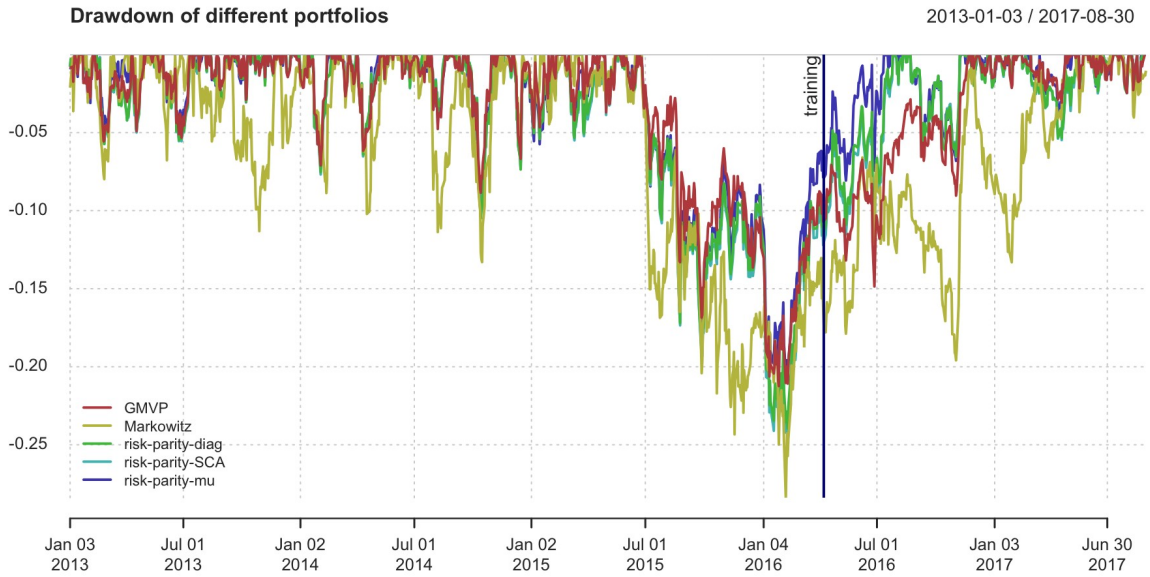
Let us plot the cumulative PnL over time (recall that it looks deceptive to the untrained eye):

```
{ chart.CumReturns(ret_all, main = "Cum PnL of different portfolios",
                   wealth.index = TRUE, legend.loc = "topleft", colorset = rainbow6equal)
  addEventLines(xts("training", index(X_lin[T_trn])), srt=90, pos=2, lwd = 2, col = "darkblue") }
```



The drawdown, on the other hand, clearly shows the superior performance of the risk-parity portfolio including the expected return:

```
{ chart.Drawdown(ret_all, main = "Drawdown of different portfolios",
                 legend.loc = "bottomleft", colorset = rainbow6equal)
  addEventLines(xts("training", index(X_lin[T_trn])), srt=90, pos=2, lwd = 2, col = "darkblue") }
```



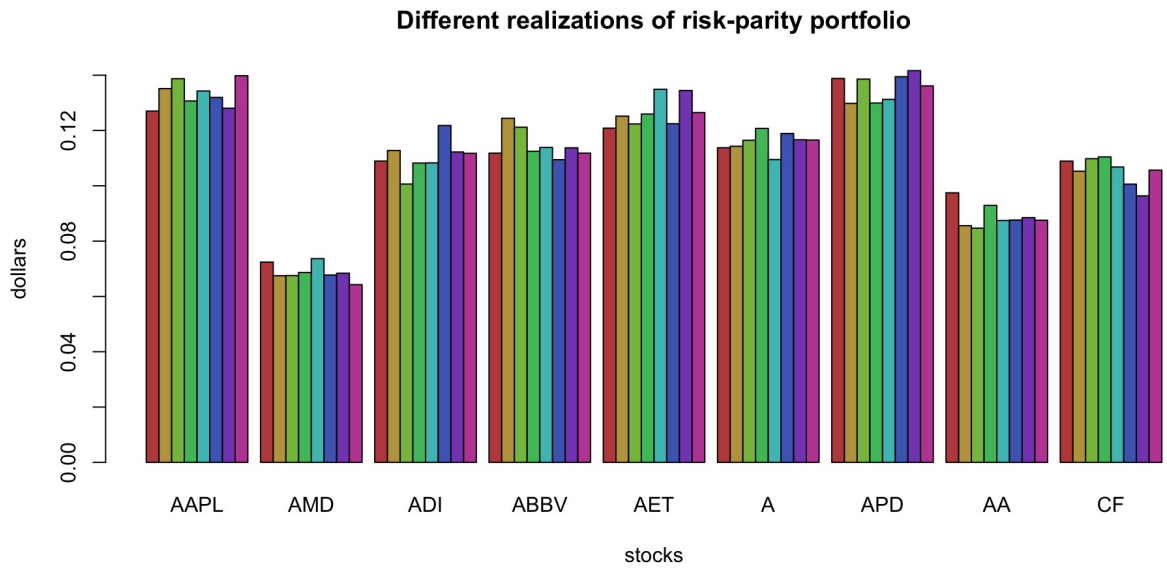
Sensitivity to parameters

To study the sensitivity with respect to the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we will design multiple portfolios based on different samples from the training set:

```
w_risk_parity_mu_acc <- w_risk_parity_acc <- NULL
for (i in 1:8) {
  # sample means with random samples
  idx <- sample(1:T_trn, T_trn/2)
  mu_ <- colMeans(X_log_trn[idx, ])
  Sigma_ <- cov(X_log_trn[idx, ])

  # design risk-parity portfolio
  w_risk_parity_acc <- cbind(w_risk_parity_acc,
                             riskParityPortfolio(Sigma_)$w)
  w_risk_parity_mu_acc <- cbind(w_risk_parity_mu_acc,
                                riskParityPortfolio(Sigma_, mu = mu_, lambda = 8e-5, formulation =
"rc-double-index")$w)
}
rownames(w_risk_parity_mu_acc) <- rownames(w_risk_parity_acc) <- colnames(X_lin)
```

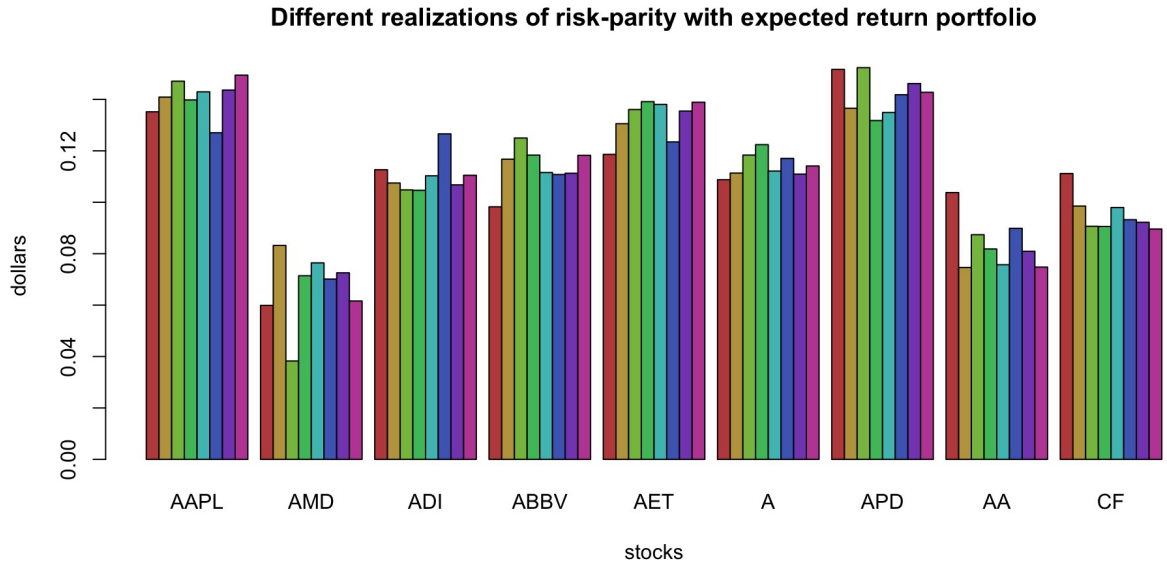
The risk-parity portfolio is not very sensitive to the parameter $\boldsymbol{\Sigma}$ since all the realizations have a similar allocation:

```
barplot(t(w_risk_parity_acc),
        main = "Different realizations of risk-parity portfolio", xlab = "stocks", ylab = "dollar
s",
        beside = TRUE, col = rainbow8equal)
```



Let's check out the risk-parity portfolio that takes into account the expecter return:

```
barplot(t(w_risk_parity_mu_acc),
        main = "Different realizations of risk-parity with expected return portfolio",
        xlab = "stocks", ylab = "dollars",
        beside = TRUE, col = rainbow8equal)
```



We can observe that this risk-parity portfolio is still not very sensitive to the parameters including $\boldsymbol{\mu}$. However, if the value of λ is increased to, say, 10^{-5} then one can observe a stronger sensitivity. In practice, one always needs to include leverage constraints $\|\mathbf{w}\|_1 = 1$, which would make it even more stable and less sensitive.

Conclusion

We can conclude with the following points:

- Markowitz porftolio, while it started the field of modern portfolio theory in 1952, has not been embraced by practitioners because
  - it does not diversify the risk and concentrates in a few assets
  - it is too sensitive to the parameters (particularly to $\boldsymbol{\mu}$).
- The risk-parity portfolio precisely aims at diversifying the risk contribution and it also happens to be less sensitive to the parameters.
- We have explored several numerical methods to design the risk-parity portfolio.
- The R package **riskParityPortfolio** (https://github.com/dppalomar/riskParityPortfolio) implements all the methods based on the aforementioned references.