# CVXR for PortfolioAnalytics

Xinran Zhao

2022/9/1

**Abstract**

The purpose of this vignette is to demonstrate examples of optimization problems that can be solved in PortfolioAnalytics with CVXR and its many supported solvers. The problem types covered include not only Linear Programming(LP), Quadratic Programming(QP) but also Second-Order Cone Programming(SOCP). Multiple solvers supported by CVXR can be selected according to optimization types. SCS and ECOS can completely cover the types of problems that ROI can deal with, such as mean-variance and ES problem. In order to better understand the functions of PortfolioAnalytics, users are recommended to read the Vignette *Introduction to PortfolioAnalytics* first.

## Contents

**1 Getting Started 3**

1.1 Load Packages . . . 3

1.2 Solvers . . . 3

1.3 Data . . . 3

1.4 Optimization Problems . . . 5

**2 Maximizing Mean Return 5**

2.1 Portfolio Object . . . 5

2.2 Optimization . . . 6

2.3 Backtesting . . . 6

**3 Minimizing Variance 7**

3.1 Global Minimum Variance Portfolio . . . 7

3.1.1 Portfolio Object . . . 7

3.1.2 Optimization . . . 7

3.2 Linearly Constrained Minimum Variance Portfolio . . . 8

**4 Maximizing Quadratic Utility 10**

4.1 Portfolio Object . . . 10

4.2 Optimization . . . 10

**5 Minimizing Expected Shortfall** **11**

5.1 Portfolio Object . . . . . 12

5.2 Optimization . . . . . 12

**6 Minimizing Expected Quadratic Shortfall** **13**

6.1 Portfolio Object . . . . . 13

6.2 Optimization . . . . . 13

6.3 Backtesting . . . . . 13

**7 Maximizing Mean Return Per Unit Risk** **16**

7.1 Maximum Sharpe Ratio Portfolios . . . . . 16

7.2 Maximum ES ratio Portfolios . . . . . 17

7.3 Maximum EQS ratio Portfolios . . . . . 18

**8 Efficient Frontier** **19**

# 1 Getting Started

## 1.1 Load Packages

Load the necessary packages.

```
library(PortfolioAnalytics)
library(CVXR)
library(data.table)
library(xts)
```

## 1.2 Solvers

The website https://cvxr.rbind.io/ shows that CVXR currently supports us to use 9 solvers, some of which are commercial (CBC, CPLEX, GUROBI, MOSEK) and the others are open source(GLPK, GLPK_MI, OSQP, SCS, ECOS).

Different solvers support different types of portfolio optimization problems. The `optimize_method=c("CVXR", {CVXRsolver})` argument of the function `optimize.portfolio` allows the user to specify the solver to use with CVXR. If the argument is `optimize_method="CVXR"`, the default solver for LP and QP type portfolio optimization problems such as maximum mean return and minimum variance portfolio optimization, will be OSQP, and the default solver for SOCP type portfolio optimizations, such as "robust portfolio optimization" to control for alpha uncertainty, and Expected Quadratic Shortfall (EQS) portfolio optimization, will be SCS.

| Solver | LP | QP | SOCP |
|---|---|---|---|
| CBC | ✓ | | |
| GLPK | ✓ | | |
| GLPK_MI | ✓ | | |
| OSQP | ✓ | ✓ | |
| SCS | ✓ | ✓ | ✓ |
| ECOS | ✓ | ✓ | ✓ |
| CPLEX | ✓ | ✓ | ✓ |
| GUROBI | ✓ | ✓ | ✓ |
| MOSEK | ✓ | ✓ | ✓ |

## 1.3 Data

The edhec data set from the PerformanceAnalytics package is used as example data for most, but not all, of the examples to follow. The edhec data contains monthly returns for 13 assets from 1997-01 to 2019-11. We use the first 4 assets as the example data to mainly show how to use the code.

```
data(edhec)
# Use the first 4 columns in edhec for a returns object
returns <- edhec[, 1:4]
colnames(returns) <- c("CA", "CTAG", "DS", "EM")
print(head(returns, 5))
```

```
##                CA   CTAG     DS     EM
## 1997-01-31 0.0119 0.0393 0.0178 0.0791
## 1997-02-28 0.0123  0.0298  0.0122  0.0525
## 1997-03-31 0.0078 -0.0021 -0.0012 -0.0120
## 1997-04-30 0.0086 -0.0170  0.0030  0.0119
## 1997-05-31 0.0156 -0.0015  0.0233  0.0315
```

```
# Get a character vector of the asset names
funds <- colnames(returns)
```

The CRSP data set is the daily log returns of 30 small cap stocks from 1993-01 to 2015-12 from the Center for Research in Security Prices (CRSP). We use this larger and more frequent data set to show more meaningful and interesting results, such as the performance of portfolios and the Efficient Frontiers. We don't want to use the large data set everywhere to slow down the code or distract the main point.

```
load("stocksCRSPdaily.rda")

stocks = stocksCRSPdaily[CapGroup == "SmallCap"]
returnMat = tapply(stocks[, ReturnD], list(stocks$Date, stocks$TickerLast), I)
smallcapD = xts(returnMat, as.Date(rownames(returnMat)))
sc_30 = c("TGNA", "AVP", "PBI", "THC", "AVY", "HAS", "TSS", "SPXC", "R", "HP", "J",
          "DBD", "HAR", "BIG", "HSC", "MLHR", "AXE", "MATX", "KBH", "BGG", "CRS",
          "UVV", "MENT", "HTLD", "BRC", "FUL", "ESND", "BOBE", "PIR", "WTS")
returns_sc = smallcapD[, sc_30]
print(head(returns_sc, 3))
```

```
##                   TGNA          AVP          PBI         THC          AVY
## 1993-01-04  0.02216749 -0.009029346  0.003134796  0.01010101 -0.004347826
## 1993-01-05 -0.01204819 -0.006833713 -0.009375000  0.00000000 -0.004366812
## 1993-01-06  0.02195122 -0.011467890 -0.015772870 -0.02000000 -0.004385965
##                     HAS         TSS         SPXC           R          HP
## 1993-01-04 -0.003831418 0.000000000 -0.013888889 0.018181818 -0.01015228
## 1993-01-05  0.003846154 0.008695652 -0.007042253 0.004464286 -0.03589744
## 1993-01-06  0.007662835 0.000000000 -0.035460994 0.000000000 -0.01595745
##                       J          DBD         HAR          BIG          HSC
## 1993-01-04 -0.004629630 -0.002066116  0.00000000 -0.006944444 -0.016501650
## 1993-01-05  0.000000000 -0.004140787 -0.02459016  0.000000000  0.003355705
## 1993-01-06 -0.009302326 -0.002079002  0.00000000 -0.013986014  0.010033445
##                  MLHR          AXE        MATX          KBH         BGG
## 1993-01-04 0.04827586  0.016393442 -0.02020202 -0.007692308 0.026881721
## 1993-01-05 0.01973684  0.037634410  0.00000000  0.007751938 0.007853403
## 1993-01-06 0.04516129 -0.005181347  0.01030928  0.023076924 0.023376623
##                   CRS         UVV       MENT       HTLD          BRC
## 1993-01-04  0.00000000 -0.02197802 0.01538462 0.04347826  0.004054054
## 1993-01-05 -0.01960784  0.01782772 0.07575758 0.00000000 -0.023648649
## 1993-01-06  0.00750000 -0.01111111 0.04225352 0.02777778  0.031141868
##                   FUL       ESND        BOBE         PIR        WTS
## 1993-01-04  0.00617284 0.08088236 -0.05649717  0.00000000 0.00795756
## 1993-01-05  0.00000000 0.02040816  0.02395210 -0.02000000 0.00000000
## 1993-01-06 -0.01226994 0.02666667  0.00000000  0.03061224 0.01052632
```

```
funds_sc = colnames(returns_sc)
```

## 1.4 Optimization Problems

All optimization problems treated will use linear constraints unless stated otherwise. There will be one equality constraint, i.e., the full-investment constraint, and one or more inequality constraints such as the long-only and box constraints. More comprehensive constraint types can be queried from *Introduction to PortfolioAnalytics*. This vignette will be organized by objective type and provide some visual examples.

## 2 Maximizing Mean Return

The objective to maximize mean return is a linear problem of the form:

$$\max_{w} \quad \hat{\boldsymbol{\mu}}'\boldsymbol{w}$$

$$\begin{aligned} s.t. \quad & A\boldsymbol{w} \geq b \\ & B\boldsymbol{w} = c \end{aligned}$$

Where $\hat{\boldsymbol{\mu}}$ is the estimated asset returns mean vector and $\boldsymbol{w}$ is the vector of portfolio weights.

### 2.1 Portfolio Object

The first step in setting up a model is to create the portfolio object. Then add constraints and a return objective.

```r
# Create portfolio object
pspec_maxret <- portfolio.spec(assets=funds)
# Add constraints to the portfolio object
pspec_maxret <- add.constraint(pspec_maxret, type="full_investment")
pspec_maxret <- add.constraint(portfolio = pspec_maxret, type = "box",
                              min = c(0.02, 0.05, 0.03, 0.02),
                              max = c(0.55, 0.6, 0.65, 0.5))
# Add objective to the portfolio object
pspec_maxret <- add.objective(portfolio = pspec_maxret,
                             type = "return", name = "mean")
pspec_maxret
```

```
## **************************************************
## PortfolioAnalytics Portfolio Specification
## **************************************************
##
## Call:
## portfolio.spec(assets = funds)
##
## Number of assets: 4
## Asset Names
## [1] "CA"   "CTAG" "DS"   "EM"
##
## Constraints
## Enabled constraint types
##      - full_investment
##      - box
##
```

```
## Objectives:
## Enabled objective names
##      - mean
```

## 2.2 Optimization

The next step is to run the optimization. Note that `optimize_method=c("CVXR", {CVXRsolver})` should be specified in the function `optimize.portfolio` to use CVXR solvers for the optimization, or use the default solver by giving `optimize_method="CVXR"`. For maximizing mean return problem, which is a linear programming, the default solver is `OSQP`.

```r
# Run the optimization with default solver
opt_maxret <- optimize.portfolio(R=returns, portfolio=pspec_maxret,
                                 optimize_method="CVXR", trace=TRUE)
opt_maxret
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = returns, portfolio = pspec_maxret, optimize_method = "CVXR",
##     trace = TRUE)
##
## Optimal Weights:
##   CA CTAG   DS   EM
## 0.02 0.05 0.65 0.28
##
## Objective Measures:
##     mean
## 0.006371
```

```r
opt_maxret$solver
```

```
## [1] "OSQP"
```

```r
# Run the optimization with specific solver
opt_maxret_glpk <- optimize.portfolio(R=returns, portfolio=pspec_maxret,
                                 optimize_method=c("CVXR", "GLPK"), trace=TRUE)
opt_maxret_glpk$solver
```

```
## [1] "GLPK"
```

## 2.3 Backtesting

An out of sample backtest is run with `optimize.portfolio.rebalancing`. In this example, an initial training period of 36 months is used and the portfolio is rebalanced quarterly.

```
bt_maxret <- optimize.portfolio.rebalancing(R=returns, portfolio=pspec_maxret,
                                            optimize_method="CVXR",
                                            rebalance_on="quarters",
                                            training_period=36)
```

The call to `optimize.portfolio.rebalancing` returns the `bt_maxret` object which is a list containing the optimal weights and objective measure at each rebalance period.

```
class(bt_maxret)
```

```
## [1] "optimize.portfolio.rebalancing"
```

```
names(bt_maxret)
```

```
## [1] "portfolio"       "R"               "call"            "elapsed_time"
## [5] "opt_rebalancing"
```

## 3 Minimizing Variance

The objective to minimize variance is a quadratic problem of the form:

$$\min_{w} \quad \boldsymbol{w}'\Sigma\boldsymbol{w}$$

subject to only the full-investment constraint, where $\Sigma$ is the estimated covariance matrix of asset returns and $\boldsymbol{w}$ is the set of weights. It is a quadratic problem.

### 3.1 Global Minimum Variance Portfolio

#### 3.1.1 Portfolio Object

In this example, the only constraint specified is the full investment constraint, therefore the optimization problem is solving for the global minimum variance portfolio.

```
# Create portfolio object
pspec_gmv <- portfolio.spec(assets=funds)
# Add full-investment constraint
pspec_gmv <- add.constraint(pspec_gmv, type="full_investment")
# Add objective of minimizing variance
pspec_gmv <- add.objective(portfolio = pspec_gmv, type = "risk", name = "var")
```

#### 3.1.2 Optimization

```
opt_gmv <- optimize.portfolio(returns, pspec_gmv, optimize_method = "CVXR")
opt_gmv
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = returns, portfolio = pspec_gmv, optimize_method = "CVXR")
##
## Optimal Weights:
##      CA    CTAG      DS      EM
##  0.3637  0.2887  0.5581 -0.2105
##
## Objective Measures:
##  StdDev
## 0.01187
```

The optimal portfolio may have short position when there is only a full-investment constraint.

## 3.2 Linearly Constrained Minimum Variance Portfolio

Various linear inequality constraint, such as box constraints, group constraints and a target mean return constraint, can be used with GMV portfolio construction. Here we demonstrate the case of linearly constrained minimum variance portfolio.

```
# portfolio object
pspec_mv <- add.constraint(pspec_gmv, type = "long_only")
pspec_mv <- add.constraint(pspec_mv, type = "group",
                           groups=list(groupA=1,
                                       groupB=c(2, 3),
                                       groupC=4),
                           group_min=c(0, 0.25, 0.01),
                           group_max=c(0.45, 0.6, 0.5))
pspec_mv <- add.constraint(pspec_mv, type = "return", return_target=0.006)
pspec_mv
```

```
## **************************************************
## PortfolioAnalytics Portfolio Specification
## **************************************************
##
## Call:
## portfolio.spec(assets = funds)
##
## Number of assets: 4
## Asset Names
## [1] "CA"   "CTAG" "DS"   "EM"
##
## Constraints
## Enabled constraint types
##      - full_investment
##      - long_only
##      - group
##      - return
##
```

```
## Objectives:
## Enabled objective names
##      - var
```

```
# optimization
opt_mv <- optimize.portfolio(returns, pspec_mv, optimize_method = "CVXR")
opt_mv
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = returns, portfolio = pspec_mv, optimize_method = "CVXR")
##
## Optimal Weights:
##     CA   CTAG     DS     EM
## 0.3900 0.0733 0.5267 0.0100
##
## Objective Measures:
##  StdDev
## 0.01452
```

```
# backtesting
bt_mv <- optimize.portfolio.rebalancing(R=returns, portfolio=pspec_mv,
                                           optimize_method="CVXR",
                                           rebalance_on="quarters",
                                           training_period=36)
```

The use of an alternative to the CVXR default solver will get the same result to many significant digits. In this example we use `optimize_method=c("CVXR", "SCS")`, since `OSQP` is the default solver, and get the same results.

```
opt_mv_scs <- optimize.portfolio(returns, pspec_mv,
                                 optimize_method = c("CVXR", "SCS"))
opt_mv_scs
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = returns, portfolio = pspec_mv, optimize_method = c("CVXR",
##     "SCS"))
##
## Optimal Weights:
##     CA   CTAG     DS     EM
## 0.3899 0.0735 0.5266 0.0100
##
## Objective Measures:
##  StdDev
## 0.01452
```

```
opt_mv$solver
```

```
## [1] "OSQP"
```

```
opt_mv_scs$solver
```

```
## [1] "SCS"
```

## 4 Maximizing Quadratic Utility

Next we demonstrate the classical quadratic utility form of Markowitz's mean-variance model, where the quadratic utility function is $\mathrm{QU}(\boldsymbol{w}) = \mu_{\mathrm{p}} - \lambda\sigma_{\mathrm{p}}^2 = \boldsymbol{\mu'w} - \lambda\boldsymbol{w'}\Sigma\boldsymbol{w}$:

$$\max_{w} \quad \hat{\boldsymbol{\mu}}'\boldsymbol{w} - \lambda\boldsymbol{w}'\Sigma\boldsymbol{w}$$

$$s.t. \quad A\boldsymbol{w} \geq b$$

Where $\hat{\boldsymbol{\mu}}$ is the estimated mean asset returns, $0 \leq \lambda < \inf$ is the risk aversion parameter, $\Sigma$ is the estimated covariance matrix of asset returns and $\boldsymbol{w}$ is the set of weights. Quadratic utility maximizes return while penalizing variance. The risk aversion parameter $\lambda$ controls how much portfolio variance is penalized, and when $\lambda = 0$ it becomes a maximum mean return problem of Section 2, and as $\lambda \to \inf$, it becomes the minimum variance problem of Section 3.

### 4.1 Portfolio Object

In this case the objectives of the portfolio should be both return and risk, and for this example we will use a risk aversion parameter $\lambda$ to be 5 by setting `risk_aversion = 5`.

```
pspec_mvo <- portfolio.spec(assets=funds)
pspec_mvo <- add.constraint(pspec_mvo, type="full_investment")
pspec_mvo <- add.constraint(pspec_mvo, type="long_only")
# Add objectives
pspec_mvo <- add.objective(portfolio = pspec_mvo, type = "return", name = "mean")
pspec_mvo <- add.objective(portfolio = pspec_mvo, type = "risk", name = "var",
                           risk_aversion = 5)
```

### 4.2 Optimization

The optimization result `opt_mvo` shows the call, optimal weights, and the objective measure. Objective measure contains quadratic utility, mean return and standard deviation.

```
opt_mvo <- optimize.portfolio(returns, pspec_mvo, optimize_method = "CVXR")
opt_mvo
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
```

```
## optimize.portfolio(R = returns, portfolio = pspec_mvo, optimize_method = "CVXR")
##
## Optimal Weights:
##     CA   CTAG     DS     EM
## 0.0000 0.0559 0.9441 0.0000
##
## Objective Measures:
## optimal value
##     -0.001042
##
##
##     mean
## 0.006484
##
##
##  StdDev
## 0.01597
```

## 5 Minimizing Expected Shortfall

Expected Shortfall(ES) is also called Conditional Value-at-Risk(CVaR) and Expected Tail Loss(ETL). The ES of a portfolio is

$$\begin{aligned} ES_\gamma(w) &= -E(r_P | r_P \le q_\gamma(\boldsymbol{w})) \\ &= -E(\boldsymbol{w'r} | \boldsymbol{w'r} \le q_\gamma(\boldsymbol{w})) \end{aligned}$$

where $r_P$ is a random return of a portfolio $P$, and $\boldsymbol{r}$ is the loss return which is negative, and $q_\gamma$ is $\gamma$-quantile and $1-\gamma$ is the tail probability. $\gamma$ is typically greater than 0.5 and the default value is 0.95, but one could also choose $\gamma < 0.5$ and the problem will take it as tail probability and use $1-\gamma$ as the quantile value.

It was shown by Rockafellar and Uryasev (2000) that the optimal minimum ES portfolio is the result of the minimization:

$$\min_{\boldsymbol{w}} ES_\gamma(\boldsymbol{w}) = \min_{\boldsymbol{w},t} F_\gamma(\boldsymbol{w}, t)$$

where

$$F_\gamma(\boldsymbol{w}, t) = -t + \frac{1}{1-\gamma} \int [t - \boldsymbol{w'r}]^+ \cdot f(\boldsymbol{r}) d\boldsymbol{r}$$

by replacing $q_\gamma$ with the free variable $t$, and with the discrete data the formula is:

$$\hat{F}_\gamma(\boldsymbol{w}, t) = -t + \frac{1}{n \cdot (1-\gamma)} \sum_{i=1}^{n} [t - \boldsymbol{w'r_i}]^+$$

Hence, the minimization of ES is equivalent to solving a linear programming problem.

The ES objective is in the form of:

$$\min_{\boldsymbol{w},t} \quad -t + (1-\gamma)^{-1} E(t - \boldsymbol{w'r_i})^+$$

where $0 < \gamma < 1$ is the quantile value, and $t$ is the value from which shortfalls are measured in the optimal solution. Many authors also use $p$ or $\alpha$ as the quantile, e.g., in Rockafellar and Uryasev (2000) and other vignettes of PortfolioAnalytics, and use $\eta$ as the risk measure variable, e.g., in Krokhmal (2007).

### 5.1 Portfolio Object

The default probability is $\gamma = 95\%$. Specific probability could be given by `arguments`.

```r
pspec_es <- portfolio.spec(assets=funds)
pspec_es <- add.constraint(pspec_es, type="full_investment")
# Add objective of minimizing ES by using the default gamma
pspec_es <- add.objective(portfolio = pspec_es, type = "risk", name = "ES")
# Add objective of minimizing ES by using the specific gamma
pspec_es_85 <- add.objective(portfolio = pspec_es, type = "risk", name = "ES",
                             arguments = list(p=0.85))
```

### 5.2 Optimization

```r
opt_es <- optimize.portfolio(returns, pspec_es, optimize_method = "CVXR")
opt_es
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = returns, portfolio = pspec_es, optimize_method = "CVXR")
##
## Optimal Weights:
##      CA    CTAG      DS      EM
##  0.0008  0.3920  0.8913 -0.2841
##
## Objective Measures:
##      ES
## 0.02087
```

```r
opt_es_85 <- optimize.portfolio(returns, pspec_es_85, optimize_method = "CVXR")
opt_es_85
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = returns, portfolio = pspec_es_85, optimize_method = "CVXR")
##
## Optimal Weights:
##      CA    CTAG      DS      EM
##  0.4873  0.2215  0.4853 -0.1942
##
## Objective Measures:
##      ES
## 0.01407
```

## 6 Minimizing Expected Quadratic Shortfall

Expected Quadratic Shortfall(EQS) is also called Second-Moment Coherent Risk Measure(SMCR). The objective to minimize EQS is in the form of:

$$\min_{\boldsymbol{w},t} \quad -t + (1-\gamma)^{-1}||(t - \boldsymbol{w'r_i})^+||_2$$

where $\gamma$ is the quantile value and $0 < \gamma < 1$, $t$ is the value from which quadratic shortfalls are measured in the optimal solution. The default probability is $\gamma = 95\%$. Minimizing EQS could be incorporated into a convex problem as a second-order cone constraints, and PortfolioAnalytics uses SCS in CVXR as the default solver for Second-Order Cone Optimization(SOCopt).

### 6.1 Portfolio Object

The default probability is $\gamma = 95\%$. Specified probability could be given by `arguments`.

```
pspec_eqs <- portfolio.spec(assets=funds)
pspec_eqs <- add.constraint(pspec_eqs, type="full_investment")
# Add objective of minimizing EQS
pspec_eqs <- add.objective(portfolio = pspec_eqs, type = "risk", name = "EQS",
                          arguments = list(p=0.95))
```

### 6.2 Optimization

```
opt_eqs <- optimize.portfolio(returns, pspec_eqs, optimize_method = "CVXR")
opt_eqs
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = returns, portfolio = pspec_eqs, optimize_method = "CVXR")
##
## Optimal Weights:
##      CA    CTAG      DS      EM
##  0.0188  0.4047  0.7450 -0.1685
##
## Objective Measures:
##     EQS
## 0.02359
```

### 6.3 Backtesting

In this example, we use CRSP data set to generate a comparative backtesting among GMV, ES and EQS portfolio. The strategy is to rebalance the portfolio at the end of each month with a rolling window of 500 days, and the performance of backtesting could be shown as a plot of cumulative returns and a plot of drawdown.

```r
## generate GMV, ES and EQS Portfolio with default gamma=0.05
pspec_sc <- portfolio.spec(assets=funds_sc)
pspec_sc <- add.constraint(pspec_sc, type="full_investment")
pspec_sc <- add.constraint(pspec_sc, type="long_only")

pspec_GMV <- add.objective(pspec_sc, type="risk", name="var")
pspec_ES <- add.objective(pspec_sc, type="risk", name="ES")
pspec_EQS <- add.objective(pspec_sc, type="risk", name="EQS")

## Optimize Portfolio at Monthly Rebalancing and 500-Day Training
bt.GMV <- optimize.portfolio.rebalancing(returns_sc, pspec_GMV,
                                          optimize_method="CVXR",
                                          rebalance_on="months",
                                          training_period=30,
                                          rolling_window=500)
bt.ES <- optimize.portfolio.rebalancing(returns_sc, pspec_ES,
                                          optimize_method="CVXR",
                                          rebalance_on="months",
                                          training_period=30,
                                          rolling_window=500)
bt.EQS <- optimize.portfolio.rebalancing(returns_sc, pspec_EQS,
                                          optimize_method="CVXR",
                                          rebalance_on="months",
                                          training_period=30,
                                          rolling_window=500)

## Extract time series of portfolio weights
wts.GMV = extractWeights(bt.GMV)
wts.GMV <- wts.GMV[complete.cases(wts.GMV),]

wts.ES = extractWeights(bt.ES)
wts.ES <- wts.ES[complete.cases(wts.ES),]

wts.EQS = extractWeights(bt.EQS)
wts.EQS <- wts.EQS[complete.cases(wts.EQS),]

## Generate monthly return
ep = endpoints(returns_sc, on= "months", k=1)
sum1 = function(x){apply(x, 2, sum)}
returnM = period.apply(returns_sc, INDEX = ep, FUN = sum1)

## Compute cumulative returns of three portfolios
GMV = Return.rebalancing(returnM, wts.GMV)
ES = Return.rebalancing(returnM, wts.ES)
EQS = Return.rebalancing(returnM, wts.EQS)

# Combine GMV, ES and EQS portfolio cumulative returns
ret.comb <- na.omit(merge(GMV, ES, EQS, all=F))
names(ret.comb) = c("GMV", "ES", "EQS")

# Compute cumulative geometric portfolios returns
R <- ret.comb
geometric = TRUE
```

```r
c.xts <- if ( geometric ) {
  cumprod(1+R)
} else {
  1 + cumsum(R)
}

# Cumulative returns panel (Peter Carl)
p <- xts::plot.xts(c.xts[,1], col="black", main = "Cumulative returns",
                   grid.ticks.lwd=1, grid.ticks.lty = "solid", grid.ticks.on = "years",
                   labels.col="grey20", cex.axis=0.8, format.labels = "%b\n%Y",
                   lty = "dotted", ylim = c(min(c.xts), max(c.xts)))
p <- xts::addSeries(c.xts[,2], on=1, lwd=2, col="dark blue", lty="dashed")
p <- xts::addSeries(c.xts[,3], on=1, lwd=2, col="dark green", lty="solid")
p <- xts::addLegend("topleft", on = 1,
                    legend.names = names(c.xts),
                    lty = c(3, 2, 1), lwd = rep(2, NCOL(c.xts)),
                    col = c("black", "dark blue", "dark green"),
                    bty = "o", box.col = "white",
                    bg=rgb(t(col2rgb("white")), alpha = 200,
                           maxColorValue = 255) )

## Drawdowns panel(Peter Carl)
d.xts <- PerformanceAnalytics::Drawdowns(R)
p <- xts::addSeries(d.xts[,1], col="black", lwd=2, main="Drawdown",
                    ylim = c(min(d.xts), 0), lty=3)
p <- xts::addSeries(d.xts[,2], on=2, lwd=2, col="dark blue", lty=2)
p <- xts::addSeries(d.xts[,3], on=2, lwd=2, col="dark green", lty=1)

## panel 1 and 2 ylim
ylim1 <- c(p$Env$ylim[[2]][1], p$Env$ylim[[2]][2])
ylim2 <- c(p$Env$ylim[[4]][1], p$Env$ylim[[4]][2])
ylim <- c(ylim1, ylim2)
# get longest drawdown dates for xts object
dt <- table.Drawdowns(R, top = 1) # just want to find the worst drawdown
dt2 <- t(dt[,c("From", "To")])
x <- as.vector(dt2[,NCOL(dt2)])
y <- as.xts(matrix(rep(ylim, length(x)),ncol=length(ylim), byrow=TRUE), order.by=as.Date(x))
i=1
p <- xts::addPolygon(y[i:(i+1),1:2], on=-1, col="lightgrey") # top panel
p <- xts::addPolygon(y[i:(i+1),3:4], on=-2, col="lightgrey") # lower panel

p
```

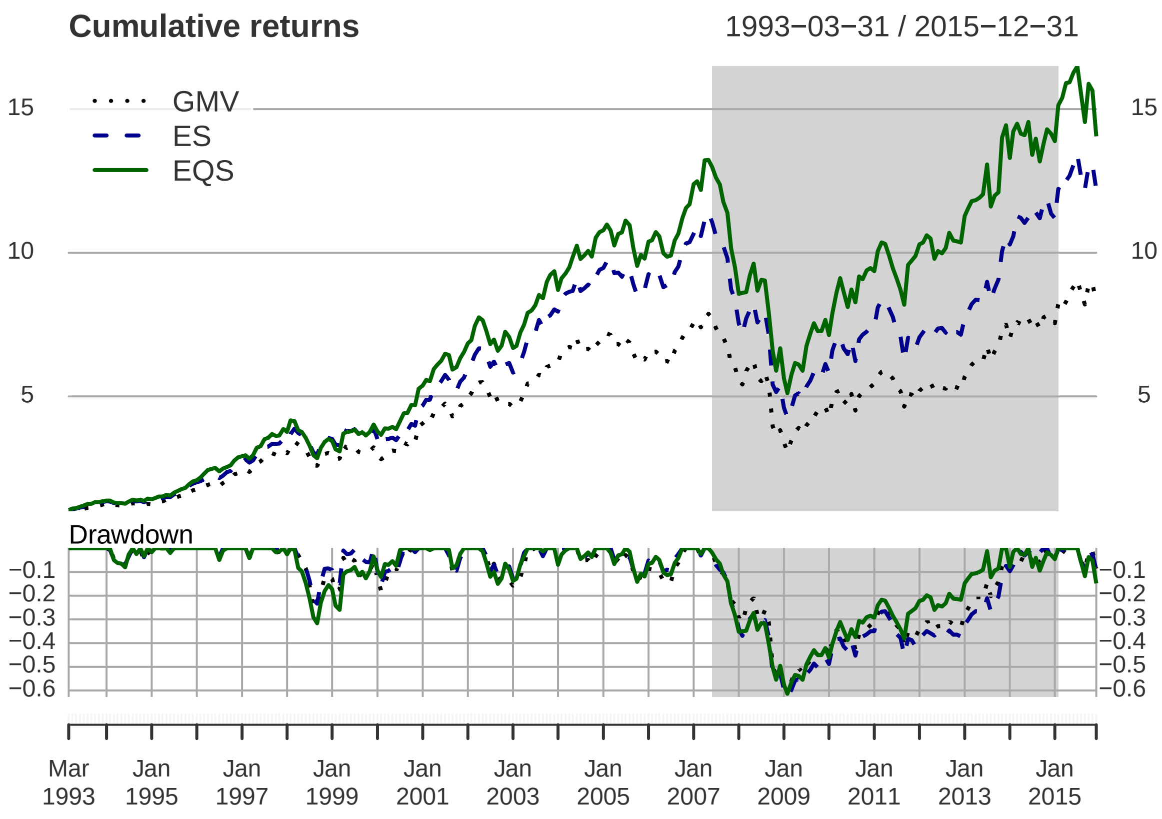

Fig 6.1

## 7 Maximizing Mean Return Per Unit Risk

There are three basic types of risk measures: variance or standard deviation, ES and EQS. The problem of maximizing mean return per unit risk can be solved in a clever way by minimizing risk with a target return constraint, as is described below. For all three of these types of problems, both return and risk objectives should be used in PortfolioAnalytics. Then for each of these three optimization problems an appropriate argument needs to be given to the `optimize.portfolio` to specify the type of problem.

### 7.1 Maximum Sharpe Ratio Portfolios

The Sharpe Ratio of a random return $r_P$ of a portfolio $P$ is defined as:

$$\frac{E(r_P) - r_f}{\sqrt{Var(r_P)}}.$$

The problem of maximizing the Sharpe Ratio can be formulated as a quadratic problem with a budget normalization constraint. It is shown in Cornuéjols, G., Peña, J., & Tütüncü, R. (2018), that this optimization problem is:

$$\begin{aligned} \underset{w}{minimize} \quad & w'\Sigma w \\ s.t. \quad (\hat{\mu} - r_f\mathbf{1})^T w &= 1 \\ \mathbf{1}^T w &= \kappa \\ \kappa &> 0 \end{aligned}$$

which has a solution$(w^*, \kappa^*)$ with $k^* \neq 0$, and the maximized Sharpe ratio given by $\tilde{w}^* = w^*/\kappa^*$.

When creating the portfolio, the argument `maxSR = TRUE` should be specified in the function `optimize.portfolio` to distinguish from the mean-variance optimization. NOTE: The default argument is `maxSR = FALSE` since the default action for dealing with both mean and var/StdDev objectives is to maximize quadratic utility.

```r
# Create portfolio object
pspec_sr <- portfolio.spec(assets=funds)
## Add constraints of maximizing Sharpe Ratio
pspec_sr <- add.constraint(pspec_sr, type="full_investment")
pspec_sr <- add.constraint(pspec_sr, type="long_only")
## Add objectives of maximizing Sharpe Ratio
pspec_sr <- add.objective(pspec_sr, type = "return", name = "mean")
pspec_sr <- add.objective(pspec_sr, type="risk", name="var")

# Optimization
optimize.portfolio(returns, pspec_sr, optimize_method = "CVXR", maxSR=TRUE)
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = returns, portfolio = pspec_sr, optimize_method = "CVXR",
##     maxSR = TRUE)
##
## Optimal Weights:
##     CA   CTAG     DS     EM
## 0.2036 0.2422 0.5542 0.0000
##
## Objective Measures:
##     mean
## 0.005797
##
##
##  StdDev
## 0.01313
##
##
## Sharpe Ratio
##       0.4414
```

## 7.2 Maximum ES ratio Portfolios

The ES ratio(ESratio), which is also called STARR in PortfolioAnalytics, is defined as:

$$\frac{E(r_P) - r_f}{ES_\alpha(r_P)}$$

Similar to maximizing Sharpe Ratio, the problem maximizing the ES ratio can be formulated as a minimizing ES problem with a budget normalization constraint.

When creating the portfolio, both return and ES objectives should be given. When solving the problem, the default argument `ESratio=TRUE` in the function `optimize.portfolio` specify the problem type. We note that this argument is equivalent to `maxSTARR=TRUE`, which is used in other vignettes. If one of these two arguments is specified as FALSE, the action will be to minimize ES ignoring the return objective.

```r
# Create portfolio object
pspec_ESratio <- portfolio.spec(assets=funds)
## Add constraints of maximizing return per unit ES
pspec_ESratio <- add.constraint(pspec_ESratio, type="full_investment")
pspec_ESratio <- add.constraint(pspec_ESratio, type="long_only")
## Add objectives of maximizing return per unit ES
pspec_ESratio <- add.objective(pspec_ESratio, type = "return", name = "mean")
pspec_ESratio <- add.objective(pspec_ESratio, type="risk", name="ES")

# Optimization
optimize.portfolio(returns, pspec_ESratio, optimize_method = "CVXR", ESratio=TRUE)
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = returns, portfolio = pspec_ESratio, optimize_method = "CVXR",
##     ESratio = TRUE)
##
## Optimal Weights:
##     CA   CTAG     DS     EM
## 0.0000 0.4288 0.5712 0.0000
##
## Objective Measures:
##     mean
## 0.005565
##
##
##      ES
## 0.02317
##
##
## ES ratio
##   0.2402
```

## 7.3 Maximum EQS ratio Portfolios

The EQS ratio of a random return $r_P$ of a portfolio $P$ is defined as:

$$\frac{E(r_P) - r_f}{EQS_\alpha(r_P)}$$

Similar to maximizing Sharpe Ratio, the problem maximizing EQS ratio could be formulated as a minimizing EQS problem with a budget normalization constraint.

When creating the portfolio, both return and EQS objectives should be given. The argument `EQSratio=` is used to specify the problem type and the default value is `EQSratio=TRUE`. If `EQSratio=FALSE`, the action will be to minimize EQS ignoring the return objective. The default $\alpha = 0.95$, and it can be specified by `arguments`.

```
# Create portfolio object
pspec_EQSratio <- portfolio.spec(assets=funds)
## Add constraints of maximizing return per unit EQS
pspec_EQSratio <- add.constraint(pspec_EQSratio, type="full_investment")
pspec_EQSratio <- add.constraint(pspec_EQSratio, type="long_only")
## Add objectives of maximizing return per unit EQS
pspec_EQSratio <- add.objective(pspec_EQSratio, type = "return", name = "mean")
pspec_EQSratio <- add.objective(pspec_EQSratio, type="risk", name="EQS",
                                arguments = list(p=0.95))

# Optimization
optimize.portfolio(returns, pspec_EQSratio, optimize_method = "CVXR", EQSratio=TRUE)
```

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = returns, portfolio = pspec_EQSratio, optimize_method = "CVXR",
##     EQSratio = TRUE)
##
## Optimal Weights:
##     CA   CTAG     DS     EM
## 0.0000 0.4518 0.5482 0.0000
##
## Objective Measures:
##     mean
## 0.005509
##
##
##     EQS
## 0.02804
##
##
## EQS ratio
##    0.1965
```

## 8 Efficient Frontier

We generate efficient frontiers with MVO, mean-ES and mean-EQS portfolios by using 30 small cap stocks from CRSP data set. We can use `create.EfficientFrontier` to calculate the mean value and risk value for the frontier, then use `chart.EfficientFrontier` to draw the frontier.

```
meanvar.ef <- create.EfficientFrontier(R=returns_sc, portfolio=pspec_sc, type="mean-StdDev")
```

```
## Registered S3 method overwritten by 'ROI':
##   method           from
##   print.constraint PortfolioAnalytics
```

```r
meanvar.ef
```

```
## **************************************************
## PortfolioAnalytics Efficient Frontier
## **************************************************
##
## Call:
## create.EfficientFrontier(R = returns_sc, portfolio = pspec_sc,
##     type = "mean-StdDev")
##
## Efficient Frontier Points: 25
##
## **************************************************
## PortfolioAnalytics Portfolio Specification
## **************************************************
##
## Call:
## portfolio.spec(assets = funds_sc)
##
## Number of assets: 30
## Asset Names
##  [1] "TGNA" "AVP"  "PBI"  "THC"  "AVY"  "HAS"  "TSS"  "SPXC" "R"    "HP"
## More than 10 assets, only printing the first 10
##
## Constraints
## Enabled constraint types
##      - full_investment
##      - long_only
```

```r
meanvar.ef$frontier[, 1:2]
```

```
##                   mean     StdDev
## result.1  0.0004915235 0.01152895
## result.2  0.0005162476 0.01155483
## result.3  0.0005409717 0.01162536
## result.4  0.0005656957 0.01173356
## result.5  0.0005904198 0.01188021
## result.6  0.0006151439 0.01206660
## result.7  0.0006398679 0.01229138
## result.8  0.0006645920 0.01255253
## result.9  0.0006893160 0.01285681
## result.10 0.0007140401 0.01321427
## result.11 0.0007387642 0.01361810
## result.12 0.0007634882 0.01406477
## result.13 0.0007882123 0.01455102
## result.14 0.0008129364 0.01510969
## result.15 0.0008376604 0.01577613
## result.16 0.0008623845 0.01664155
## result.17 0.0008871085 0.01801916
## result.18 0.0009118326 0.01988267
## result.19 0.0009365567 0.02210956
## result.20 0.0009612807 0.02460134
## result.21 0.0009860048 0.02730151
## result.22 0.0010107289 0.03166859
## result.23 0.0010354529 0.03809384
## result.24 0.0010601770 0.04571756
## result.25 0.0010849010 0.05403484
```

Then the Sharpe ratio could be calculated and the maximum Sharpe ratio could be found.

```
sr = meanvar.ef$frontier[, 1]/meanvar.ef$frontier[, 2]
cat("maximum Sharpe ratio:", max(sr))
```

```
## maximum Sharpe ratio: 0.05428375
```

```
cat("mean of the maximum SR portfolio:", meanvar.ef$frontier[, 1][sr == max(sr)])
```

```
## mean of the maximum SR portfolio: 0.0007634882
```

```
cat("StdDev of the maximum SR portfolio:", meanvar.ef$frontier[, 2][sr == max(sr)])
```

```
## StdDev of the maximum SR portfolio: 0.01406477
```

Then generate a plot of mean-StdDev efficient frontier and identify the GMV portfolio in blue point.

```
# MVO
## Add mean return objective
pspec_GMV <- add.objective(portfolio=pspec_GMV, type="return", name="mean")
opt_GMV <- optimize.portfolio(returns_sc, pspec_GMV, optimize_method = "CVXR", trace = TRUE)
chart.EfficientFrontier(opt_GMV, match.col="StdDev", type="l",
                        chart.assets = FALSE, main="Mean-StdDev Efficient Frontier",
                        RAR.text="Sharpe Ratio", pch=4)
```

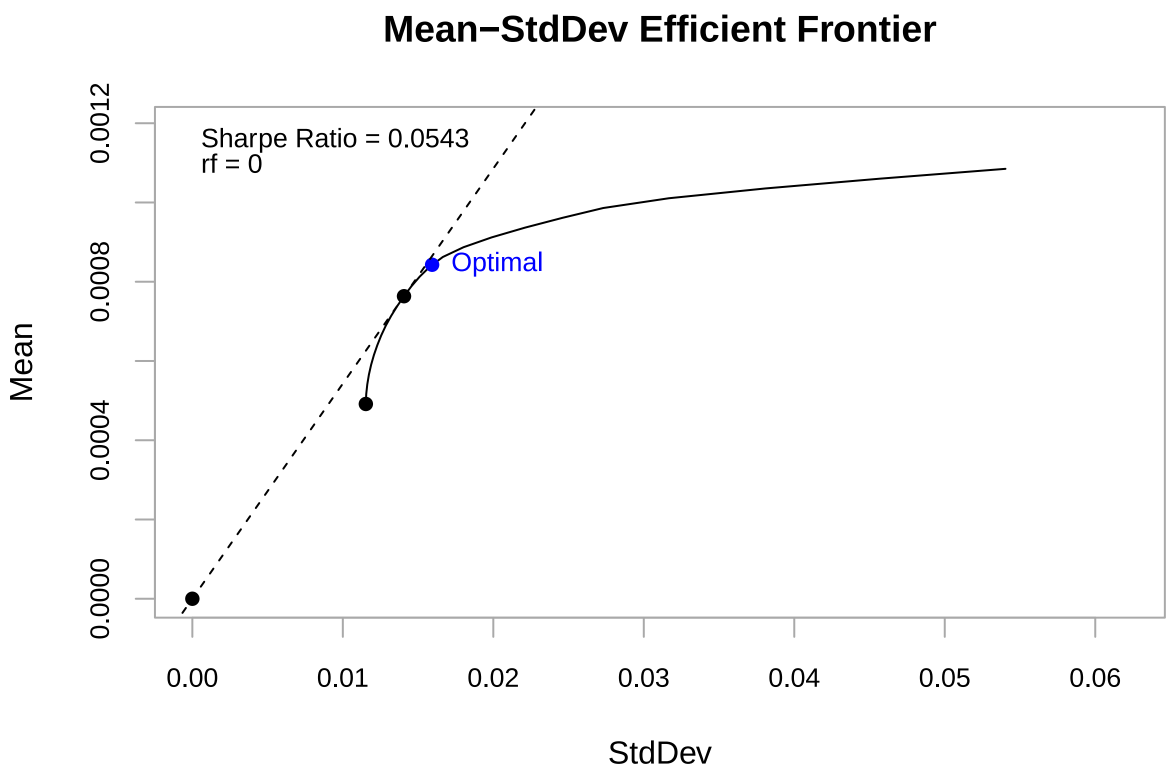
Mean−StdDev Efficient Frontier
Sharpe Ratio = 0.0543
rf = 0
Optimal
Mean
0.0000
0.0004
0.0008
0.0012
StdDev
0.00
0.01
0.02
0.03
0.04
0.05
0.06

Fig 8.1

```r
# mean-ES
meanetl.ef <- create.EfficientFrontier(R=returns_sc, portfolio=pspec_sc, type="mean-ES")
chart.EfficientFrontier(meanetl.ef, match.col="ES", type="l", col="blue",
                        chart.assets = FALSE, main="Mean-ES Efficient Frontier",
                        RAR.text="ES ratio", pch=4)
```

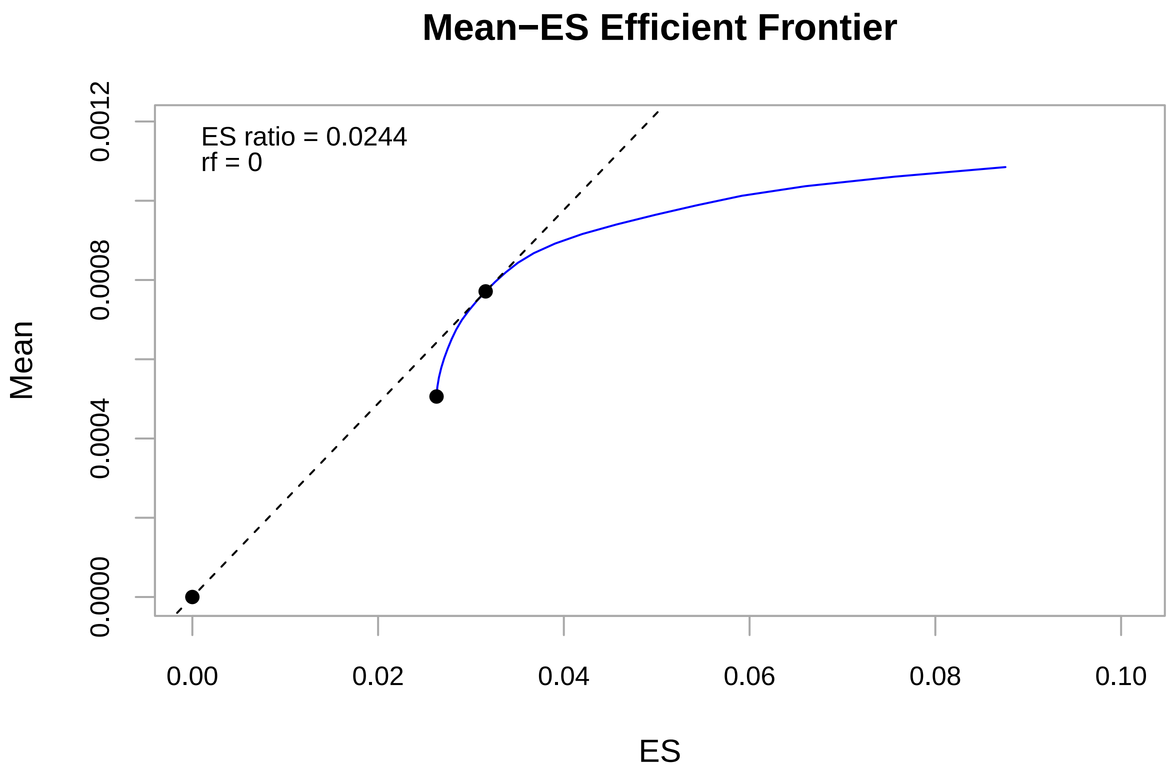

Fig 8.2

```r
# mean-EQS
meaneqs.ef <- create.EfficientFrontier(R=returns_sc, portfolio=pspec_sc, type="mean-EQS")
chart.EfficientFrontier(meaneqs.ef, match.col="EQS", type="l", col="blue",
                        chart.assets = FALSE, main="Mean-EQS Efficient Frontier",
                        RAR.text="EQS ratio", pch=4)
```

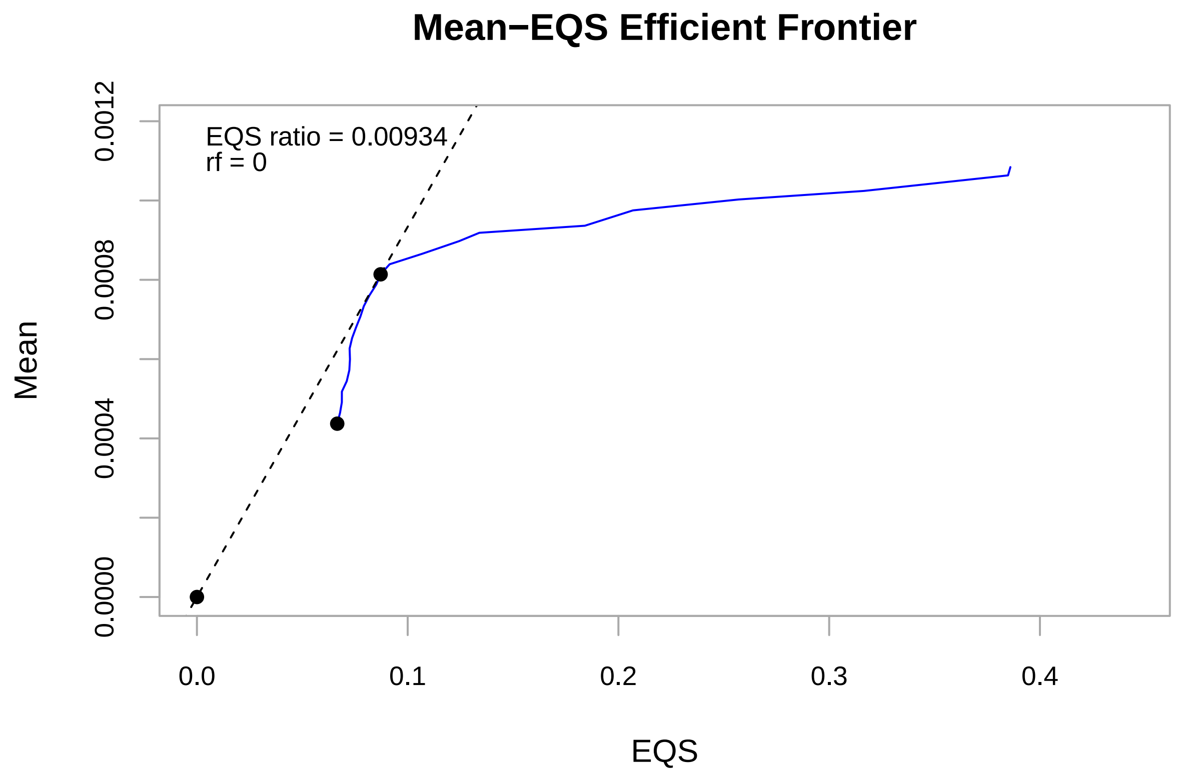

Fig 8.3