CVXR for PortfolioAnalytics

Xinran Zhao

August 18, 2024

Contents

1 Introduction

CVXR is an R package that provides an object-oriented modeling language for convex optimization, including the Second-Order Cone Optimization(SOCopt) required to minimize Coherent Second Moment(CSM) problem, which is not supported by other solvers in PortfolioAnalytics. Hence, CVXR is a significant extension of PortfolioAnalytics.

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. For example, 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 functionality and use of PortfolioAnalytics, users are recommended to read the Vignette *Introduction to PortfolioAnalytics* first.

The R code `demo_cvxrPortfolioAnalytics.R` in the PortfolioAnalytics `demo` folder, reproduces the results in this Vignette.

2 Getting Started

2.1 Load Packages

Load the necessary packages.

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

2.2 Solvers

The website https://cvxr.rbind.io/ shows that CVXR currently supports the use of 9 solvers, some of which are commercial (CBC, CPLEX, GUROBI, MOSEK)[1] and the others are open source(GLPK, GLPK_MI, OSQP, SCS, ECOS).

The PortfolioAnalytics package provides the following two main wrapper functions for constrainted optimization of portfolios using a wide variety of methods:

`optimize.portfolio(R =, portfolio =, optimize_method =)`

`optimize.portfolio.rebalancing(R =, portfolio =, optimize_method =, rebalance_on =, training_period =, rolling_window =)`

Different solvers support different types of portfolio optimization problems, which should be specified by the argument `optimize_method`. The `optimize_method=c("CVXR", {CVXRsolver})` argument of the function `optimize.portfolio` and `optimize.portfolio.rebalancing` allows the user to specify the solver to use with CVXR. If the argument is `optimize_method="CVXR"`, the default solver for QP type portfolio optimization problems, such as minimum variance portfolio optimization is OSQP, and the default solver for LP and SOCP type portfolio optimizations, such as maximum mean return, “robust portfolio optimization” to control for alpha uncertainty, and Coherent Second Moment (CSM) portfolio optimization, is SCS.

[1]MOSEK ,GUROBI, and CPLEX require licenses to use, but free academic licenses are available for all three.

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

2.3 Data

The edhec data set from the PerformanceAnalytics package is used as example data for examples in Section 3 to Section 8. The edhec data set is an xts object that contains monthly returns for 13 hedge fund style indexes from 1997-01 to 2019-11. We use the edhec data of the last 5 years as the example data to show how to use the code.

```
data(edhec)
# Use edhec for a returns object
ret_edhec <- tail(edhec, 60)
colnames(ret_edhec) <- c("CA", "CTAG", "DS", "EM", "EMN", "ED", "FIA",
                         "GM", "LSE", "MA", "RV", "SS", "FF")
print(head(ret_edhec, 5))
#>                  CA    CTAG      DS      EM     EMN      ED     FIA      GM
#> 2014-12-31 -0.0066  0.0088 -0.0089 -0.0220  0.0013 -0.0022 -0.0035 -0.0004
#> 2015-01-31  0.0013  0.0399 -0.0155 -0.0034  0.0048 -0.0104 -0.0004  0.0229
#> 2015-02-28  0.0121 -0.0029  0.0185  0.0162  0.0020  0.0270  0.0086  0.0070
#> 2015-03-31  0.0021  0.0097  0.0028  0.0039  0.0080  0.0043  0.0021  0.0101
#> 2015-04-30  0.0157 -0.0232  0.0071  0.0378 -0.0029  0.0113  0.0051 -0.0091
#>                 LSE     MA      RV      SS     FF
#> 2014-12-31  0.0012 0.0032 -0.0016  0.0033 0.0021
#> 2015-01-31 -0.0009 0.0004  0.0025  0.0109 0.0017
#> 2015-02-28  0.0252 0.0139  0.0150 -0.0385 0.0171
#> 2015-03-31  0.0036 0.0056  0.0033  0.0006 0.0069
#> 2015-04-30  0.0055 0.0066  0.0069 -0.0143 0.0026
```

```
# Get a character vector of the asset names
fund_edhec <- colnames(ret_edhec)
```

tsPlotMP is a function of R package PCRA which can plot time series for the return data.

```r
tsPlotMP(ret_edhec, layout = c(2, 7), main = "Time Series Plot of Edhec Return")
```

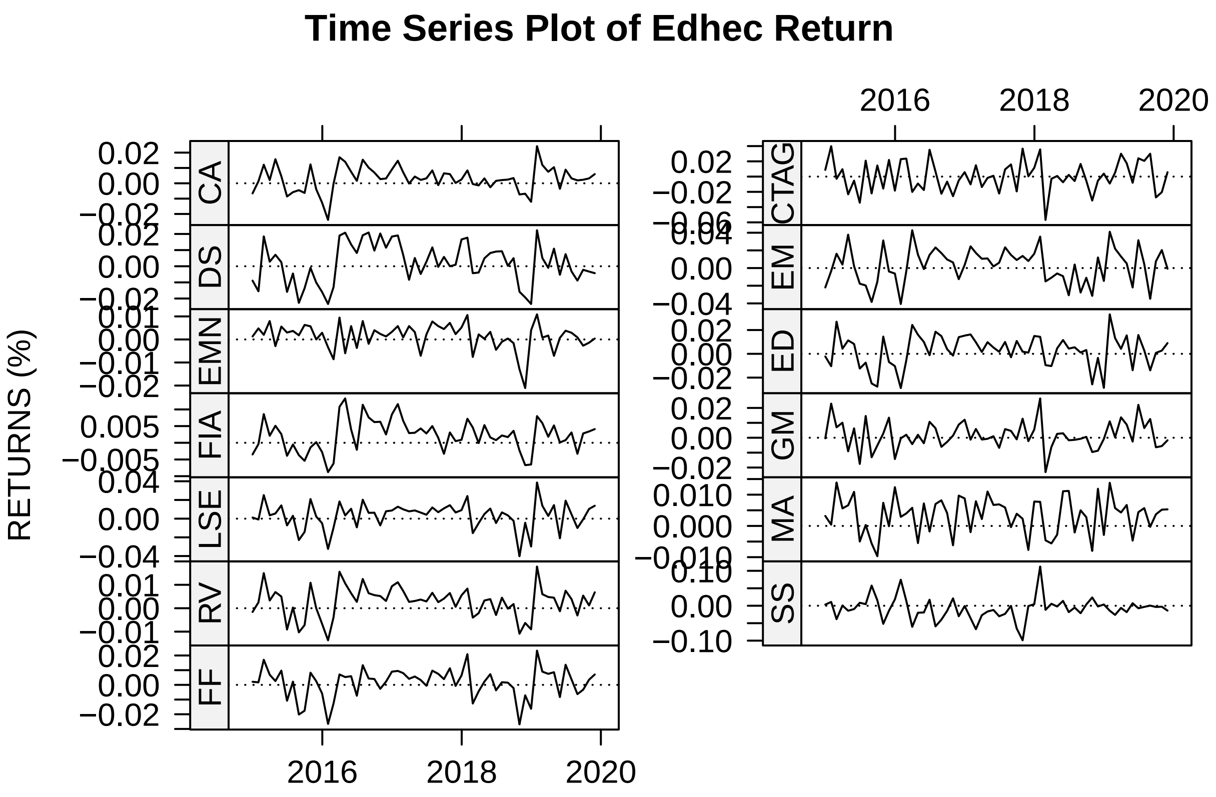


Fig 2.1

2.4 Optimization Problems

In this Vignette, all mean vectors and covariance matrices in the optimization formula will use standard sample based estimates. 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 found in the vignette Ross (2018) *Introduction to PortfolioAnalytics*.

This vignette will be organized by objective type and provide some visual examples.

3 Maximizing Mean Return

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

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

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

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

3.1 Portfolio Object

The first step in setting up a model is to create the portfolio object, which contains the form of the constraints, and the objective specifications. In the following we create full-investment and box constraints specifications, and a maximum return objective specification.

```
# Create portfolio object
pspec_maxret <- portfolio.spec(assets = fund_edhec)
# 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 = rep(0.02, 13),
                               max = c(rep(0.15, 8), rep(0.1, 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 = fund_edhec)
#>
#> Number of assets: 13
#> Asset Names
#>  [1] "CA"   "CTAG" "DS"   "EM"   "EMN"  "ED"   "FIA"  "GM"   "LSE"  "MA"
#> More than 10 assets, only printing the first 10
#>
#> Constraints
#> Enabled constraint types
#>      - full_investment
#>      - box
#>
#> Objectives:
#> Enabled objective names
#>      - mean
```

3.2 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 `SCS`.

```
# Run the optimization with default solver
opt_maxret <- optimize.portfolio(R = ret_edhec, portfolio = pspec_maxret,
                                 optimize_method = "CVXR", trace = TRUE)
opt_maxret
#> ***********************************
#> PortfolioAnalytics Optimization
#> ***********************************
#>
```

```
#> Call:
#> optimize.portfolio(R = ret_edhec, portfolio = pspec_maxret, optimize_method = "CVXR",
#>     trace = TRUE)
#>
#> Optimal Weights:
#>     CA   CTAG     DS     EM    EMN     ED    FIA     GM    LSE     MA     RV
#> 0.1502 0.0202 0.0202 0.1502 0.0202 0.1502 0.1277 0.0202 0.1002 0.1002 0.1002
#>     SS     FF
#> 0.0202 0.0202
#>
#> Objective Measures:
#>     mean
#> 0.002497
```

```
opt_maxret$solver
#> [1] "SCS"
```

```
# Run the optimization with specific solver
opt_maxret_glpk <- optimize.portfolio(R = ret_edhec, portfolio = pspec_maxret,
                            optimize_method = c("CVXR", "GLPK"), trace = TRUE)
opt_maxret_glpk$solver
#> [1] "GLPK"
```

3.3 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 = ret_edhec, 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"
```

4 Minimizing Variance

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

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

subject to portfolio managers' desired constraints, where Σ is the estimated covariance matrix of asset returns and $\boldsymbol{w}$ is the set of portfolio weights. It is a quadratic problem.

4.1 Global Minimum Variance Portfolio

4.1.1 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 (GMV) portfolio.

```r
# Create portfolio object
pspec_gmv <- portfolio.spec(assets = fund_edhec)
# 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")
```

4.1.2 Optimization

```r
opt_gmv <- optimize.portfolio(ret_edhec, pspec_gmv, optimize_method = "CVXR")
opt_gmv
#> ***********************************
#> PortfolioAnalytics Optimization
#> ***********************************
#>
#> Call:
#> optimize.portfolio(R = ret_edhec, portfolio = pspec_gmv, optimize_method = "CVXR")
#>
#> Optimal Weights:
#>      CA    CTAG      DS      EM     EMN      ED     FIA      GM     LSE      MA
#>  0.0691  0.0141 -0.1101 -0.0199  0.1677 -0.1318  0.5433  0.0404  0.0184  0.3427
#>      RV      SS      FF
#>  0.3225  0.0178 -0.2743
#>
#> Objective Measures:
#>   StdDev
#> 0.002011
```

As this example illustrates, a global minimum variance portfolio with only a full-investment constraint can have short positions.

4.2 Long-Only and Group 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.

```r
# 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(2:12),
                                         groupC=13),
                           group_min = c(0, 0.05, 0.05),
                                group_max = c(0.4, 0.8, 0.5))
pspec_mv <- add.constraint(pspec_mv, type = "return", return_target = 0.003)
pspec_mv
#> **************************************************
#> PortfolioAnalytics Portfolio Specification
#> **************************************************
#>
#> Call:
#> portfolio.spec(assets = fund_edhec)
#>
#> Number of assets: 13
#> Asset Names
#>  [1] "CA"   "CTAG" "DS"   "EM"   "EMN"  "ED"   "FIA"  "GM"   "LSE"  "MA"
#> More than 10 assets, only printing the first 10
#>
#> Constraints
#> Enabled constraint types
#>      - full_investment
#>      - long_only
#>      - group
#>      - return
#>
#> Objectives:
#> Enabled objective names
#>      - var
```

```
# optimization
opt_mv <- optimize.portfolio(ret_edhec, pspec_mv, optimize_method = "CVXR")
opt_mv
#> ***********************************
#> PortfolioAnalytics Optimization
#> ***********************************
#>
#> Call:
#> optimize.portfolio(R = ret_edhec, portfolio = pspec_mv, optimize_method = "CVXR")
#>
#> Optimal Weights:
#>     CA   CTAG     DS     EM    EMN     ED    FIA     GM    LSE     MA     RV
#> 0.1500 0.0000 0.0000 0.0000 0.0000 0.0000 0.1989 0.0000 0.0000 0.6011 0.0000
#>     SS     FF
#> 0.0000 0.0500
#>
#> Objective Measures:
#>   StdDev
#> 0.005052
```

Compared to Section 4.1, the result shows that the more constraints, the larger the optimal value may be. But this example is closer to the real situation. The optimal weights show that the first group constraint is not binding, but the second one is binding with FIA plus MA at the upper bound of 0.8, and the third group constraint is binding at the lower bound with FF.

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", "ECOS")`, since `OSQP` is the default solver, and get the very similar results.

```
opt_mv_ecos <- optimize.portfolio(ret_edhec, pspec_mv, optimize_method = c("CVXR", "ECOS"))
opt_mv_ecos
#> ***********************************
#> PortfolioAnalytics Optimization
#> ***********************************
#>
#> Call:
#> optimize.portfolio(R = ret_edhec, portfolio = pspec_mv, optimize_method = c("CVXR",
#>     "ECOS"))
#>
#> Optimal Weights:
#>     CA   CTAG     DS     EM    EMN     ED    FIA     GM    LSE     MA     RV
#> 0.1500 0.0000 0.0000 0.0000 0.0000 0.0000 0.1989 0.0000 0.0000 0.6011 0.0000
#>     SS     FF
#> 0.0000 0.0500
#>
#> Objective Measures:
#>   StdDev
#> 0.005053
```

```
opt_mv$solver
#> [1] "OSQP"
```

```
opt_mv_ecos$solver
#> [1] "ECOS"
```

5 Maximizing Quadratic Utility

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

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

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

Here $\boldsymbol{\mu}$ is the vector of estimated mean asset returns, $0 \leq \lambda < \inf$ is the risk aversion parameter, Σ is the estimated covariance matrix of asset returns, and $\boldsymbol{w}$ is the vector of weights. Quadratic utility maximizes return while penalizing variance risk. The risk aversion parameter λ controls how much portfolio variance is penalized, and when $\lambda = 0$ it becomes a maximum mean return problem of Section 3, and as $\lambda \to \inf$, it becomes the global minimum variance problem of Section 4.

5.1 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 λ to be 20 by setting `risk_aversion = 20`.

```
pspec_mvo <- portfolio.spec(assets = fund_edhec)
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 = 20)
```

5.2 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(ret_edhec, pspec_mvo, optimize_method = "CVXR")
opt_mvo
#> ***********************************
#> PortfolioAnalytics Optimization
#> ***********************************
#>
#> Call:
#> optimize.portfolio(R = ret_edhec, portfolio = pspec_mvo, optimize_method = "CVXR")
#>
#> Optimal Weights:
#>     CA   CTAG     DS     EM    EMN     ED    FIA     GM    LSE     MA     RV
#> 0.1502 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.8498 0.0000
#>     SS     FF
#> 0.0000 0.0000
#>
#> Objective Measures:
#> optimal value
#>    -0.0001352
#>
#>
#>     mean
#> 0.003327
#>
#>
#>   StdDev
#> 0.005583
```

6 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

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

where r_P is a random return of a portfolio P, and $\boldsymbol{r}$ is the loss return which is negative, and q_γ is γ-quantile and γ is usually a "tail" probability such as 0.01, 0.05, in which case ES is a tail risk measure. But one could also choose $\gamma = 0.25$ or $\gamma = 0.5$, in which case ES is just a "downside" risk measure, and if $\gamma > 0.5$, the problem will take $1-\gamma$ as the tail probability.

It was shown by Rockafellar, Uryasev, et al. (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

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

by replacing q_γ with the free variable t, and with the discrete data the formula is:

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

The positive part function, $[t - \boldsymbol{w'r_i}]^+$, can easily be converted to a collection of linear constraints, hence, the minimization of ES is equivalent to solving a linear programming problem.

The ES minimization problem is

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

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

6.1 Portfolio Object

The default probability is $\gamma = 5\%$. Specific probability could be given by `arguments`.

```r
pspec_es <- portfolio.spec(assets = fund_edhec)
pspec_es <- add.constraint(pspec_es, type = "full_investment")
pspec_es <- add.constraint(pspec_es, type = "long_only")
# 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=0.1
pspec_es_1 <- add.objective(portfolio = pspec_es, type = "risk", name = "ES",
                            arguments = list(p=0.1))
```

6.2 Optimization

Notice that if `optimize_method` is not declared, the default solver is `DEoptim`. But we highly recommend using `ROI` solvers or `CVXR` solvers, because they provide accurate theoretical solutions for minES problem.

```r
# GMES with default gamma=0.05
opt_es <- optimize.portfolio(ret_edhec, pspec_es, optimize_method = "CVXR")
opt_es
#> ***********************************
#> PortfolioAnalytics Optimization
#> ***********************************
#>
#> Call:
#> optimize.portfolio(R = ret_edhec, portfolio = pspec_es, optimize_method = "CVXR")
#>
#> Optimal Weights:
#>     CA   CTAG     DS     EM    EMN     ED    FIA     GM    LSE     MA     RV
#> 0.0000 0.0000 0.0000 0.0000 0.0890 0.0000 0.4184 0.0000 0.0000 0.4348 0.0000
#>     SS     FF
#> 0.0578 0.0000
#>
```

```
#> Objective Measures:
#>       ES
#> 0.003456
```

```
# GMES with specific gamma=0.1
opt_es_1 <- optimize.portfolio(ret_edhec, pspec_es_1, optimize_method = "CVXR")
opt_es_1
#> ***********************************
#> PortfolioAnalytics Optimization
#> ***********************************
#>
#> Call:
#> optimize.portfolio(R = ret_edhec, portfolio = pspec_es_1, optimize_method = "CVXR")
#>
#> Optimal Weights:
#>     CA   CTAG     DS     EM    EMN     ED    FIA     GM    LSE     MA     RV
#> 0.0000 0.0000 0.0000 0.0000 0.2007 0.0000 0.3567 0.0000 0.0000 0.3861 0.0000
#>     SS     FF
#> 0.0564 0.0000
#>
#> Objective Measures:
#>       ES
#> 0.002952
```

7 Minimizing Coherent Second Moment

Coherent Second Moment(CSM) is also called Second-Moment Coherent Risk Measure(SMCR) in some situations, for example, in Krokhmal (2007). The objective to minimize CSM is in the form of:

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

where γ is the tail probability and $0 < \gamma < 1$, t is the value from which quadratic shortfalls are measured in the optimal solution. The default probability is $\gamma = 5\%$. Minimizing CSM 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).

7.1 Portfolio Object

The default probability is $\gamma = 5\%$. Specified probability could be given by `arguments`.

```
pspec_csm <- portfolio.spec(assets = fund_edhec)
pspec_csm <- add.constraint(pspec_csm, type = "full_investment")
pspec_csm <- add.constraint(pspec_csm, type = "long_only")
# Add objective of minimizing CSM
pspec_csm <- add.objective(portfolio = pspec_csm, type = "risk", name = "CSM",
                           arguments = list(p=0.05))
```

7.2 Optimization

```
opt_csm <- optimize.portfolio(ret_edhec, pspec_csm, optimize_method = "CVXR")
opt_csm
#> ***********************************
#> PortfolioAnalytics Optimization
#> ***********************************
#>
#> Call:
#> optimize.portfolio(R = ret_edhec, portfolio = pspec_csm, optimize_method = "CVXR")
#>
#> Optimal Weights:
#>     CA   CTAG     DS     EM    EMN     ED    FIA     GM    LSE     MA     RV
#> 0.0000 0.0000 0.0000 0.0000 0.1966 0.0000 0.5136 0.0000 0.0000 0.2263 0.0000
#>     SS     FF
#> 0.0634 0.0000
#>
#> Objective Measures:
#>      CSM
#> 0.003693
```

8 Maximizing Mean Return Per Unit Risk

There are three basic types of risk measures: variance or standard deviation, ES and CSM. 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, as we describe below.

8.1 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 Cornuejols, Pena, and Tutuncu (2018), that this optimization problem can be formulated as the equivalent optimization:

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

which has a solution(w^*, κ^*) 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.

```
# Create portfolio object
pspec_sr <- portfolio.spec(assets = fund_edhec)
## 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(ret_edhec, pspec_sr, optimize_method = "CVXR", maxSR = TRUE)
#> ***********************************
#> PortfolioAnalytics Optimization
#> ***********************************
#>
#> Call:
#> optimize.portfolio(R = ret_edhec, portfolio = pspec_sr, optimize_method = "CVXR",
#>     maxSR = TRUE)
#>
#> Optimal Weights:
#>     CA   CTAG     DS     EM    EMN     ED    FIA     GM    LSE     MA     RV
#> 0.0000 0.0029 0.0000 0.0000 0.1026 0.0000 0.4058 0.0000 0.0000 0.4865 0.0000
#>     SS     FF
#> 0.0022 0.0000
#>
#> Objective Measures:
#>     mean
#> 0.002658
#>
#>
#>   StdDev
#> 0.003975
#>
#>
#> Sharpe Ratio
#>       0.6687
```

8.2 Maximum ES ratio Portfolios

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

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

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. The default $\gamma = 0.05$, and it can be specified by `arguments`. When solving the problem, the default argument `ESratio = TRUE` in the function `optimize.portfolio` specifies 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 = fund_edhec)
## 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",
                               arguments = list(p=0.05))

# Optimization
optimize.portfolio(ret_edhec, pspec_ESratio, optimize_method = "CVXR", ESratio = TRUE)
#> ***********************************
#> PortfolioAnalytics Optimization
#> ***********************************
#>
#> Call:
#> optimize.portfolio(R = ret_edhec, portfolio = pspec_ESratio,
#>     optimize_method = "CVXR", ESratio = TRUE)
#>
#> Optimal Weights:
#>     CA   CTAG     DS     EM    EMN     ED    FIA     GM    LSE     MA     RV
#> 0.0000 0.0000 0.0000 0.0000 0.2277 0.0000 0.3194 0.0000 0.0000 0.4369 0.0000
#>     SS     FF
#> 0.0160 0.0000
#>
#> Objective Measures:
#>     mean
#> 0.002353
#>
#>
#>       ES
#> 0.004485
#>
#>
#> ES ratio
#>   0.5247
```

8.3 Maximum CSM ratio Portfolios

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

$$\frac{E(r_P) - r_f}{CSM_\gamma(r_P)}$$

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

When creating the portfolio, both return and CSM objectives should be given. The argument `CSMratio =` is used to specify the problem type and the default value is `CSMratio = TRUE`. If `CSMratio = FALSE`, the action will be to minimize CSM ignoring the return objective. The default $\gamma = 0.05$, and it can be specified by `arguments`.

```
# Create portfolio object
pspec_CSMratio <- portfolio.spec(assets = fund_edhec)
## Add constraints of maximizing return per unit CSM
pspec_CSMratio <- add.constraint(pspec_CSMratio, type = "full_investment")
pspec_CSMratio <- add.constraint(pspec_CSMratio, type = "long_only")
## Add objectives of maximizing return per unit CSM
pspec_CSMratio <- add.objective(pspec_CSMratio, type = "return", name = "mean")
pspec_CSMratio <- add.objective(pspec_CSMratio, type = "risk", name = "CSM",
                                arguments = list(p=0.05))

# Optimization
optimize.portfolio(ret_edhec, pspec_CSMratio, optimize_method = "CVXR", CSMratio = TRUE)
#> ***********************************
#> PortfolioAnalytics Optimization
#> ***********************************
#>
#> Call:
#> optimize.portfolio(R = ret_edhec, portfolio = pspec_CSMratio,
#>     optimize_method = "CVXR", CSMratio = TRUE)
#>
#> Optimal Weights:
#>     CA   CTAG     DS     EM    EMN     ED    FIA     GM    LSE     MA     RV
#> 0.0000 0.0000 0.0000 0.0000 0.1051 0.0000 0.6408 0.0000 0.0000 0.2336 0.0000
#>     SS     FF
#> 0.0205 0.0000
#>
#> Objective Measures:
#>     mean
#> 0.002178
#>
#>
#>      CSM
#> 0.004276
#>
#>
#> CSM ratio
#>    0.5094
```

9 Performance of minCSM, minES and minVar Portfolios

CVXR solvers provide the Second-Order Cone Optimization (SOCopt) capability required to compute minCSM portfolios. In this Section we use this capability to compute a minCSM portfolio and compare its performance that of minES and minVar portfolios. For these computations, we use a CRSP® daily returns data set to generate these optimal portfolios and show their performance by plotting cumulative gross returns and efficient frontiers.[2] Specifically, the data set consists of 30 smallcap stocks from 1993-01 to 2015-12, contained in the stocksCRSPdaily data object in the PCRA package available on CRAN. The 30 smallcaps are the 30 of the 106 smallcap stocks with the largest market capitalization.

The entire back-testing part is relatively slow, and takes most of the time in this Vignette. To give a sense of typical times, I provide the computing times for my MacBook Air with M2, 8-Core CPU, 8-Core GPU

[2]CRSP® stands for the Center for Research in Security Prices, LLC.

and 16-core Neural Engine. The backtesting in Sections 9.1 and Section 9.2 takes about 3 minutes each to run, and the average running time for generating efficient frontiers in Section 9.3 is about 30 seconds.

```
# Get daily returns of the 30 smallcap stocks
library(PCRA)
stocksCRSPdaily <- getPCRAData(dataset = "stocksCRSPdaily")

smallcapTS <- selectCRSPandSPGMI(
  periodicity = "daily",
  stockItems = c("Date", "TickerLast", "CapGroupLast", "Return"),
  factorItems = NULL,
  subsetType = "CapGroupLast",
  subsetValues = "SmallCap",
  outputType = "xts")

# find top 30 small cap stocks based on the market capitalization
smallcapDT <- factorsSPGMI[CapGroupLast == "SmallCap"]
scSize <- smallcapDT[, mean(LogMktCap), by = "TickerLast"]
names(scSize)[2] <- "Size"
scSize <- scSize[order(scSize$Size, decreasing = TRUE),]
sc30largest <- scSize[,TickerLast][1:30]

# daily return of top 30 stocks
retD_CRSP <- smallcapTS[ , sc30largest]
print(head(retD_CRSP, 3))
#>                    AVP         PBI        ITT         MUR         GHC
#> 1993-02-01 0.024122806 0.018461538 0.01043478 0.003472222 0.003215434
#> 1993-02-02 0.008565310 0.000000000 0.00000000 0.003460208 0.004273504
#> 1993-02-03 0.008492569 0.006042296 0.01893287 0.006896552 0.010638298
#>                   THC        AMD          FMC        BMS         DDS
#> 1993-02-01 0.01190476 0.02000000 -0.002564103 0.01932367 -0.01275510
#> 1993-02-02 0.00000000 0.05882353 -0.002570694 0.01421801  0.01033592
#> 1993-02-03 0.01176471 0.01234568 -0.015463918 0.00000000  0.04347826
#>                       R            J        RDC          DBD          BC
#> 1993-02-01 -0.004115226 -0.013698630 0.05454545  0.007692308 0.000000000
#> 1993-02-02  0.024793388  0.000000000 0.00000000 -0.020992367 0.007874016
#> 1993-02-03  0.016129032  0.009259259 0.03448276 -0.015594542 0.015625000
#>                     EAT          DLX         BIG         HSC       GNTX
#> 1993-02-01 -0.002717391 -0.002881844 -0.00729927  0.01572327 0.02450980
#> 1993-02-02  0.027247956 -0.002890173  0.03676471  0.00000000 0.03349282
#> 1993-02-03  0.000000000  0.005797102  0.06382979 -0.00619195 0.03703704
#>                     CY          KMT         MLHR         CBRL         AXE
#> 1993-02-01  0.04545455  0.000000000  0.029585799 -0.005649718 0.020512821
#> 1993-02-02  0.03260870 -0.017241379  0.005747126  0.022727273 0.005025126
#> 1993-02-03 -0.01052632  0.004385965 -0.011428571  0.011111111 0.005000000
#>                    CTB        IDTI          CBB MATX          GGG
#> 1993-02-01 0.011194030  0.01666667 -0.006802721    0 -0.005235602
#> 1993-02-02 0.003690037 -0.01639344  0.013698630    0  0.021052632
#> 1993-02-03 0.029411765  0.00000000  0.020270269    0  0.020618556
```

In the following part, we only show the time series of monthly returns of 10 CRSP® stocks in the last five years, but you can use this code to view the time series of any subset of stocks in any frequency and any time period.

```
# monthly return of top 30 stocks in last 5 years
ep <- endpoints(retD_CRSP, on= "months", k=1)
prod1 <- function(x){apply(x+1, 2, prod)}
retM_CRSP <- period.apply(retD_CRSP, INDEX = ep, FUN = prod1) - 1
retM_CRSP_5 <- tail(retM_CRSP, 60)

# time series plot of 10 stocks
tsPlotMP(retM_CRSP_5[, 1:10])
```

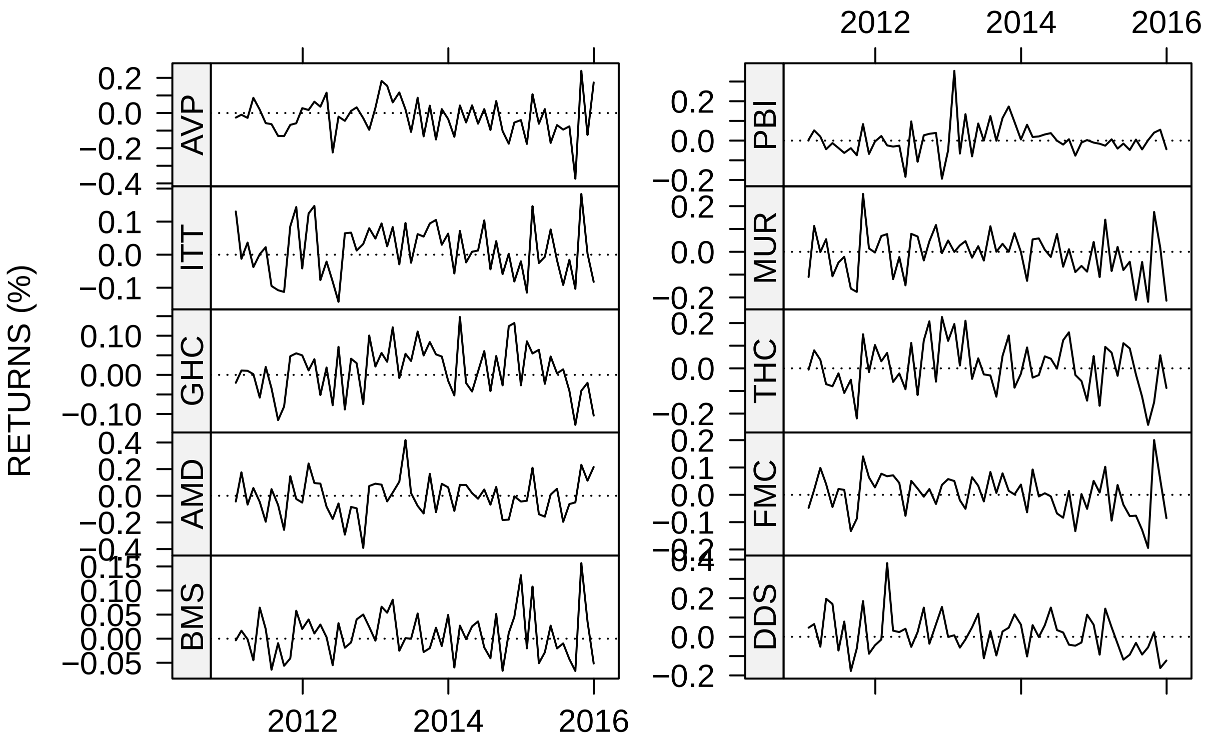


Fig 9.1

9.1 Backtesting with GMV, GMES, GMCSM Portfolios

In this example, we use daily return of all the CRSP® 30 stocks to generate a comparative backtesting among Global Minimum Variance (GMV), Global Minimum ES (GMES) and Global Minimum CSM (GMCSM) 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.

```
# Generate GMV, GMES and GMCSM portfolios
pspec_sc <- portfolio.spec(assets = sc30largest)
pspec_sc <- add.constraint(pspec_sc, type = "full_investment")
pspec_sc <- add.constraint(pspec_sc, type = "long_only")
```

```r
pspec_GMV <- add.objective(pspec_sc, type = "risk", name = "var")
pspec_GMES <- add.objective(pspec_sc, type = "risk", name = "ES")
pspec_GMCSM <- add.objective(pspec_sc, type = "risk", name = "CSM")

# Optimize Portfolio at Monthly Rebalancing and 500-Day Training
bt.GMV <- optimize.portfolio.rebalancing(retD_CRSP, pspec_GMV,
                                         optimize_method = "CVXR",
                                         rebalance_on = "months",
                                         training_period = 30,
                                         rolling_window = 500)
bt.ES <- optimize.portfolio.rebalancing(retD_CRSP, pspec_GMES,
                                         optimize_method = "CVXR",
                                         rebalance_on = "months",
                                         training_period = 30,
                                         rolling_window = 500)
bt.CSM <- optimize.portfolio.rebalancing(retD_CRSP, pspec_GMCSM,
                                         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.CSM <- extractWeights(bt.CSM)
wts.CSM <- wts.CSM[complete.cases(wts.CSM),]

# Compute cumulative returns of three portfolios
GMV <- Return.rebalancing(retM_CRSP, wts.GMV)
ES <- Return.rebalancing(retM_CRSP, wts.ES)
CSM <- Return.rebalancing(retM_CRSP, wts.CSM)

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

backtest.plot(ret.comb, colorSet = c("black", "darkblue", "darkgreen"),
              ltySet = c(3, 2, 1))
```

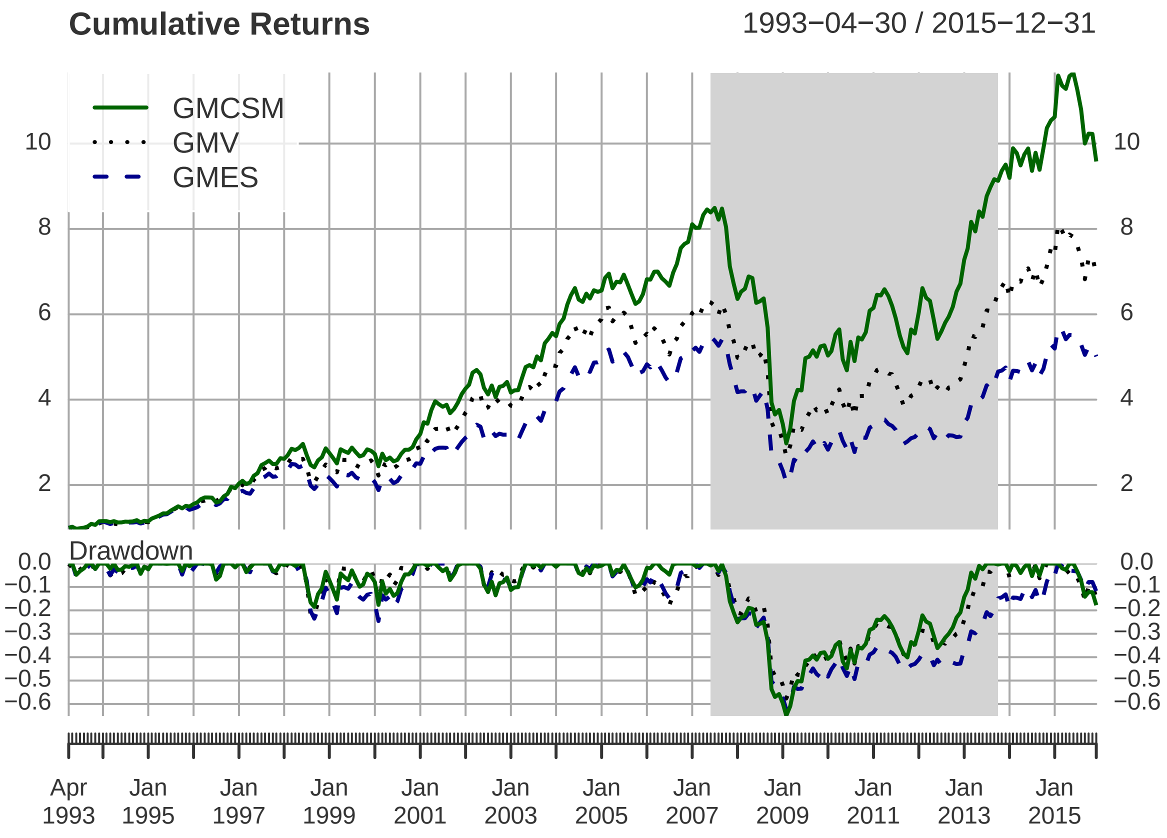


Fig 9.2

```
#> The run time for Figure 9.2 is 2.22 mins
```

9.2 Backtesting with SR, ESratio, CSMratio Portfolios

In this example, we use daily return of all the CRSP® 30 stocks to generate a comparative backtesting among Maximum Sharpe Ratio (SR), Maximum ES Ratio (ESratio) and Maximum CSM Ratio (CSMratio) 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.

```
# Generate GMV, GMES and GMCSM portfolios
pspec_sc_ratio <- add.objective(pspec_sc, type = "return", name = "mean")
pspec_Sr <- add.objective(pspec_sc_ratio, type = "risk", name = "var")
pspec_ESr <- add.objective(pspec_sc_ratio, type = "risk", name = "ES")
pspec_CSMr <- add.objective(pspec_sc_ratio, type = "risk", name = "CSM")

# Optimize Portfolio at Monthly Rebalancing and 500-Day Training
bt.Sr <- optimize.portfolio.rebalancing(retD_CRSP, pspec_Sr, maxSR = TRUE,
                                        optimize_method = "CVXR",
                                        rebalance_on = "months",
                                        training_period = 30,
                                        rolling_window = 500)
```

```r
bt.ESr <- optimize.portfolio.rebalancing(retD_CRSP, pspec_ESr,
                                         optimize_method = "CVXR",
                                         rebalance_on = "months",
                                         training_period = 30,
                                         rolling_window = 500)
bt.CSMr <- optimize.portfolio.rebalancing(retD_CRSP, pspec_CSMr,
                                         optimize_method = "CVXR",
                                         rebalance_on = "months",
                                         training_period = 30,
                                         rolling_window = 500)

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

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

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

# Compute cumulative returns of three portfolios
Sr <- Return.rebalancing(retM_CRSP, wts.Sr, rebalance_on = "months")
ESr <- Return.rebalancing(retM_CRSP, wts.ESr, rebalance_on = "months")
CSMr <- Return.rebalancing(retM_CRSP, wts.CSMr, rebalance_on = "months")

# Combine Sr, ESr and CSMr portfolio cumulative returns
ret.comb <- na.omit(merge(Sr, ESr, CSMr, all=F))
names(ret.comb) <- c("Sharpe ratio", "ES ratio", "CSM ratio")

backtest.plot(ret.comb, colorSet = c("black", "darkblue", "darkgreen"),
              ltySet = c(3, 2, 1))
```

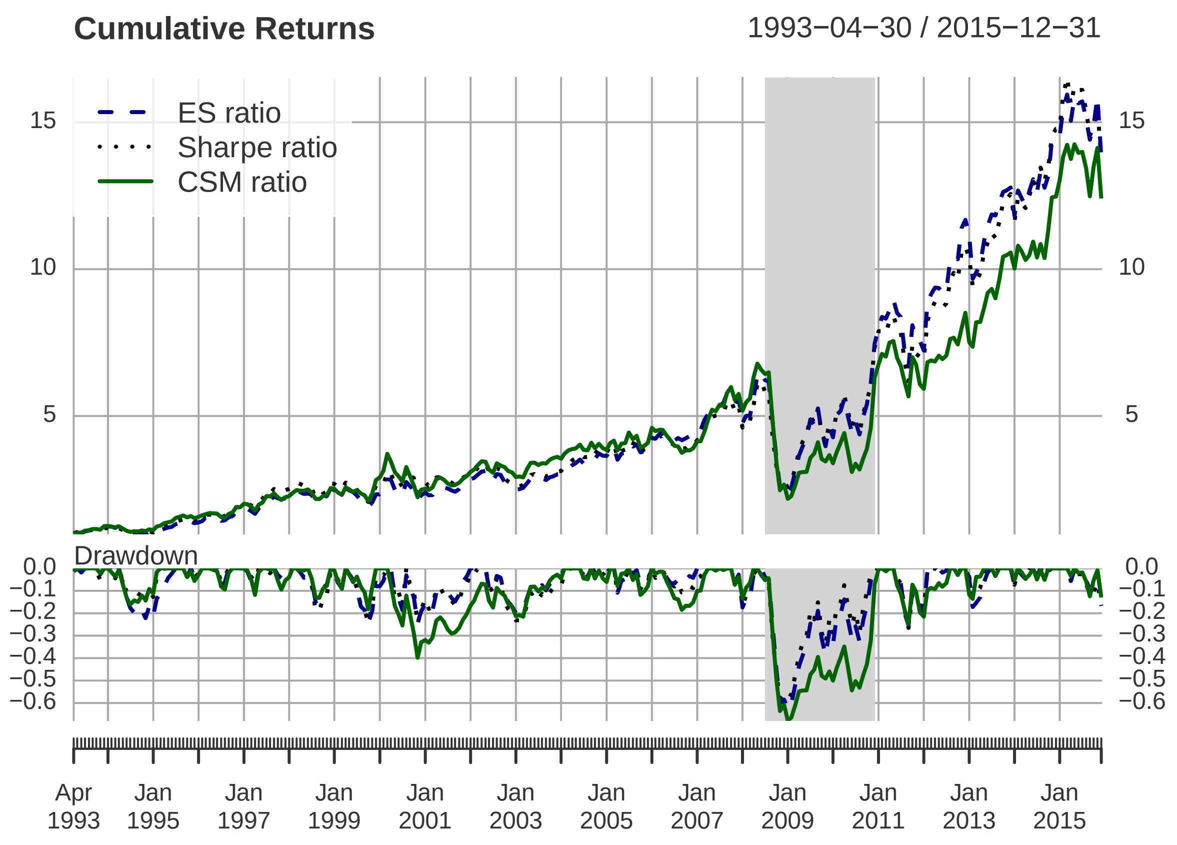


Fig 9.3

```
#> The run time for Figure 9.3 is 2.82 mins
```

9.3 Efficient Frontier

We generate efficient frontiers with mean-StdDev, mean-ES and mean-CSM portfolios using 30 stocks from CRSP® data set. For illustrative purposes, we use the last five years Considering that the data may show different properties over a long period of time, we only use the monthly return in the last 5 years to generate efficient frontiers, that is from 2011-01 to 2015-12 and defined in Section 2.3 as `retM_CRSP_5`. We can use `create.EfficientFrontier` to calculate the mean value and risk value for the frontier, then use `chart.EfficientFrontier` to draw the frontier.

9.3.1 Mean-StdDev Efficient Frontier

```
# mean-var efficient frontier
meanvar.ef <- create.EfficientFrontier(R = retM_CRSP_5, portfolio = pspec_sc,
                                        type = "mean-StdDev")
meanvar.ef
#> ****************************************************
#> PortfolioAnalytics Efficient Frontier
#> ****************************************************
#>
```

```
#> Call:
#> create.EfficientFrontier(R = retM_CRSP_5, portfolio = pspec_sc,
#>     type = "mean-StdDev")
#>
#> Efficient Frontier Points: 25
#>
#> **************************************************
#> PortfolioAnalytics Portfolio Specification
#> **************************************************
#>
#> Call:
#> portfolio.spec(assets = sc30largest)
#>
#> Number of assets: 30
#> Asset Names
#>  [1] "AVP" "PBI" "ITT" "MUR" "GHC" "THC" "AMD" "FMC" "BMS" "DDS"
#> More than 10 assets, only printing the first 10
#>
#> Constraints
#> Enabled constraint types
#>      - full_investment
#>      - long_only
```

```
chart.EfficientFrontier(meanvar.ef, match.col = "StdDev", type = "l",
                        chart.assets = FALSE, main = "Mean-StdDev Efficient Frontier",
                        RAR.text = "Sharpe ratio", pch = 1)
```

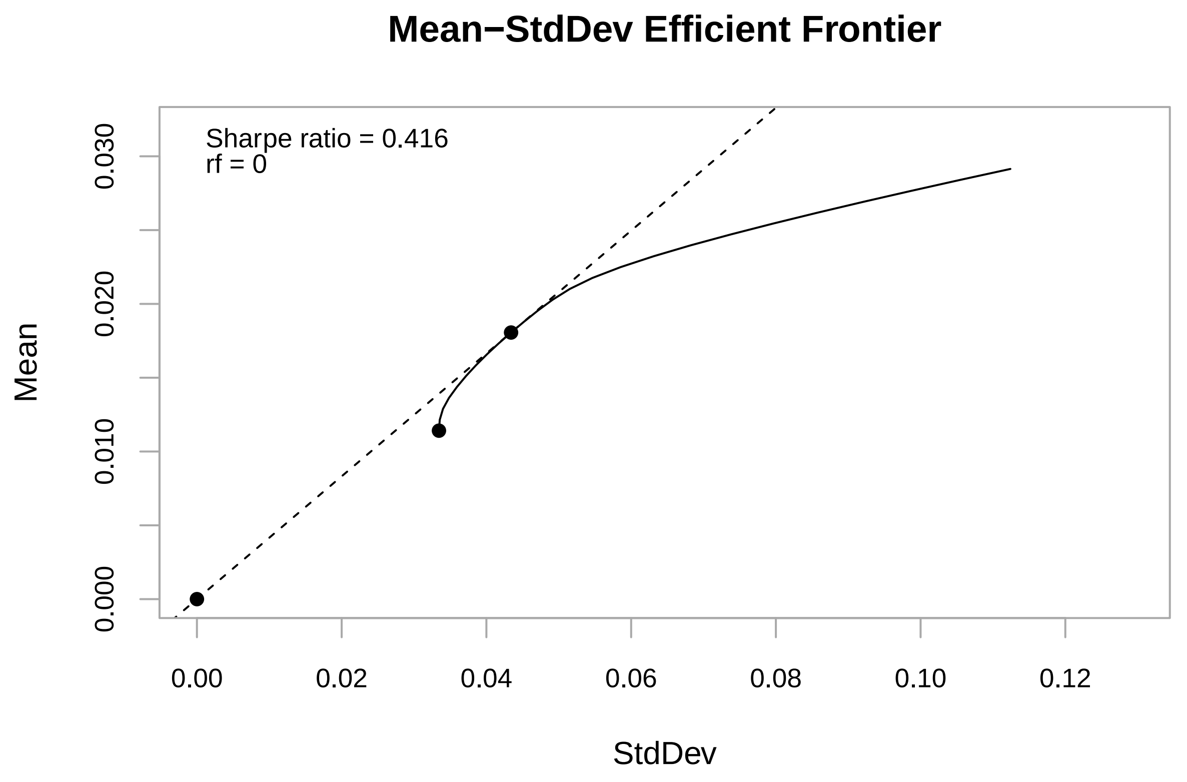


Fig 9.4

Here we compute the maximum Sharpe ratio on the efficient frontier using the following set of efficient frontier mean and standard deviation values:

```
meanvar.ef$frontier[, 1:2]
#>                 mean     StdDev
#> result.1  0.01141173 0.03344369
#> result.2  0.01215050 0.03355371
#> result.3  0.01288927 0.03399953
#> result.4  0.01362804 0.03482453
#> result.5  0.01436681 0.03591935
#> result.6  0.01510558 0.03717268
#> result.7  0.01584435 0.03856204
#> result.8  0.01658312 0.04007328
#> result.9  0.01732189 0.04169314
#> result.10 0.01806066 0.04340948
#> result.11 0.01879943 0.04521131
#> result.12 0.01953820 0.04709002
#> result.13 0.02027697 0.04913130
#> result.14 0.02101574 0.05153252
#> result.15 0.02175451 0.05463076
#> result.16 0.02249328 0.05854817
#> result.17 0.02323205 0.06313372
#> result.18 0.02397082 0.06825285
#> result.19 0.02470959 0.07379462
#> result.20 0.02544836 0.07967088
#> result.21 0.02618713 0.08581858
#> result.22 0.02692591 0.09219807
#> result.23 0.02766468 0.09876525
#> result.24 0.02840345 0.10549834
#> result.25 0.02914222 0.11240968
```

```
sr = meanvar.ef$frontier[, 1]/meanvar.ef$frontier[, 2]
cat("maximum Sharpe ratio:", max(sr))
#> maximum Sharpe ratio: 0.4160534
```

```
cat("mean of the maximum SR portfolio:", meanvar.ef$frontier[, 1][sr == max(sr)])
#> mean of the maximum SR portfolio: 0.01806066
```

```
cat("StdDev of the maximum SR portfolio:", meanvar.ef$frontier[, 2][sr == max(sr)])
#> StdDev of the maximum SR portfolio: 0.04340948
```

Note that we have introduced a method of finding the maximum Sharpe ratio portfolio in Section 8.1, which may have the values a little different from the estimated maximum Sharpe ratio calculated using the discrete efficient frontier values.

We now use that method to compute the maximum Sharpe ratio portfolio directly:

```
# Mean-StdDev Efficient Frontier
pspec_MV <- add.objective(pspec_sc, type = "risk", name = "var")
pspec_MV <- add.objective(portfolio = pspec_MV, type = "return", name = "mean")
opt_MV <- optimize.portfolio(retM_CRSP_5, pspec_MV, optimize_method = "CVXR",
                             maxSR = TRUE, trace = TRUE)
opt_MV
#> ***********************************
#> PortfolioAnalytics Optimization
#> ***********************************
#>
#> Call:
#> optimize.portfolio(R = retM_CRSP_5, portfolio = pspec_MV, optimize_method = "CVXR",
#>     trace = TRUE, maxSR = TRUE)
#>
#> Optimal Weights:
#>    AVP    PBI    ITT    MUR    GHC    THC    AMD    FMC    BMS    DDS      R
#> 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0755 0.0203 0.0000
#>      J    RDC    DBD     BC    EAT    DLX    BIG    HSC   GNTX     CY    KMT
#> 0.0000 0.0000 0.0000 0.0000 0.3166 0.0718 0.0486 0.0000 0.0000 0.0000 0.0000
#>   MLHR   CBRL    AXE    CTB   IDTI    CBB   MATX    GGG
#> 0.0000 0.2753 0.0000 0.0136 0.1218 0.0130 0.0436 0.0000
#>
#> Objective Measures:
#>    mean
#> 0.01819
#>
#>
#>  StdDev
#> 0.04372
```

```
#>
#>
#> Sharpe Ratio
#>       0.4161
```

The above direct computation of the maximum Sharpe ratio portolio's Sharpe ratio = 0.4161, agrees to three significant digits with the value obtained by searching the discrete set of efficient frontier mean and standard deviation values.

```r
chart.EfficientFrontier(opt_MV, match.col = "StdDev", chart.assets = FALSE,
                        main = "Mean-StdDev Efficient Frontier",
                        RAR.text = "Sharpe Ratio", pch = 1, xlim = c(0, 0.1))
```

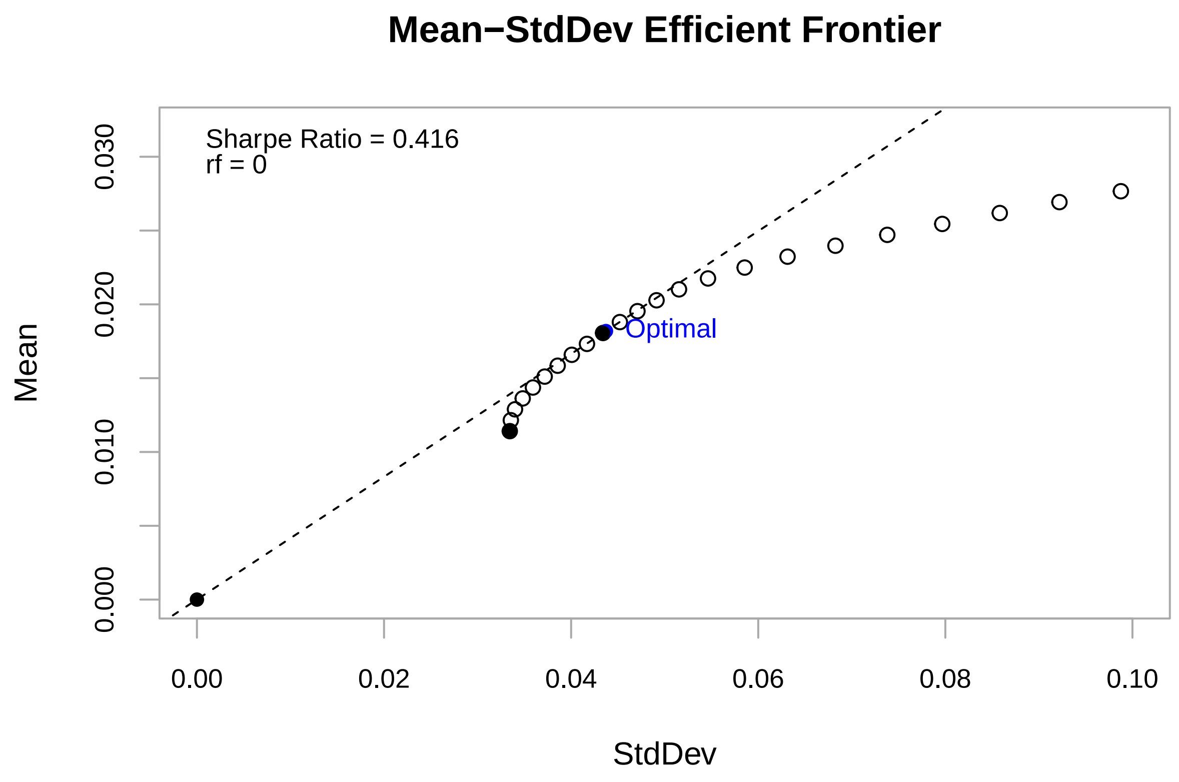


Fig 9.5

The theoretical maximum Sharpe ratio portfolio is very close to the result generated by the efficient frontier, and the Sharpe ratio value is same to 3 significant digits.

With different constraint types, we can create mean-StdDev efficient frontiers for multiple portfolios and overlay the plots.

```r
pspec_sc_init <- portfolio.spec(assets = sc30largest)
pspec_sc_init <- add.constraint(pspec_sc_init, type = "full_investment")
```

```r
# Portfolio with long-only constraints
pspec_sc_lo <- add.constraint(portfolio = pspec_sc_init, type = "long_only")

# Portfolio with long-only box constraints
pspec_sc_lobox <- add.constraint(portfolio = pspec_sc_init, type = "box",
                                 min = 0.02, max = 0.1)

# Portfolio with long-short box constraints
pspec_sc_lsbox <- add.constraint(portfolio = pspec_sc_init, type = "box",
                                 min = -0.1, max = 0.1)

# Combine the portfolios into a list
portf_list <- combine.portfolios(list(pspec_sc_lo, pspec_sc_lobox, pspec_sc_lsbox))

# Plot the efficient frontier overlay of the portfolios with varying constraints
legend_labels <- c("Long Only", "Long Only Box", "Long Short Box")
chart.EfficientFrontierOverlay(R = retM_CRSP_5, portfolio_list = portf_list,
                               type = "mean-StdDev", match.col = "StdDev",
                               legend.loc = "bottomright", chart.assets = FALSE,
                               legend.labels = legend_labels, cex.legend = 1,
                               labels.assets = FALSE, lwd = c(3,3,3),
                               col = c("black", "dark red", "dark green"),
                               main = "Overlay Mean-StdDev Efficient Frontiers",
                               xlim = c(0.03, 0.11), ylim = c(0.005, 0.035))
```

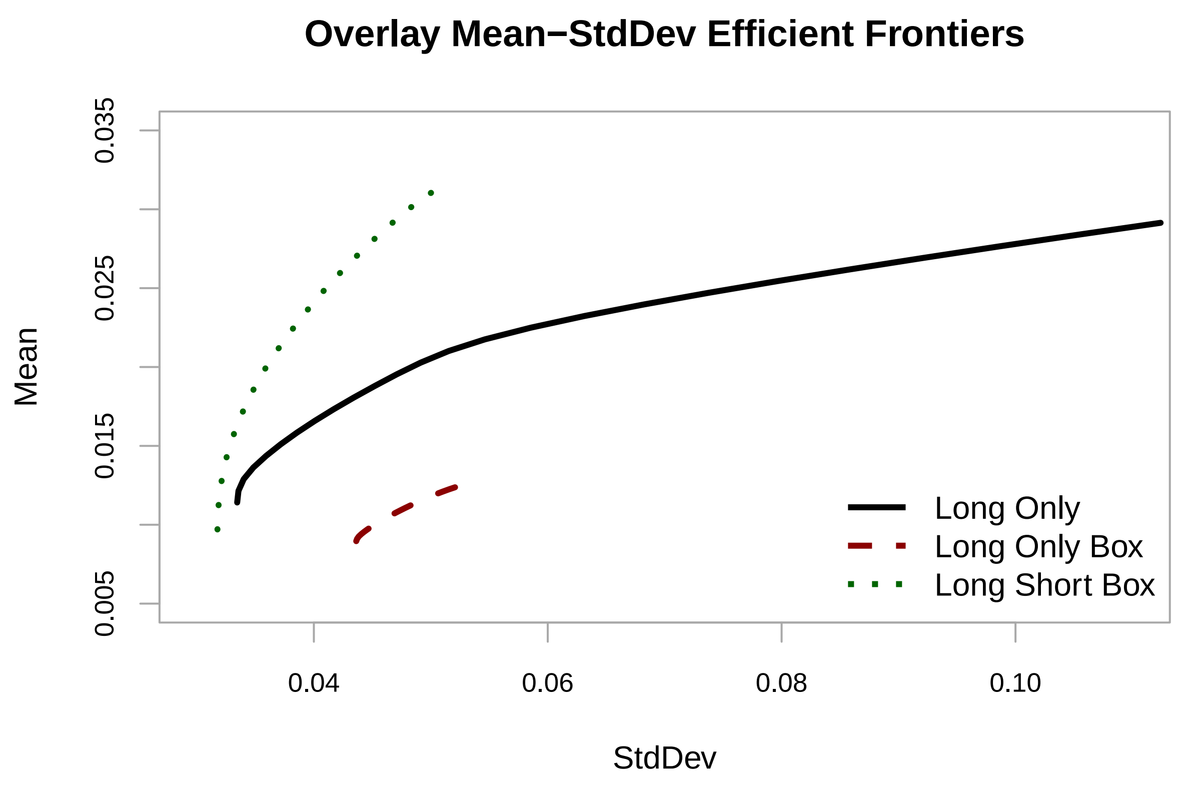


Fig 9.6

The plot clearly shows that the long-short box constrained portfolio has the best performance, though it also requires shorting which may not be possible for many real-world portfolios.

9.3.2 Mean-ES Efficient Frontier

Generate the mean-ES efficient frontier:

```r
# Mean-ES Efficient Frontier
meanetl.ef <- create.EfficientFrontier(R = retM_CRSP_5, portfolio = pspec_sc,
                                       type = "mean-ES")
chart.EfficientFrontier(meanetl.ef, match.col = "ES", type = "l",
                        chart.assets = FALSE, main = "Mean-ES Efficient Frontier",
                        RAR.text = "ES ratio", pch = 1)
```

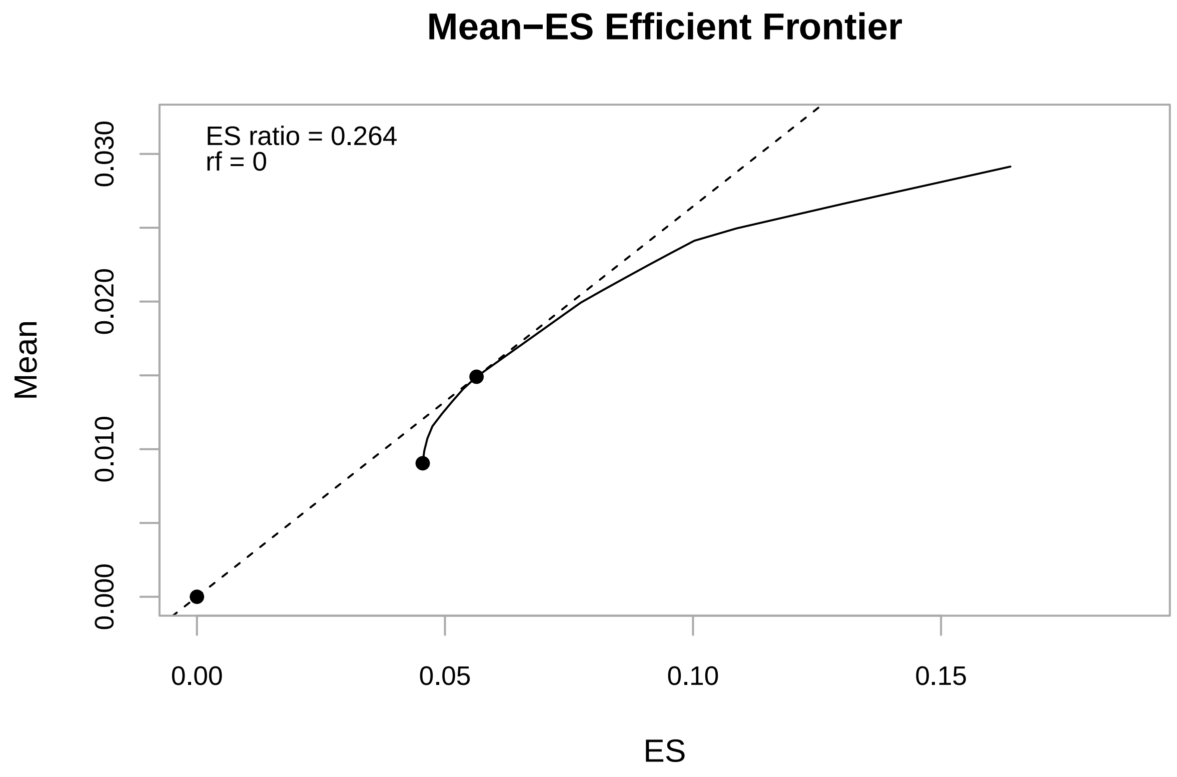


Fig 9.7

Generate multiple mean-ES efficient frontiers and overlay the plots.

```r
legend_labels <- c("Long Only ES (p=0.05)",
                   "Long Only Box ES (p=0.05)", "Long Short Box ES (p=0.05)")
chart.EfficientFrontierOverlay(R = retM_CRSP_5, portfolio_list = portf_list,
                               type = "mean-ES", match.col = "ES",
                               legend.loc = "bottomright", chart.assets = FALSE,
                               legend.labels = legend_labels, cex.legend = 1,
                               labels.assets = FALSE, lwd = c(3,3,3),
                               col = c("black", "dark red", "dark green"),
                               main = "Overlay Mean-ES Efficient Frontiers",
                               xlim = c(0.03, 0.17), ylim = c(0.005, 0.035))
```

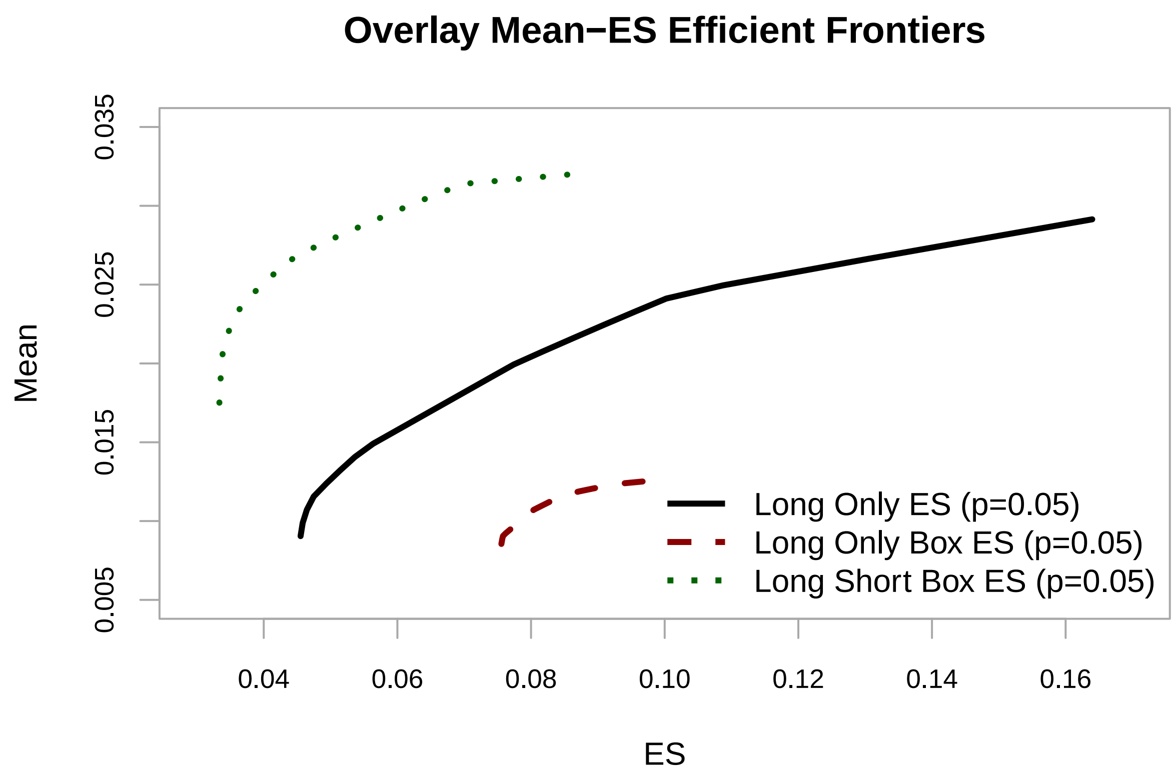


Fig 9.8

For each mean-ES efficient frontier, the left endpoint is the global minimum ES and its corresponding mean return, the right endpoint is the maximum mean return and and its corresponding ES, that is, the minimum ES when the maximum mean return is the target return.

With the same target return, the portfolio under the long-short box constraints can have a minimum ES value. It performs best, but shorting may not be possible for many real-world portfolios. The mean-ES efficient frontier of the long-only portfolio covers the widest range, that is because it can reach a larger return range than the long-box portfolio, but it is not as easy to reach the target return as the long-short portfolio, it needs to take more risks.

We could also notice that the ordering of the three constrained portfolios is the same of for the minVar efficient frontiers.

Instead of generating efficient frontiers with different constraint types, we can also generate mean-ES efficient frontiers with different tail probability γ.

```r
# Create long-only ES portfolios with different tail probabilities
ES_05 <- add.objective(portfolio = pspec_sc_lo, type = "risk", name = "ES",
                       arguments = list(p=0.05))

ES_10 <- add.objective(portfolio = pspec_sc_lo, type = "risk", name = "ES",
                       arguments = list(p=0.1))

ES_15 <- add.objective(portfolio = pspec_sc_lo, type = "risk", name = "ES",
                       arguments = list(p=0.15))

# Combine the portfolios into a list
portf_ES_list <- combine.portfolios(list(ES_05, ES_10, ES_15))

# Plot the efficient frontier overlay of the portfolios with varying tail probabilities
legend_ES_labels <- c("ES (p=0.05)", "ES (p=0.1)", "ES (p=0.15)")
chart.EfficientFrontierOverlay(R = retM_CRSP_5, portfolio_list = portf_ES_list,
                               type = "mean-ES", match.col = "ES",
                               legend.loc = "bottomright", chart.assets = FALSE,
                               legend.labels = legend_ES_labels, cex.legend = 1,
                               labels.assets = FALSE, lwd = c(3,3,3),
                               col = c("black", "dark red", "dark green"),
                               main = "Overlay Mean-ES Efficient Frontiers",
                               xlim = c(0.035, 0.165), ylim = c(0.005, 0.03))
```

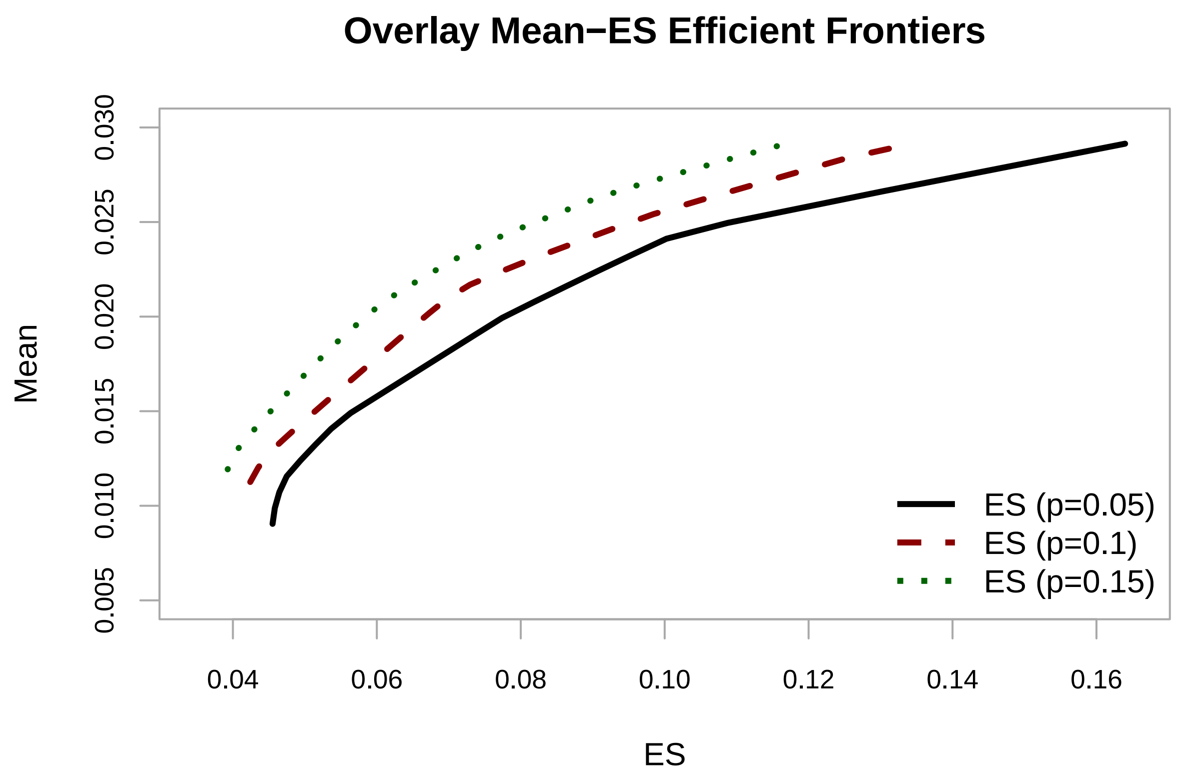


Fig 9.9

ES portfolio with a larger tail probability will have better performance, but this is only because larger tail probability gives a wider ES calculation range, so the expected ES value is smaller. Actually, these three lines cannot be compared because their x-lables are ES with different tail probabilities.

This plot mainly shows that the same portfolio will have different ES values under different tail probabilities. For example, the portfolio that reaches the maximum return, which is the right endpoint of each mean-ES efficient frontier. They are the same portfolio, but the ES values are different. It informs us the importance of tail probability for calculating ES. Different tail probabilities only focus on risks in different degrees, rather than changes in risk levels.

9.3.3 Mean-CSM Efficient Frontier

```r
# Mean-CSM Efficient Frontier
meancsm.ef <- create.EfficientFrontier(R = retM_CRSP_5, portfolio = pspec_sc,
                                       type = "mean-CSM")
chart.EfficientFrontier(meancsm.ef, match.col = "CSM", type = "l",
                        chart.assets = FALSE, main = "Mean-CSM Efficient Frontier",
                        RAR.text = "CSM ratio", pch = 1)
```

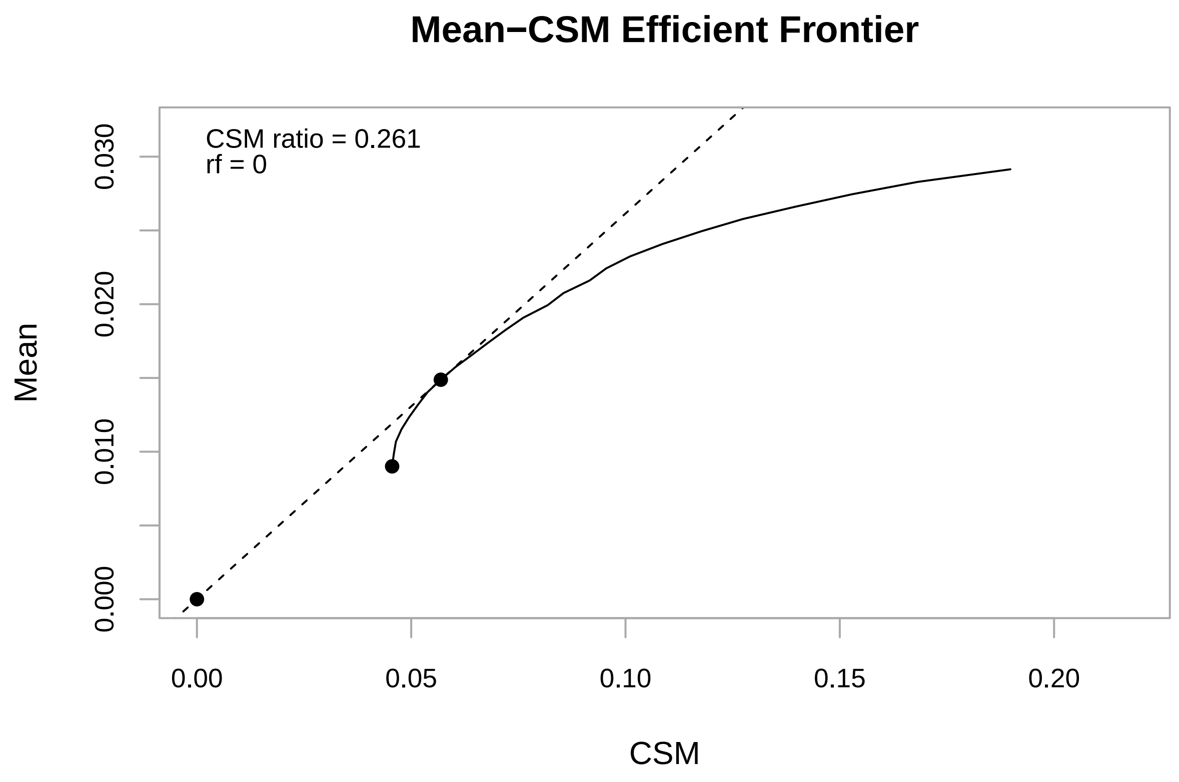


Fig 9.10

Mean-CSM efficient frontier is more like a piecewise function rather than a smooth curve.

9.4 Comparison among minStd, minES and minCSM Portfolios

To facilitate comparison of performance of various portfolios, we provide some new functions:

1. `extract_risk`: Calculate StdDev, ES and CSM value based on weights and historical returns.
2. `chart.EfficientFrontierCompare`: Efficient Frontier Comparison plot among different minRisk portfolios.
3. `plotFrontiers`: Allow users to visualize all the known frontiers and compare them easily.

9.4.1 Comparison of Portfolio Risk

In order to calculate the value of StdDev, ES and CSM of a portfolio, and thus help compare the performance of the portfolio, we provide a new function `extract_risk(R =, w =)`.

Asset returns `R` and the portfolio weights `w` should be specified when using the function, `ES_alpha` and `CSM_alpha` are alternative arguments and the default values are 0.05. Besides, covariance matrix is also required when calculating StdDev, so we provide an optional argument `moment_setting` to allow user to customize the moment values.

```r
# minStd Portfolio
minstd_port <- add.objective(pspec_sc, type = "risk", name = "StdDev")
minstd_opt <- optimize.portfolio(retM_CRSP_5, minstd_port, optimize_method = "CVXR")
minstd_w <- minstd_opt$weight

# extract risk example 1: risk values with default alpha = 0.05
extract_risk(retM_CRSP_5, minstd_w)
#> $mean
#> [1] 0.01141173
#>
#> $StdDev
#>            [,1]
#> [1,] 0.03344369
#>
#> $ES
#> [1] 0.05088981
#>
#> $CSM
#> [1] 0.05615236
```

```r
# extract risk example 2: risk values with specific alpha
extract_risk(retM_CRSP_5, minstd_w, ES_alpha = 0.1, CSM_alpha = 0.1)
#> $mean
#> [1] 0.01141173
#>
#> $StdDev
#>            [,1]
#> [1,] 0.03344369
#>
#> $ES
#> [1] 0.04633688
#>
#> $CSM
#> [1] 0.05615236
```

```
# extract risk example 3: risk values with customized momentFUN
minstd_opt_custM <- optimize.portfolio(retM_CRSP_5, minstd_port, optimize_method = "CVXR",
                                       momentFUN = 'custom.covRob.Mcd')
extract_risk(retM_CRSP_5, minstd_w, moment_setting = minstd_opt_custM$moment_values)
#> $mean
#>              [,1]
#> [1,] 0.008111925
#>
#> $StdDev
#>             [,1]
#> [1,] 0.03368338
#>
#> $ES
#> [1] 0.05088981
#>
#> $CSM
#> [1] 0.05615236
```

9.4.2 Comparison among Frontiers of Different Minimum Risk Portfolios

For comparing a specific risk type in minStd, minES and minCSM portfolios and generating multiple efficient frontiers, we provide a new function

`chart.EfficientFrontierCompare(R =, portfolio =, risk_type =, match.col =, guideline =)`

where `risk_type` is the risk to be compared, and `match.col` is the vector of the portfolios that participate in the comparison. If there are only two frontiers in the comparison, the default of the argument `guideline` is `TRUE`.

```
# EFCompare example 1: Compare StdDev of minStd and minES portfolios with guideline
chart.EfficientFrontierCompare(R = retM_CRSP_5, portfolio = pspec_sc, risk_type = "StdDev",
                               match.col = c("StdDev", "ES"), lwd = c(2, 2))
```

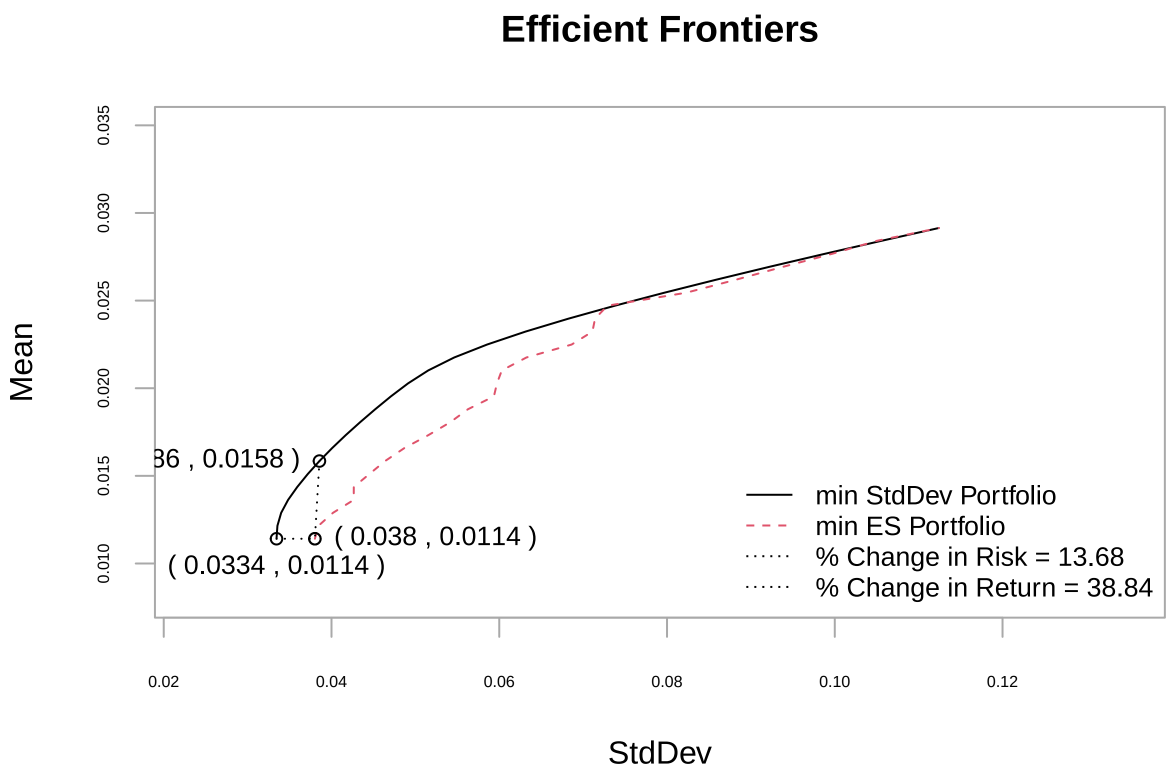


Fig 9.11

```
# EFCompare example 2: Compare ES of minStd, minES and minCSM portfolios without guideline
chart.EfficientFrontierCompare(R = retM_CRSP_5, portfolio = pspec_sc, risk_type = "ES",
                               match.col = c("StdDev", "ES", "CSM"), guideline = FALSE,
                               col = c(1,2,4), lty = c(1, 2, 4), lwd = c(2, 2, 2))
```

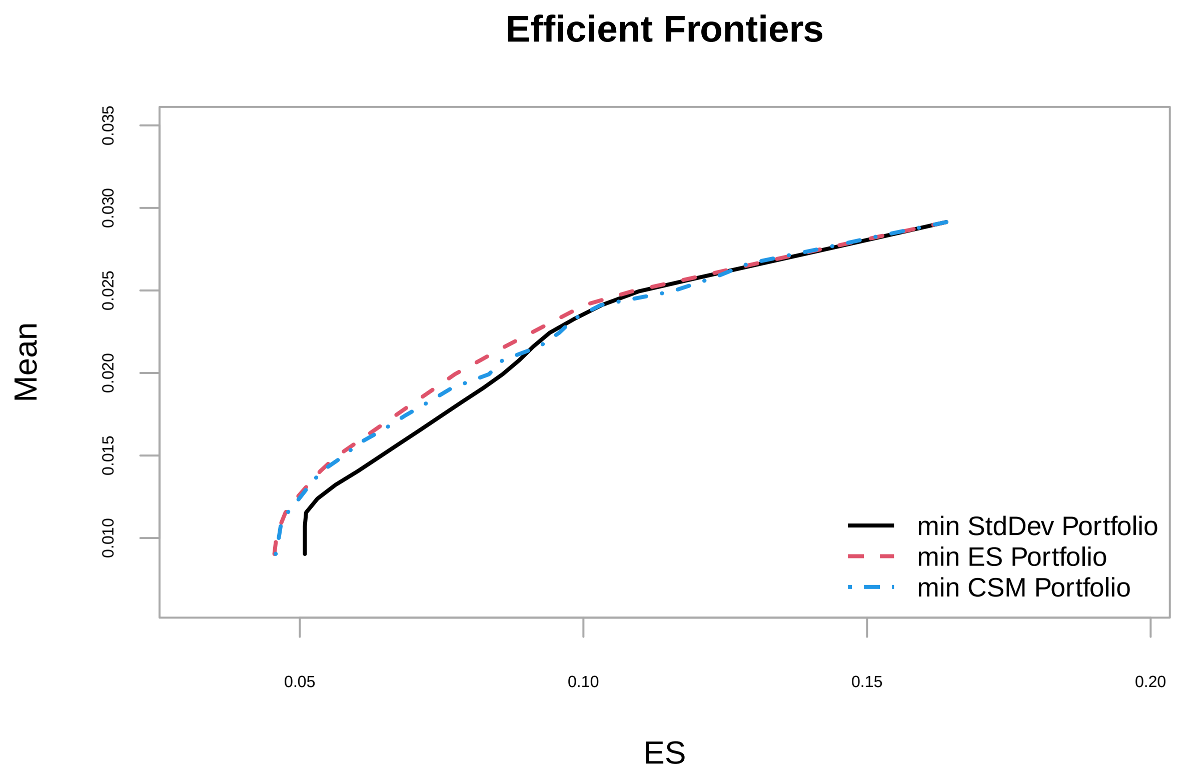


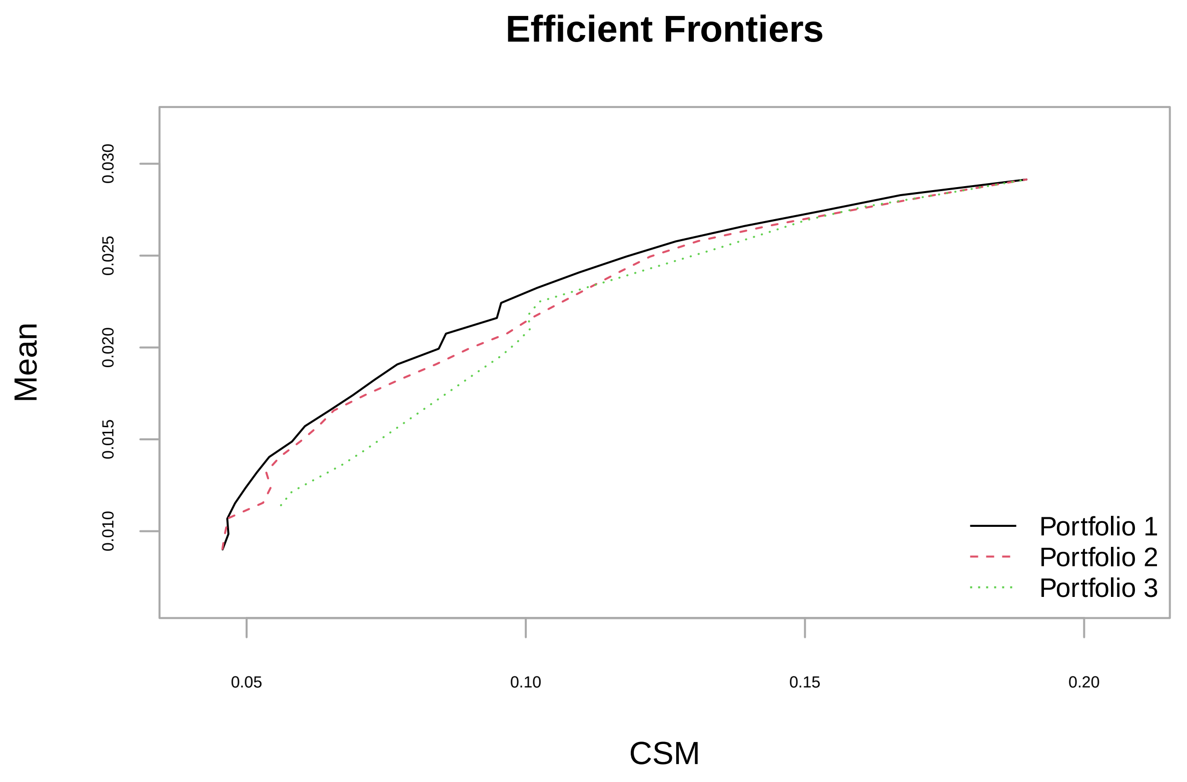
Fig 9.12

9.4.3 A Broader Comparison of Frontiers

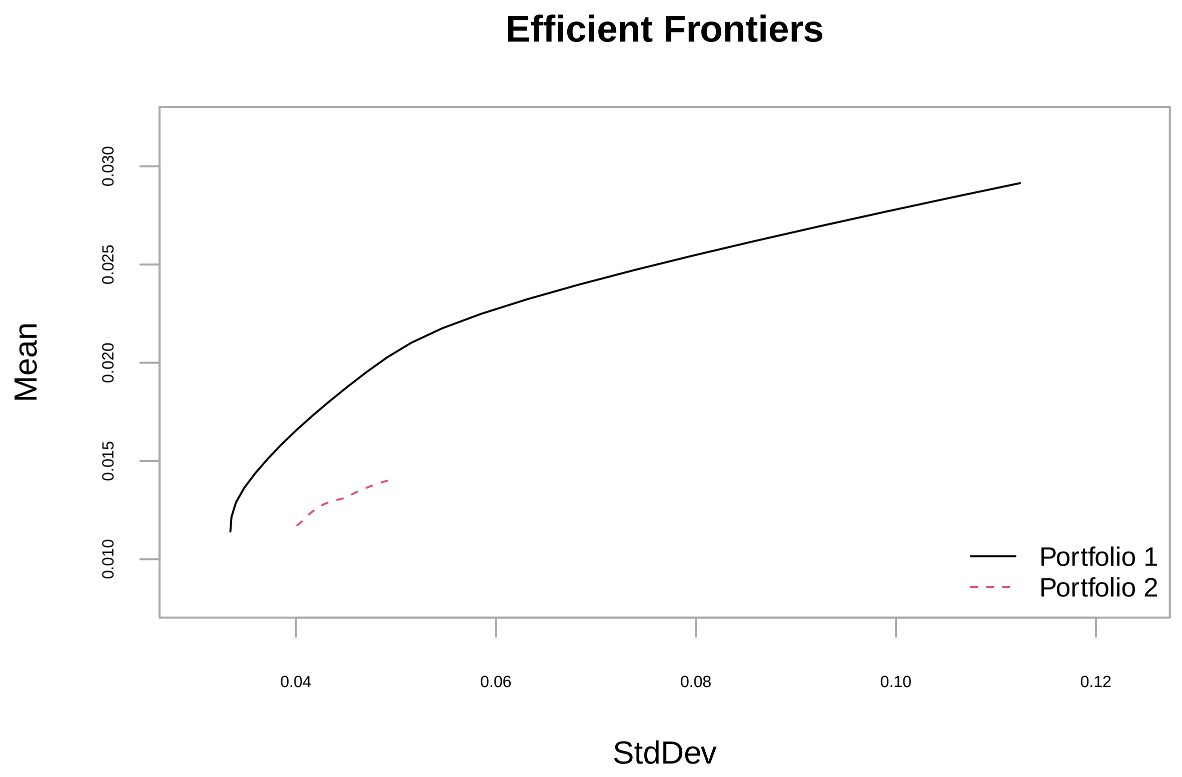
Then we may think about making more comparisons. For example, different min-Risk portfolios, different alpha value when comparing ES, different moment settings when calculating StdDev and so on.

Considering that PortfolioAnalytics already provides functions to generate frontiers, which are `meanvar.efficient.frontier`, `meanetl.efficient.frontier` and `meancsm.efficient.frontier`, we create a new function `plotFrontiers` to visualize all the frontier values for broader comparison.

```r
# plotFrontiers example 1: Compare mean-CSM frontiers for minvar, minES, minCSM portfolio
ef1 = meancsm.efficient.frontier(pspec_sc, retM_CRSP_5, optimize_method = 'CVXR')
ef2 = meanetl.efficient.frontier(pspec_sc, retM_CRSP_5, optimize_method = 'CVXR')
ef3 = meanvar.efficient.frontier(pspec_sc, retM_CRSP_5, optimize_method = 'CVXR')
plotFrontiers(R=retM_CRSP_5, frontiers=list(ef1, ef2, ef3), risk='CSM')
```



```r
# plotFrontiers example 2: Compare mean-var frontiers with different moment settings
ef4 = meanvar.efficient.frontier(pspec_sc, retM_CRSP_5, optimize_method = 'CVXR')
ef5 = meanvar.efficient.frontier(pspec_sc, retM_CRSP_5, optimize_method = 'CVXR',
                                 momentFUN = 'custom.covRob.TSGS')
plotFrontiers(R=retM_CRSP_5, frontiers=list(ef4, ef5), risk='StdDev')
```



Note that `plotFrontiers` only plots the values in the frontier object, and it is up to the users to control the accuracy of the values in the frontier object and interpret the economic meaning of the comparison.

Acknowledgements

This vignette aims to show the extended functionality of PortfolioAnalytics. CVXR solvers and CSM objective in PortfolioAnalytics were implemented by me, Xinran Zhao, when I was a student in the University of Washington Department of Applied Mathematics Computational Finance and Risk Management M.S. degree program (MS-CFRM), and was supported by Google Summer of Code (GSoC 2022) in the summer of 2022 with mentors Doug Martin and Steve Murray.

In the following year, I continuously generated more functions and improved this vignette, not limited to the implementation of CVXR and CSM, but more focused on their applications, such as back-testing analysis and efficient frontiers analysis. This process is very long, and I am grateful to everyone who accompanies and inspires me.

Doug Martin provided a lot of guidance on the derivation, application, and subsequent research directions of CSM. He also provided feedback on the style and content of this vignette. I am very grateful for learning the habit of research in this project.

Steve Murray is an expert in optimization. I'm greatly benefited by his analyzing different CVXR solvers, mathematical derivation in second-order cone optimization and mean-CSM efficient frontier, and his assistance in R coding.

Yifu Kang, who is my classmate in MS-CFRM and was funded by another PortfolioAnalytics GSoC 2022 project, expanded PortfolioAnalytics in custom robust covariance matrix. We have collaborated extensively on improving existing functions and interacting with each other's functions.

Finally, this project could not have been completed without the help of Brian Peterson, who is an Author and the Maintainer of the PortfolioAnalytics package on CRAN.

Reference

Cornuejols, Gerard, Javier Pena, and Reha Tutuncu. 2018. "Optimization Methods in Finance Second Edition."

Krokhmal, Pavlo A. 2007. "Higher Moment Coherent Risk Measures."

Rockafellar, R Tyrrell, Stanislav Uryasev, et al. 2000. "Optimization of Conditional Value-at-Risk." *Journal of Risk* 2: 21–42.