# CVXR for PortfolioAnalytics

Xinran Zhao

2022/9/18

## 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 Expected Quadratic Shortfall(EQS) problem, which is not supported by other solvers in PortfolioAnalytics. Hence, CVXR is a great 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 functions of PortfolioAnalytics, users are recommended to read the Vignette *Introduction to PortfolioAnalytics* first.

# 2 Getting Started

## 2.1 Load Packages

Load the necessary packages.

```
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 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 | ✓ | ✓ | ✓ |

| Solver | LP | QP | SOCP |
|---|---|---|---|
| MOSEK | ✓ | ✓ | ✓ |

## 2.3 Data

The edhec data set from the PerformanceAnalytics package is used as example data for examples from Section 3 to Section 8. The edhec data contains monthly returns for 13 assets from 1997-01 to 2019-11. We use the edhec data of the last 5 years as the example data to mainly 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.

```
tsPlotMP(ret_edhec, layout = c(2, 7))
```

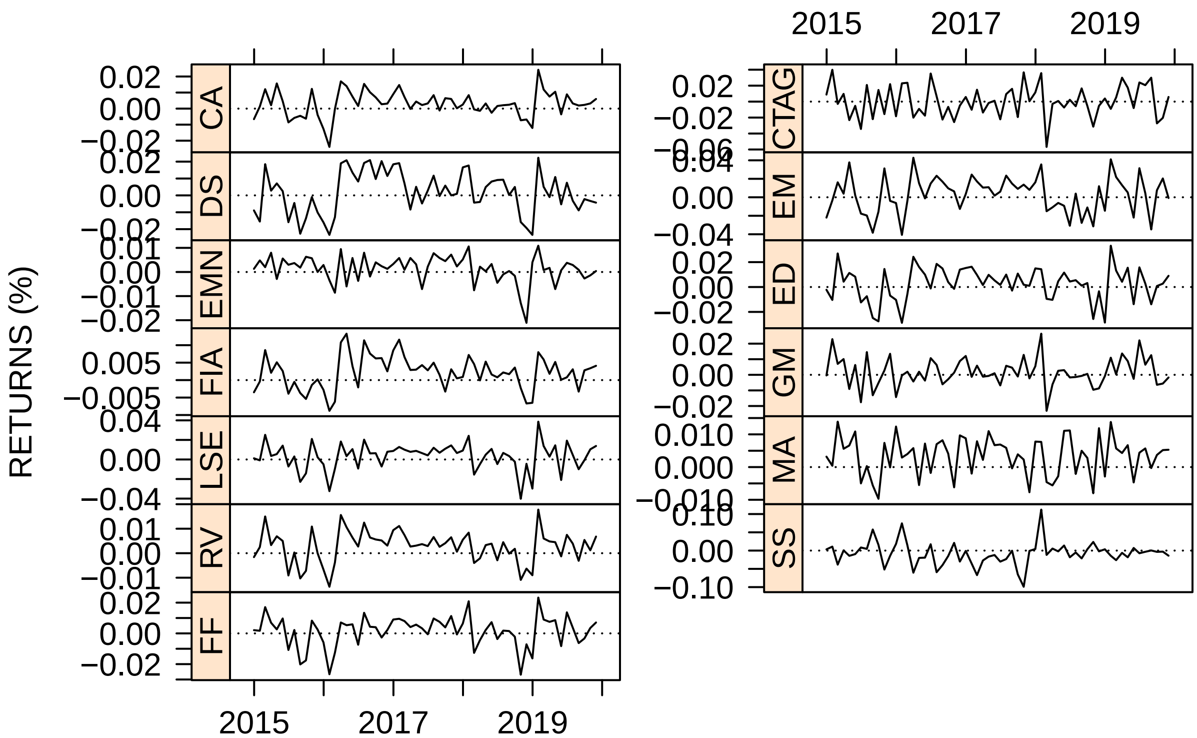


Fig 2.1

The CRSP data set is the daily 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 in Section 9. 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[, Return], 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")
#ret_CRSP <- smallcapD[, sc_30]
ret_CRSP = smallcapD[, apply(smallcapD, 2, function(y) all(!is.na(y)))]
print(head(ret_CRSP, 3))
```

```
##                     AIN         ALOG        ASNA          AXE         BGG
## 1993-01-04 -0.024000000 -0.043859649 -0.04827586  0.016393442 0.026881721
## 1993-01-05  0.000000000 -0.009174312  0.03623188  0.037634410 0.007853403
## 1993-01-06 -0.008196721 -0.009259259  0.02097902 -0.005181347 0.023376623
##                   BOBE          BRC        CASY         CLC          CW
## 1993-01-04 -0.05649717  0.004054054  0.014925373 -0.01298701  0.008097166
```

```
## 1993-01-05  0.02395210 -0.023648649 -0.007352941 -0.02631579  0.008032128
## 1993-01-06  0.00000000  0.031141868  0.000000000  0.00000000 -0.003984064
##                  ESND        FELE          FUL       HTLD         KMT
## 1993-01-04 0.08088236 -0.02083333  0.00617284 0.04347826 0.013157895
## 1993-01-05 0.02040816  0.00000000  0.00000000 0.00000000 0.004329004
## 1993-01-06 0.02666667  0.02127660 -0.01226994 0.02777778 0.004310345
##                   MATX        MENT          RBC         UVV
## 1993-01-04 -0.02020202 0.01538462  0.005847953 -0.02197802
## 1993-01-05  0.00000000 0.07575758  0.000000000  0.01782772
## 1993-01-06  0.01030928 0.04225352 -0.005813953 -0.01111111
```

```
fund_CRSP <- colnames(ret_CRSP)
```

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 check the time series performance of all stocks in any frequency and any time period.

```
# generate monthly return in last 5 years
ep <- endpoints(ret_CRSP, on= "months", k=1)
prod1 <- function(x){apply(x+1, 2, prod)}
retM_CRSP <- period.apply(ret_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])
```

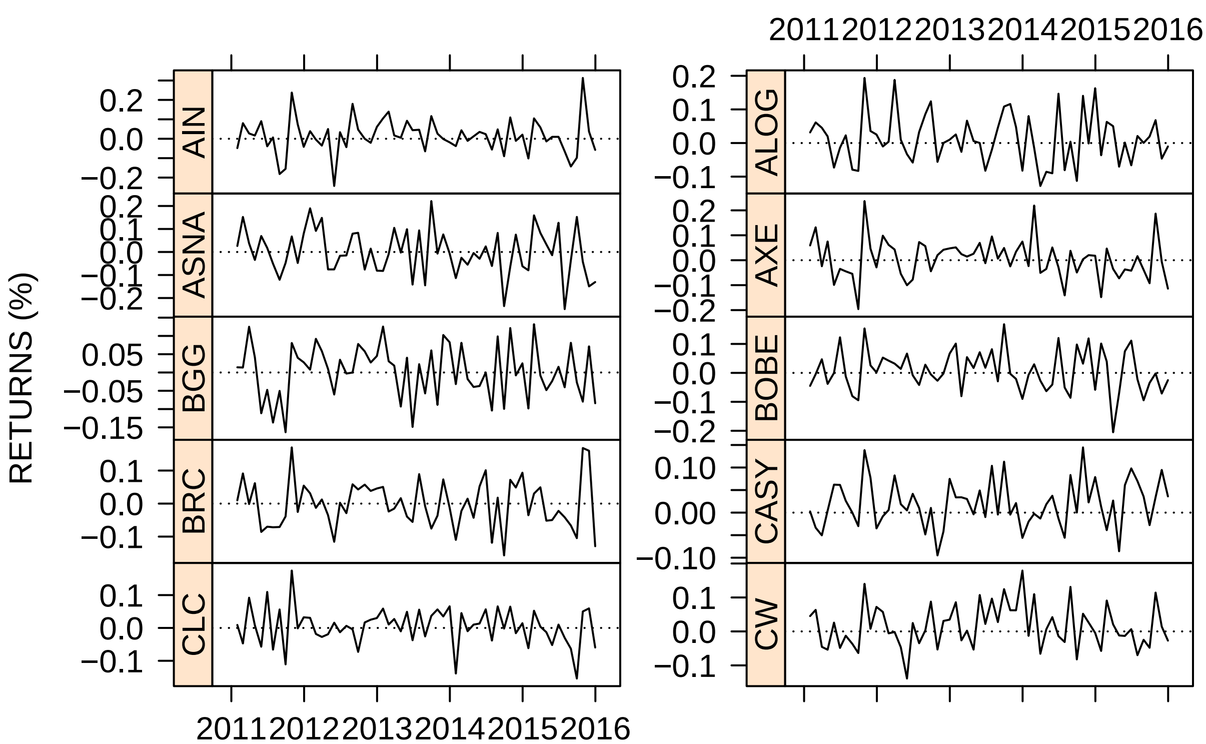

Fig 2.2

### 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. Then add constraints and a return objective.

```
# 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(0.6, 4), rep(0.2, 9)))
# 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 `OSQP`.

```r
# 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   SS   FF
## 0.40 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.20 0.20 0.02 0.02 0.02
##
## Objective Measures:
##     mean
## 0.002728
```

```r
opt_maxret$solver
```

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

```r
# 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 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.

## 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 portfolio.

```
# 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

```
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 can have short positions.

## 4.2 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(2:12),
                                       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
```

```
# backtesting
bt_mv <- optimize.portfolio.rebalancing(R=ret_edhec, 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", "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 model, where the quadratic utility function is $\mathrm{QU}(\boldsymbol{w}) = \mu_{\mathrm{p}} - \lambda\sigma_{\mathrm{p}}^2 = \boldsymbol{\mu'w} - \lambda\boldsymbol{w'}\Sigma\boldsymbol{w}$:

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

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

Where $\boldsymbol{\mu}$ 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 3, and as $\lambda \to \inf$, it becomes the 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 $\lambda$ 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 $q_\gamma$ is $\gamma$-quantile and $\gamma$ 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 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

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

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

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

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 objective is in the form of:

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

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).

### 6.1 Portfolio Object

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

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

### 6.2 Optimization

```
# 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
## -0.2185  0.0019 -0.2170 -0.0426  0.1553 -0.1615  0.5702  0.1538  0.4500  0.3607
##      RV      SS      FF
##  0.9412 -0.0059 -0.9876
##
## Objective Measures:
##         ES
## -0.0007227
```

```r
# 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
## -0.1422  0.0092 -0.2292 -0.0432  0.1632 -0.1456  0.5058  0.1074  0.4259  0.3558
##      RV      SS      FF
##  0.9560  0.0016 -0.9648
##
## Objective Measures:
##         ES
## -0.0007416
```

# 7 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:

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

where $\gamma$ 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 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).

## 7.1 Portfolio Object

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

```r
pspec_eqs <- portfolio.spec(assets=fund_edhec)
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=0.05))
```

## 7.2 Optimization

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

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = ret_edhec, portfolio = pspec_eqs, optimize_method = "CVXR")
##
## Optimal Weights:
##      CA    CTAG      DS      EM     EMN      ED     FIA      GM     LSE      MA
## -0.2120 -0.0018 -0.1960 -0.0367  0.1508 -0.1826  0.5591  0.1415  0.4410  0.3717
##      RV      SS      FF
##  0.9326 -0.0062 -0.9616
##
## Objective Measures:
##        EQS
## -0.0007107
```

# 8 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.

## 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 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.

```
# 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")
```

```r
## 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 0.2119 0.0000 0.4194 0.0000 0.0000 0.3638 0.0000
##     SS     FF
## 0.0049 0.0000
##
## Objective Measures:
##     mean
## 0.002397
##
##
##       ES
## 0.004555
##
##
## ES ratio
##   0.5262
```

## 8.3 Maximum EQS ratio Portfolios

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

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

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 $\gamma = 0.05$, and it can be specified by `arguments`.

```
# Create portfolio object
pspec_EQSratio <- portfolio.spec(assets=fund_edhec)
## Add constraints of maximizing return per unit EQS
```

```r
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=0.05))

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

```
## ***********************************
## PortfolioAnalytics Optimization
## ***********************************
##
## Call:
## optimize.portfolio(R = ret_edhec, portfolio = pspec_EQSratio,
##     optimize_method = "CVXR", EQSratio = TRUE)
##
## Optimal Weights:
##     CA   CTAG     DS     EM    EMN     ED    FIA     GM    LSE     MA     RV
## 0.0841 0.0000 0.0000 0.0000 0.1089 0.0000 0.4380 0.0000 0.0000 0.3022 0.0000
##     SS     FF
## 0.0667 0.0000
##
## Objective Measures:
##    mean
## 0.00185
##
##
##      EQS
## 0.004375
##
##
## EQS ratio
##    0.4229
```

# 9 Performance of Portfolios

CVXR solvers provide the Second-Order Cone Optimization (SOCopt) capability required to minimize EQS problem, and managing EQS is of great significance for building portfolios.

In this section, we use the CRSP data set to generate GMV, ES and EQS portfolios and show their performance by plotting cumulative returns and efficient frontiers. In this process, we would like to show the value of EQS in managing portfolios.

## 9.1 Backtesting with GMV, GMES, GMEQS portfolios

In this example, we use daily return of all the CRSP 30 small cap stocks to generate a comparative backtesting among Global Minimum Variance, Global Minimum ES and Global Minimum 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, GMES and GMEQS portfolios
pspec_sc <- portfolio.spec(assets=fund_CRSP)
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_GMES <- add.objective(pspec_sc, type="risk", name="ES")
pspec_GMEQS <- add.objective(pspec_sc, type="risk", name="EQS")

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

## Compute cumulative returns of three portfolios
GMV = Return.rebalancing(retM_CRSP, wts.GMV)
ES = Return.rebalancing(retM_CRSP, wts.ES)
EQS = Return.rebalancing(retM_CRSP, 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", "GMES", "GMEQS")

# Compute cumulative gross portfolios returns
R <- ret.comb
gross_cum <- TRUE
c.xts <- if ( gross_cum ) {
  cumprod(1+R)
} else {
  cumsum(R)
}
```

```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
```

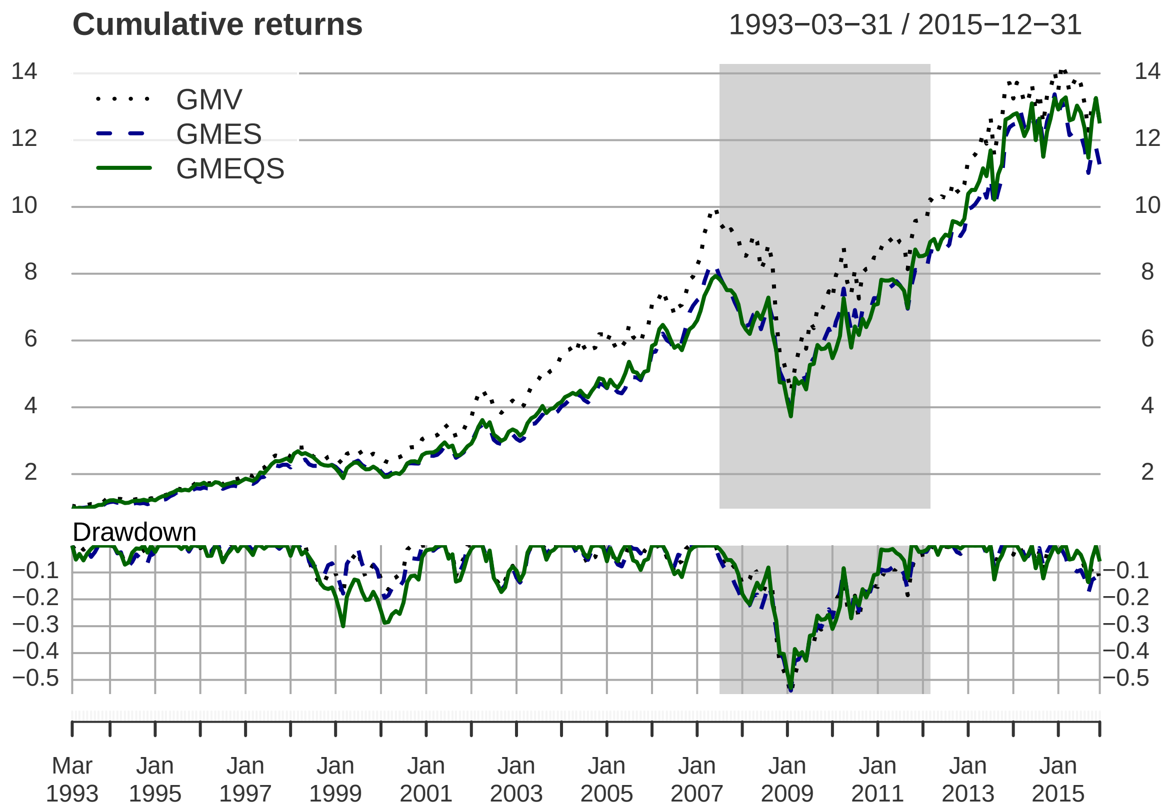


Fig 9.1

## 9.2 Backtesting with SR, ESratio, EQSratio portfolios

In this example, we use daily return of all the CRSP 30 small cap stocks to generate a comparative backtesting among Maximum Sharpe Ratio, Maximum ES Ratio and Maximum EQS Ratio 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 GMEQS 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_EQSr <- add.objective(pspec_sc_ratio, type="risk", name="EQS")

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

```
                                           rolling_window=500)
bt.EQSr <- optimize.portfolio.rebalancing(ret_CRSP, pspec_EQSr,
                                           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.EQSr = extractWeights(bt.EQSr)
wts.EQSr <- wts.EQSr[complete.cases(wts.EQSr),]

## Compute cumulative returns of three portfolios
Sr = Return.rebalancing(retM_CRSP, wts.Sr)
ESr = Return.rebalancing(retM_CRSP, wts.ESr)
EQSr = Return.rebalancing(retM_CRSP, wts.EQSr)

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

# Compute cumulative gross portfolios returns
R <- ret.comb
gross_cum <- TRUE
c.xts <- if ( gross_cum ) {
  cumprod(1+R)
} else {
  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", "Trough")])
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
```

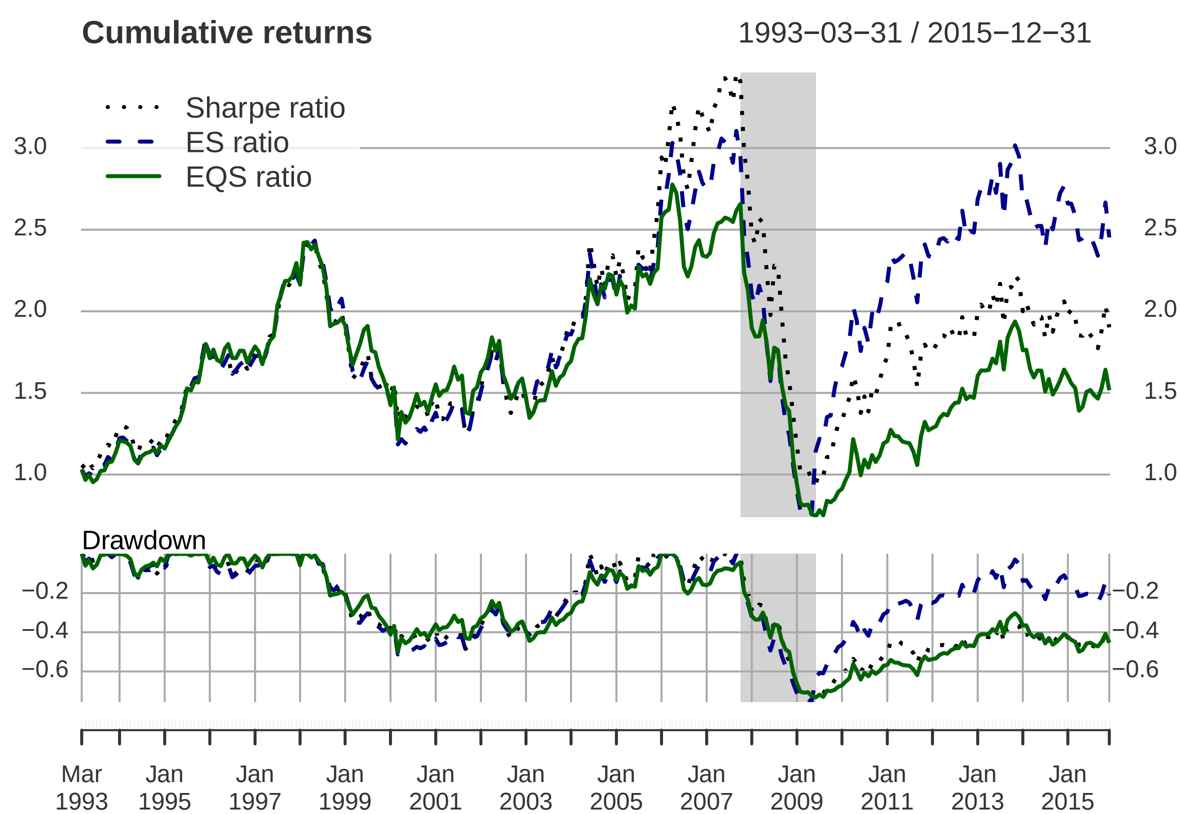


Fig 9.2

## 9.3 Efficient Frontier

We generate efficient frontiers with mean-StdDev, mean-ES and mean-EQS portfolios by using 30 small cap stocks from CRSP data set. 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")
```

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

```
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 = fund_CRSP)
##
## Number of assets: 19
## Asset Names
##  [1] "AIN"  "ALOG" "ASNA" "AXE"  "BGG"  "BOBE" "BRC"  "CASY" "CLC"  "CW"
## 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)
```

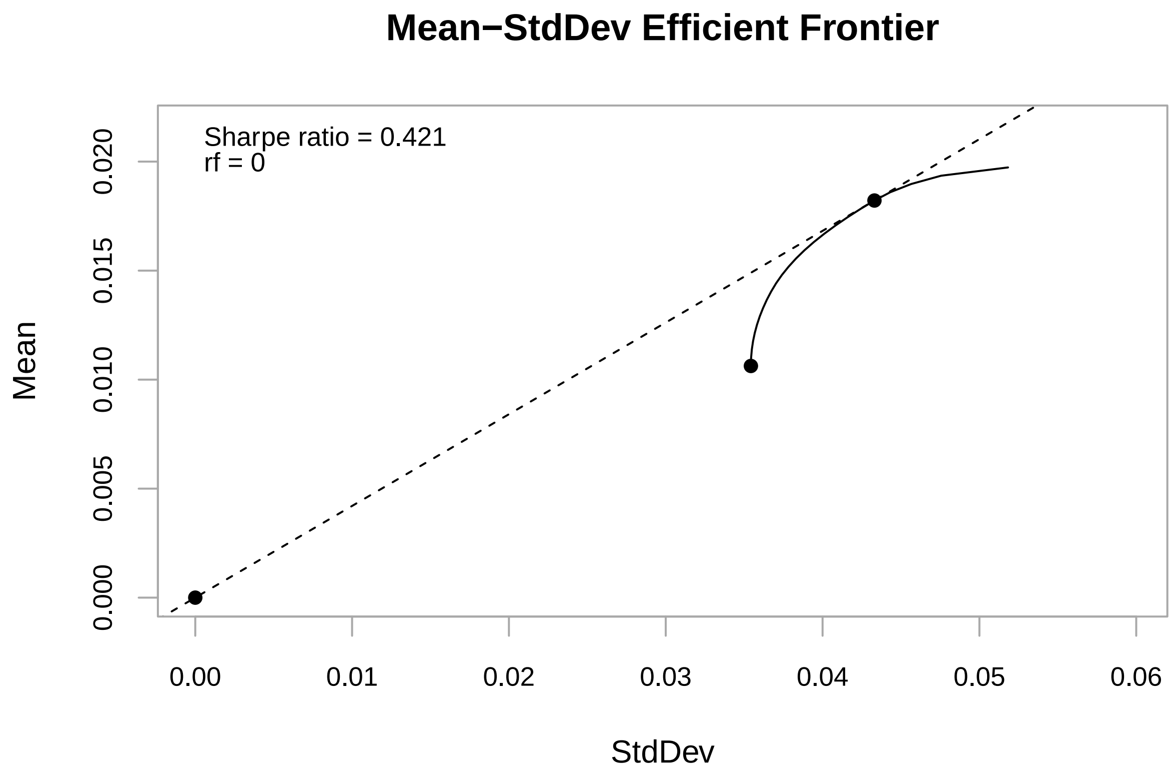


Fig 9.3

The Sharpe ratio could be calculated by the frontier value and the maximum Sharpe ratio could be found.

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

```
##                  mean     StdDev
## result.1  0.01062645 0.03542888
## result.2  0.01100585 0.03544404
## result.3  0.01138525 0.03548915
## result.4  0.01176465 0.03556670
## result.5  0.01214404 0.03567732
## result.6  0.01252344 0.03582070
## result.7  0.01290284 0.03599645
## result.8  0.01328224 0.03620409
## result.9  0.01366163 0.03644400
## result.10 0.01404103 0.03672163
## result.11 0.01442043 0.03703817
## result.12 0.01479983 0.03740682
## result.13 0.01517923 0.03783088
## result.14 0.01555862 0.03831024
## result.15 0.01593802 0.03885239
## result.16 0.01631742 0.03945621
## result.17 0.01669682 0.04011890
## result.18 0.01707621 0.04083760
```

```
## result.19 0.01745561 0.04160942
## result.20 0.01783501 0.04243168
## result.21 0.01821441 0.04330721
## result.22 0.01859381 0.04431183
## result.23 0.01897320 0.04565624
## result.24 0.01935260 0.04755942
## result.25 0.01973200 0.05182683
```

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

```
## maximum Sharpe ratio: 0.4205861
```

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

```
## mean of the maximum SR portfolio: 0.01821441
```

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

```
## StdDev of the maximum SR portfolio: 0.04330721
```

Note that we have introduced the method of finding the theoretical maximum Sharpe ratio portfolio in Section 8.1, which may be a little different from the estimated maximum Sharpe ratio calculated by the discrete efficient frontier value. It is because the function of efficient frontier uses the mean value of the maximum mean return portfolio and the minimum variance portfolio as boundary values, then divides the mean interval equally and calculates the corresponding StdDev value, and then gives discrete mean-StdDev points to fit the efficient frontier curve. The "maximum" Sharpe ratio found by the efficient frontier function is the maximum value calculated by limited number of discrete points. The default number of points is 25, and the specific number could be given by `n.portfolios = {number}`.

We can identify the maximum Sharpe ratio portfolio in blue point on the mean-StdDev efficient frontier.

```
# 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:
##    AIN   ALOG   ASNA    AXE    BGG   BOBE    BRC   CASY    CLC     CW   ESND
## 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.6683 0.0000 0.1498 0.0000
##   FELE    FUL   HTLD    KMT   MATX   MENT    RBC    UVV
```

```
## 0.0000 0.0000 0.0061 0.0000 0.1289 0.0469 0.0000 0.0000
##
## Objective Measures:
##    mean
## 0.01821
##
##
## StdDev
## 0.0433
##
##
## Sharpe Ratio
##       0.4206
```

```
chart.EfficientFrontier(opt_MV, match.col="StdDev", chart.assets = FALSE,
                        main="Mean-StdDev Efficient Frontier",
                        RAR.text="Sharpe Ratio", pch=1, xlim = c(0, 0.06))
```

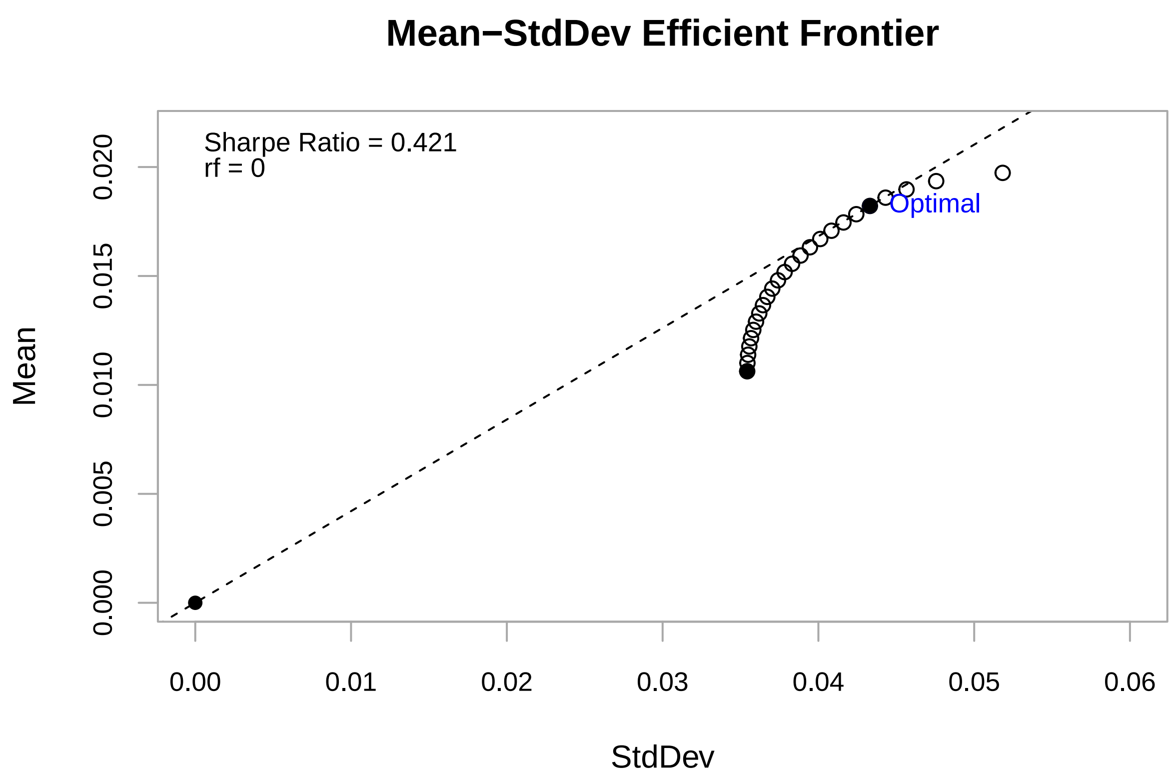


Fig 9.4

The theoretical maximum Sharpe ratio portfolio is very close to the result generated by the efficient frontier, and the Sharpe ratio value is almost the same but the mean and StdDev value are slightly different.

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=fund_CRSP)
pspec_sc_init <- add.constraint(pspec_sc_init, type="full_investment")

# 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="topleft", 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, 0.06), ylim = c(0.005, 0.025))
```

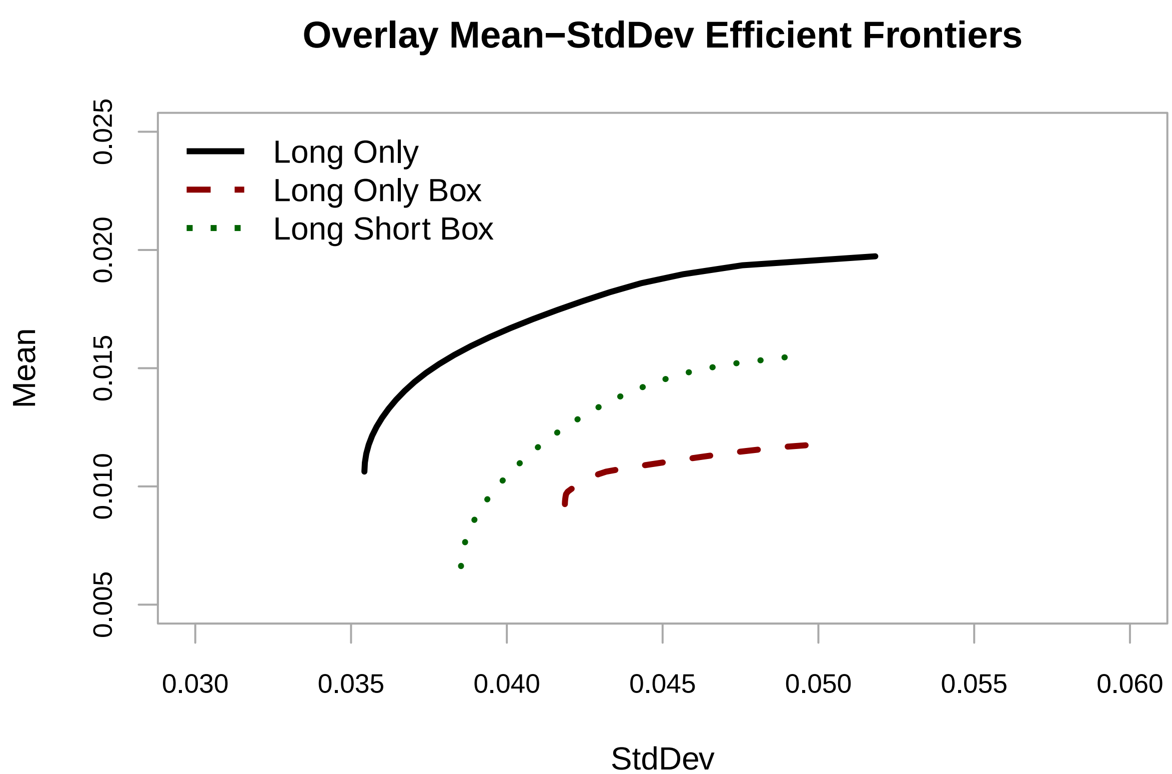

Fig 9.5

The plot clearly show that the portfolio under the long-short box constraints 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:

```
# 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)
```

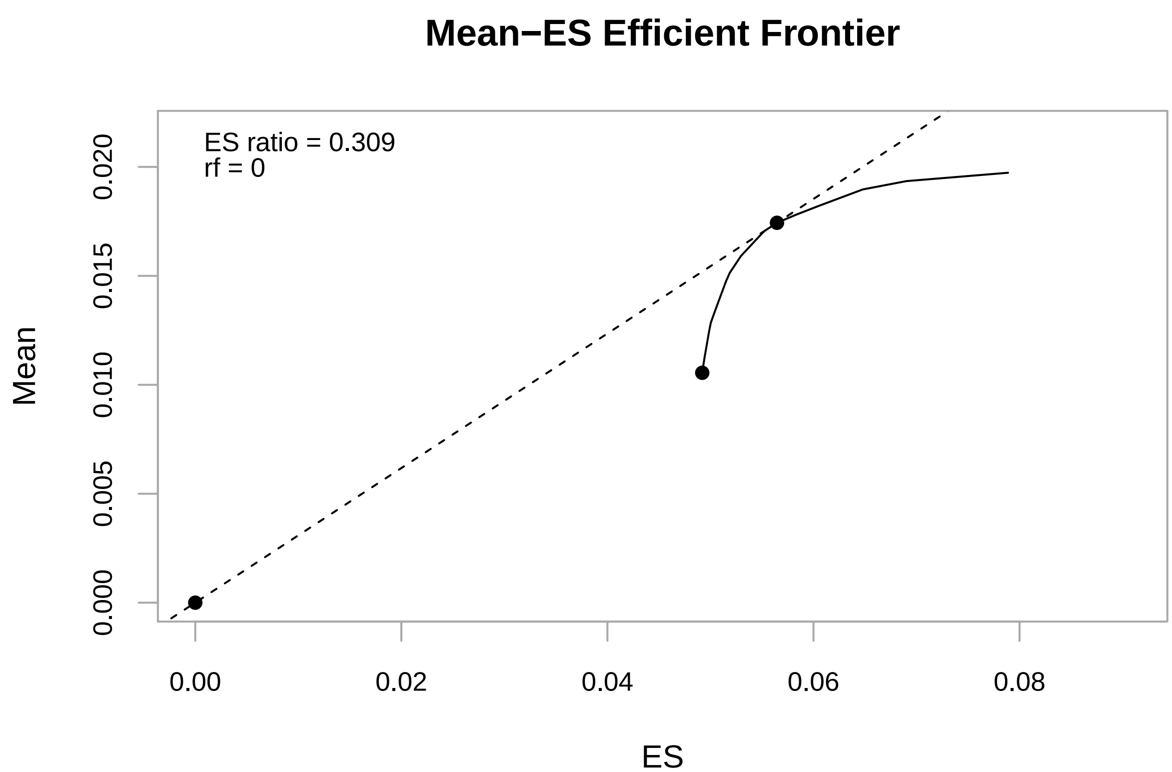


Fig 9.6

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

```
chart.EfficientFrontierOverlay(R=retM_CRSP_5, portfolio_list=portf_list,
                               type="mean-ES", match.col="ES",
                               legend.loc="topleft", 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(0.04, 0.12), ylim = c(0.005, 0.03))
```

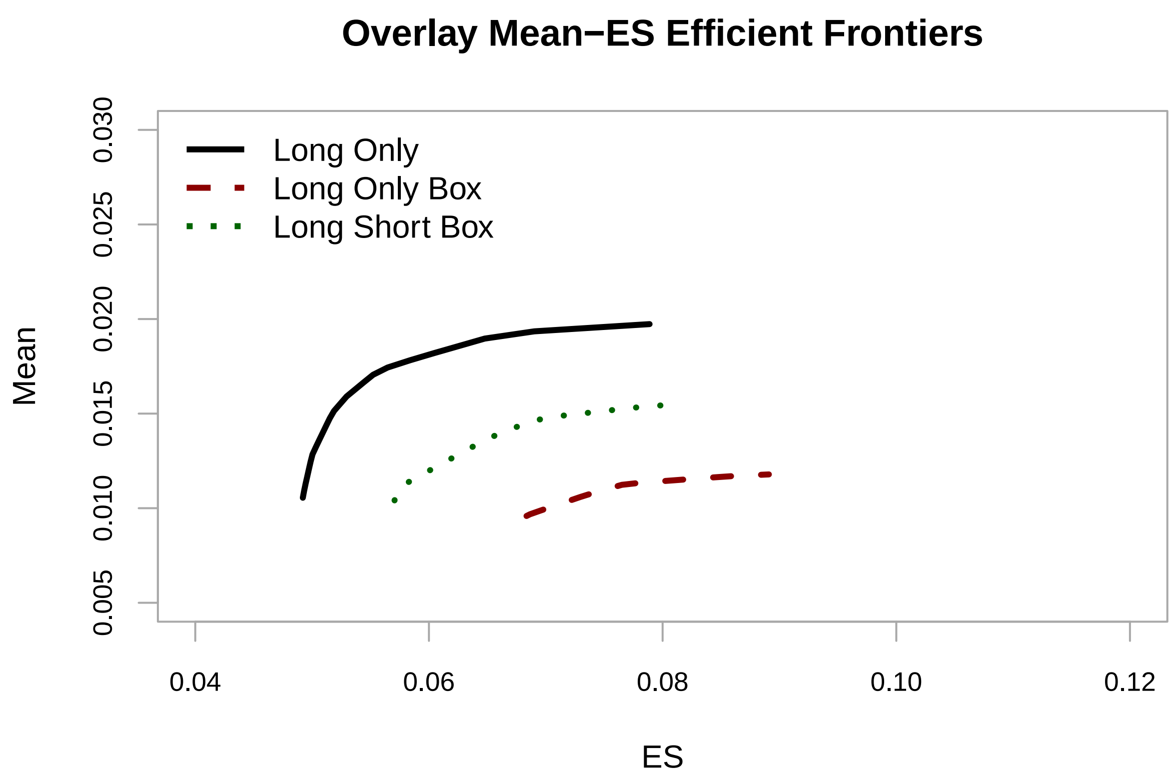


Fig 9.7

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

```
# 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="topleft", 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(0.03, 0.1), ylim = c(0.005, 0.025))
```

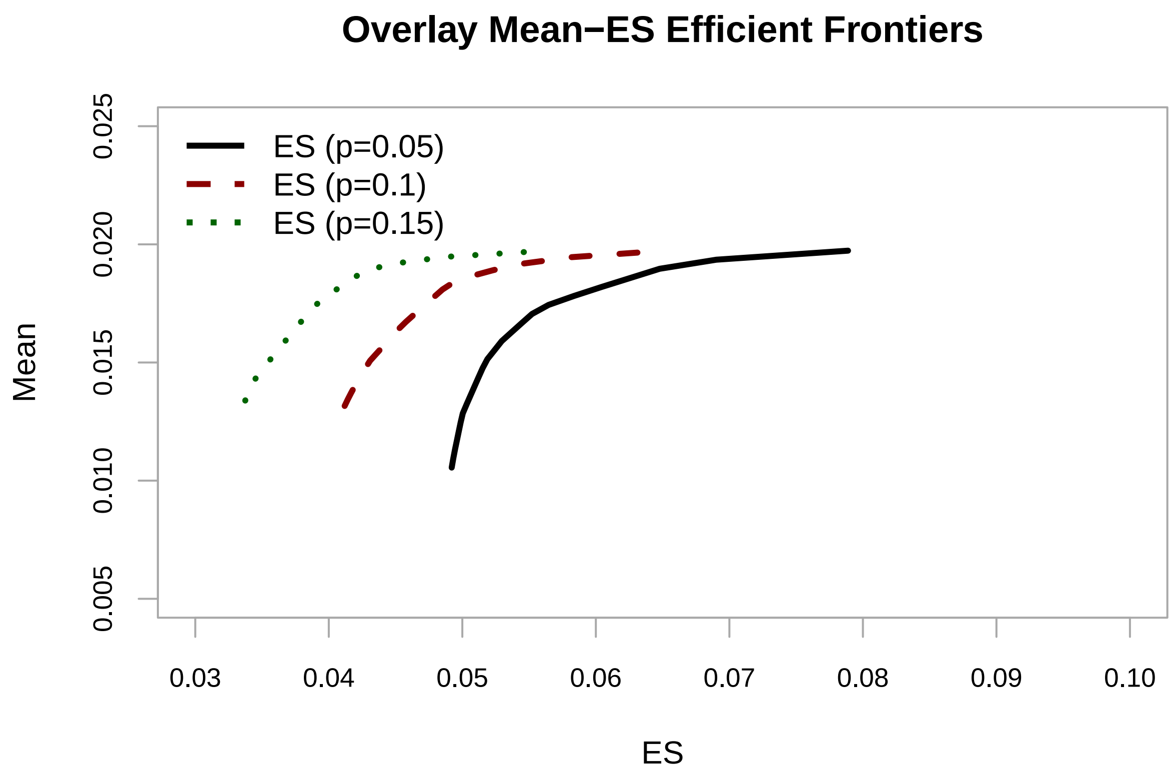


Fig 9.8

ES portfolio with a larger tail probability will have better performance.

### 9.3.3 Mean-EQS Efficient Frontier

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

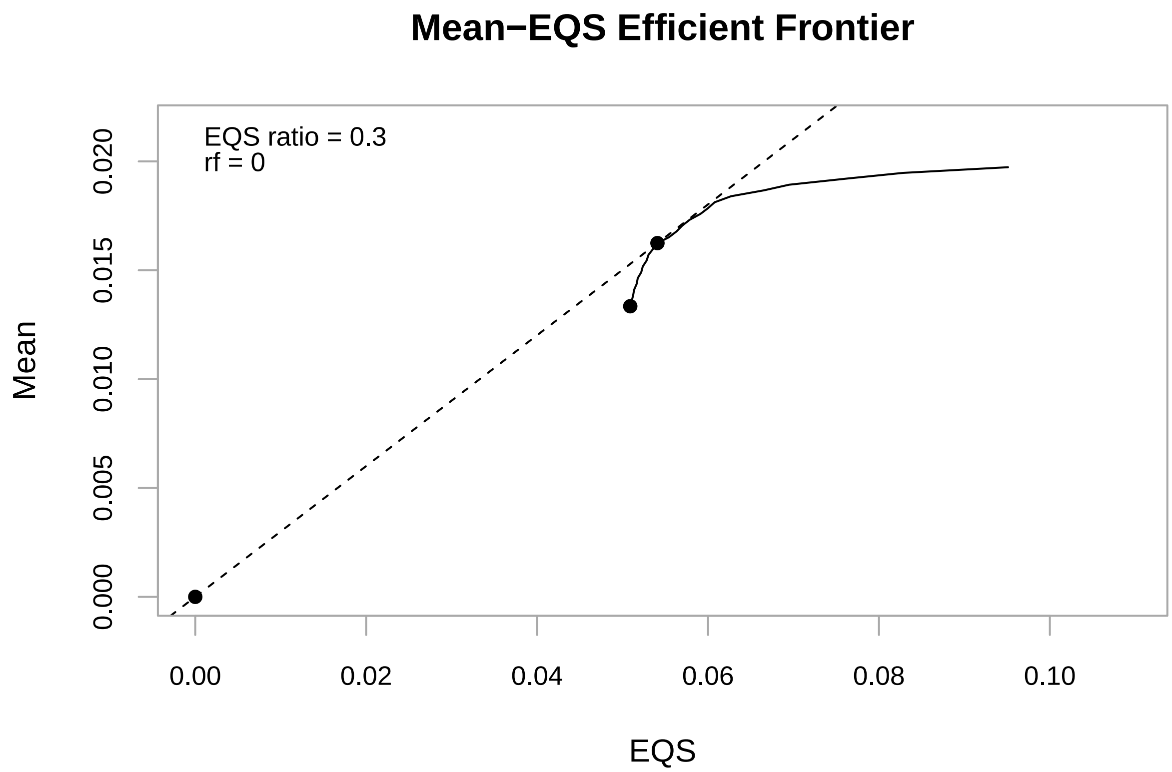


Fig 9.9

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