PortfolioAnalytics Constraints Functionality

Ross Bennett

July 2, 2013

Abstract

The purpose of this vignette is to demonstrate the functionality of constraints in PortfolioAnalytics.

Contents

1 Constraints

The following constraints are currently supported

weight_sum The weight_sum constraint is used to constrain the sum of weights. Common use cases of this are to apply a full investment, dollar neutral, or leverage constraint.

box Box constraints are used to constrain the minimum and maximum weights of assets. Standard box constraints with a single upper bound and single lower bound as well as per asset inequality constraints on weights can be specified. A special case of box constraints is a long only constraint where the minimum weight is 0 and maximum weight is 1.

group Group constraints are used to specify the minimum and maximum weights of groups of assets. A common use case to group assets by market cap or style. Note that group constraints is only implemented for the ROI solvers. Implementing the group constraints for other solvers should be possible in `constrained_objective` using the `constrained_group_tmp` function.

turnover Turnover can be specified as a constraint, but is not currently implemented in any solvers. Turnover constraint may not be able to be implemented in the ROI glpk

solver. It is implemented for the ROI quadprog solver in sandbox/testing_turnover.gmv.R. Currently, turnover can be implemented as an objective function and the function has been added to the file `R/objectiveFUN.R`. The user can specify a turnover target `turnover_target`. Any deviation from the target will be penalized.

diversification Diversification can be specified as a constraint, but is not currently implemented in any solvers. This can be done in the mapping function in the next part or implemented inside `constrained_objective`. The user can specify a diversification target value `div_target`. Any deviation from the target will be penalized.

volatility Volatility can be specified as a constraint, but it is not currently implemented for any solvers. This can be done in the mapping function in the next part or implemented inside `constrained_objective`. See `constrained_objective` for how volatility is handled as an objective. The user can specify a volatility target value `vol_target`. Any deviation from the target will be penalized.

position_limit Integer constraint for max position cardinality constraint. This may be able to be implemented in `randomize_portfolio` by generating portfolios with the number of non-zero weights equal to `max_pos`, then fill in weights of zero so the length of the weights vector is equal to the number of assets, then scramble the weights vector. The number of non-zero weights could also be random so that the number of non-zero weights is not always equal to `max_pos`. This could be implemented in the DEoptim solver with the mapping function. This might be do-able in Rglpk for max return and min ETL. Rglpk supports mixed integer types, but solve.QP does not. May be able to use branch-and-bound technique using solve.QP.

Constraint TODO

Quadratic Need more help on this. Note that the ROI solvers quadprog and glpk do not support quadratic constraints, they only support linear constraints. The ROI pluging for cplex does support quadratic constraints, but this is a commercial product. What are some use case examples other than diversification and volatility?

```
library(PortfolioAnalytics)

## Loading required package:  zoo
##
## Attaching package:  'zoo'
## The following object(s) are masked from 'package:base':
##
## as.Date, as.Date.numeric
```

```
## Loading required package:  xts
## Loading required package:  PerformanceAnalytics
##
## Attaching package:  'PerformanceAnalytics'
## The following object(s) are masked from 'package:graphics':
##
## legend


data(edhec)
ret <- edhec[, 1:4]
fund.names <- colnames(ret)

pspec <- portfolio.spec(assets = fund.names)

## assuming equal weighted seed portfolio
```

Add full investment constraint

```
pspec <- add.constraint(portfolio = pspec, type = "weight_sum", min_sum = 1,
    max_sum = 1, enabled = TRUE)
pspec$constraints[[1]]

## $type
## [1] "weight_sum"
##
## $enabled
## [1] TRUE
##
## $min_sum
## [1] 1
##
## $max_sum
## [1] 1
##
## $call
## add.constraint(portfolio = pspec, type = "weight_sum", enabled = TRUE,
##     min_sum = 1, max_sum = 1)
##
## attr(,"class")
## [1] "weight_sum_constraint" "constraint"
```

Add box constraints for long only

```
pspec <- add.constraint(portfolio = pspec, type = "box", min = 0, max = 1, enabled =

## min not passed in as vector, replicating min to length of length(assets)
## max not passed in as vector, replicating max to length of length(assets)

pspec$constraints[[2]]

## $type
## [1] "box"
##
## $enabled
## [1] TRUE
##
## $min
## Convertible Arbitrage            CTA Global Distressed Securities
##                     0                     0                     0
##      Emerging Markets
##                     0
##
## $max
## Convertible Arbitrage            CTA Global Distressed Securities
##                     1                     1                     1
##      Emerging Markets
##                     1
##
## $call
## add.constraint(portfolio = pspec, type = "box", enabled = TRUE,
##     min = 0, max = 1)
##
## attr(,"class")
## [1] "box_constraint" "constraint"
```

Update the box constraints to specify per asset weight constraints.

```
pspec <- add.constraint(portfolio = pspec, type = "box", min = c(0.05, 0.02,
    0.04, 0.06), max = c(0.35, 0.55, 0.55, 0.65), enabled = TRUE, indexnum = 2)
pspec$constraints[[2]]

## $type
```

```
## [1] "box"
##
## $enabled
## [1] TRUE
##
## $min
## Convertible Arbitrage            CTA Global Distressed Securities
##                  0.05                  0.02                  0.04
##      Emerging Markets
##                  0.06
##
## $max
## Convertible Arbitrage            CTA Global Distressed Securities
##                  0.35                  0.55                  0.55
##      Emerging Markets
##                  0.65
##
## $call
## add.constraint(portfolio = pspec, type = "box", enabled = TRUE,
##     min = c(0.05, 0.02, 0.04, 0.06), max = c(0.35, 0.55, 0.55,
##         0.65), indexnum = 2)
##
## attr(,"class")
## [1] "box_constraint" "constraint"
```

Add group constraints. The assets are grouped in 2 groups of 2 assets. The sum of asset weights of the first group must be greater than or equal to 0.15 and less than or equal to 0.65. The sum asset weights of the second group must be greater than or equal to 0.25 and less than or equal to 0.55. Labels for the groups can be specified (e.g. size, asset class, style, etc.). By default, the group labels will be group1, group2, ..., groupN for N groups.

```
pspec <- add.constraint(portfolio = pspec, type = "group", groups = c(2, 2),
    group_labels = c("Style A", "Style B"), group_min = c(0.15, 0.25), group_max = c
        0.55), enabled = TRUE)
pspec$constraints[[3]]

## $type
## [1] "group"
##
```

```
## $enabled
## [1] TRUE
##
## $groups
## [1] 2 2
##
## $group_labels
## [1] "Style A" "Style B"
##
## $cLO
## [1] 0.15 0.25
##
## $cUP
## [1] 0.65 0.55
##
## $call
## add.constraint(portfolio = pspec, type = "group", enabled = TRUE,
##     groups = c(2, 2), group_labels = c("Style A", "Style B"),
##     group_min = c(0.15, 0.25), group_max = c(0.65, 0.55))
##
## attr(,"class")
## [1] "group_constraint" "constraint"
```

Add turnover constraint. Any deviation from `turnover_target` is penalized.

```
pspec <- add.constraint(portfolio = pspec, type = "turnover", turnover_target = 0.6,
    enabled = TRUE)
pspec$constraints[[4]]

## $type
## [1] "turnover"
##
## $enabled
## [1] TRUE
##
## $turnover_target
## [1] 0.6
##
## $call
## add.constraint(portfolio = pspec, type = "turnover", enabled = TRUE,
```

```
##     turnover_target = 0.6)
##
## attr(,"class")
## [1] "turnover_constraint" "constraint"
```

Add diversification constraint. Any deviation from `div_target` will be penalized.

```
pspec <- add.constraint(portfolio = pspec, type = "diversification", div_target = 0
    enabled = TRUE)
pspec$constraints[[5]]

## $type
## [1] "diversification"
##
## $enabled
## [1] TRUE
##
## $div_target
## [1] 0.7
##
## $call
## add.constraint(portfolio = pspec, type = "diversification", enabled = TRUE,
##     div_target = 0.7)
##
## attr(,"class")
## [1] "diversification_constraint" "constraint"
```

Add volatility constraint. Any deviation from `vol_target` will be penalized.

```
pspec <- add.constraint(portfolio = pspec, type = "volatility", vol_target = 0.035,
    enabled = TRUE)
pspec$constraints[[6]]

## $type
## [1] "volatility"
##
## $enabled
## [1] TRUE
##
## $vol_target
```

```
## [1] 0.035
##
## $call
## add.constraint(portfolio = pspec, type = "volatility", enabled = TRUE,
##     vol_target = 0.035)
##
## attr(,"class")
## [1] "volatility_constraint" "constraint"
```

Add position_limit constraint. Constraint on the maximum number of positions or number of assets with non-zero weights.

```
pspec <- add.constraint(portfolio = pspec, type = "position_limit", max_pos = 3,
    enabled = TRUE)
pspec$constraints[[7]]

## $type
## [1] "position_limit"
##
## $enabled
## [1] TRUE
##
## $max_pos
## [1] 3
##
## $call
## add.constraint(portfolio = pspec, type = "position_limit", enabled = TRUE,
##     max_pos = 3)
##
## attr(,"class")
## [1] "position_limit_constraint" "constraint"
```