Creating a Portfolio Object with PortfolioAnalytics

Ross Bennett

August 12, 2013

Abstract

The purpose of this vignette is to demonstrate the new interface in PortfolioAnalytics to specify a portfolio object and to add constraints and objectives.

Contents

1 Getting Started

1.1 Load Packages

Load the necessary packages.

```
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
## Warning:  package 'PerformanceAnalytics' was built under R version 2.15.2
##
## Attaching package:  'PerformanceAnalytics'
## The following object(s) are masked from 'package:graphics':
##
## legend
```

1.2 Data

The edhec data set from the PerformanceAnalytics package will be used as example data.

```
data(edhec)

# Use the first 4 columns in edhec for a returns object
returns <- edhec[, 1:4]
print(head(returns, 5))

##            Convertible Arbitrage CTA Global Distressed Securities
## 1997-01-31                0.0119     0.0393                0.0178
## 1997-02-28                0.0123     0.0298                0.0122
## 1997-03-31                0.0078    -0.0021               -0.0012
## 1997-04-30                0.0086    -0.0170                0.0030
## 1997-05-31                0.0156    -0.0015                0.0233
##            Emerging Markets
## 1997-01-31           0.0791
## 1997-02-28           0.0525
## 1997-03-31          -0.0120
## 1997-04-30           0.0119
## 1997-05-31           0.0315
```

```
# Get a character vector of the fund names
fund.names <- colnames(returns)
```

2 Creating the ”portfolio” object

The portfolio object is instantiated with the `portfolio.spec` function. The main argument to `portfolio.spec` is assets, this is a required argument. The assets argument can be a scalar value for the number of assets, a character vector of fund names, or a named vector of seed weights. If seed weights are not specified, an equal weight portfolio will be assumed.

The `pspec` object is an S3 object of class ”portfolio”. When first created, the portfolio object has an element named `assets` with the seed weights, an element named `category_labels`, an element named `weight_seq` with a seed sequence of weights if specified, an empty constraints list and an empty objectives list.

```
# Specify a portfolio object by passing a character vector for the assets
# argument.
pspec <- portfolio.spec(assets = fund.names)
print.default(pspec)

## $assets
## Convertible Arbitrage            CTA Global Distressed Securities
##                  0.25                  0.25                  0.25
##      Emerging Markets
##                  0.25
##
## $category_labels
## NULL
##
## $weight_seq
## NULL
##
## $constraints
## list()
##
## $objectives
## list()
##
```

```
## $call
## portfolio.spec(assets = fund.names)
##
## attr(,"class")
## [1] "portfolio.spec" "portfolio"
```

3 Adding Constraints to the Portfolio Object

Adding constraints to the portfolio object is done with `add.constraint`. The `add.constraint` function is the main interface for adding and/or updating constraints to the portfolio object. This function allows the user to specify the portfolio to add the constraints to, the type of constraints, arguments for the constraint, and whether or not to enable the constraint (`enabled=TRUE` is the default). If updating an existing constraint, the indexnum argument can be specified.

Here we add a constraint that the weights must sum to 1, or the full investment constraint.

```
# Add the full investment constraint that specifies the weights must sum
# to 1.
pspec <- add.constraint(portfolio = pspec, type = "weight_sum", min_sum = 1,
    max_sum = 1)

# The full investment constraint can also be specified with
# type='full_investment' pspec <- add.constraint(portfolio=pspec,
# type='full_investment')

# Another common constraint is that portfolio weights sum to 0.  This can
# be specified any of the following ways pspec <-
# add.constraint(portfolio=pspec, type='weight_sum', min_sum=0, max_sum=0)
# pspec <- add.constraint(portfolio=pspec, type='dollar_neutral') pspec <-
# add.constraint(portfolio=pspec, type='active')
```

Here we add box constraints for the asset weights so that the minimum weight of any asset must be greater than or equal to 0.05 and the maximum weight of any asset must be less than or equal to 0.4. The values for min and max can be passed in as scalars or vectors. If min and max are scalars, the values for min and max will be replicated as vectors to the length of assets. If min and max are not specified, a minimum weight of 0 and maximum weight of 1 are assumed. Note that min and max can be specified as vectors with different weights for linear inequality constraints.

```
# Add box constraints
pspec <- add.constraint(portfolio = pspec, type = "box", min = 0.05, max = 0.4)

# min and max can also be specified per asset pspec <-
# add.constraint(portfolio=pspec, type='box', min=c(0.05, 0, 0.08, 0.1),
# max=c(0.4, 0.3, 0.7, 0.55))

# A special case of box constraints is long only where min=0 and max=1 The
# default action is long only if min and max are not specified pspec <-
# add.constraint(portfolio=pspec, type='box') pspec <-
# add.constraint(portfolio=pspec, type='long_only')
```

The portfolio object now has 2 objects in the constraints list. One object for the sum of weights constraint and another for the box constraint.

```
print(pspec)

## ***************************************************
## PortfolioAnalytics Portfolio Specification
## ***************************************************
##
## Call:
## portfolio.spec(assets = fund.names)
##
## Assets
## Number of assets: 4
##
## Constraints
## Number of constraints: 2
## Number of enabled constraints: 2
## Enabled constraint types
##  - weight_sum
##  - box
## Number of disabled constraints: 0
##
## Objectives
## Number of objectives: 0
## Number of enabled objectives: 0
## Number of disabled objectives: 0
```

The `summary` function gives a more detailed view of the constraints.

```
summary(pspec)

## ***************************************************
## PortfolioAnalytics Portfolio Specification Summary
## ***************************************************
## Assets and Seed Weights:
## Convertible Arbitrage            CTA Global Distressed Securities
##                  0.25                  0.25                  0.25
##      Emerging Markets
##                  0.25
##
## Constraints:
##
## ****************************************
## weight_sum constraint
## ****************************************
## $type
## [1] "weight_sum"
##
## $enabled
## [1] TRUE
##
## $message
## [1] FALSE
##
## $min_sum
## [1] 1
##
## $max_sum
## [1] 1
##
## $call
## add.constraint(portfolio = pspec, type = "weight_sum", min_sum = 1,
##     max_sum = 1)
##
## attr(,"class")
## [1] "weight_sum_constraint" "constraint"
##
```

```
##
## ****************************************
## box constraint
## ****************************************
## $type
## [1] "box"
##
## $enabled
## [1] TRUE
##
## $min
## Convertible Arbitrage            CTA Global Distressed Securities
##                  0.05                   0.05                   0.05
##      Emerging Markets
##                  0.05
##
## $max
## Convertible Arbitrage            CTA Global Distressed Securities
##                   0.4                    0.4                    0.4
##      Emerging Markets
##                   0.4
##
## $call
## add.constraint(portfolio = pspec, type = "box", min = 0.05, max = 0.4)
##
## attr(,"class")
## [1] "box_constraint" "constraint"
##
##
## Objectives:
```

Another common constraint that can be added is a group constraint. Group constraints are currently supported by the ROI, DEoptim, and random portfolio solvers. The following code groups the assets such that the first 3 assets are grouped together labeled GroupA and the fourth asset is in its own group labeled GroupB. The `group_min` argument specifies that the sum of the weights in GroupA must be greater than or equal to 0.1 and the sum of the weights in GroupB must be greater than or equal to 0.15. The `group_max` argument specifies that the sum of the weights in GroupA must be less than or equal to 0.85 and the sum of the weights in GroupB must be less than or equal to 0.55.The `group_labels` argument is optional and is useful for labeling groups in terms of market capitalization,

sector, etc.

```
# Add group constraints
pspec <- add.constraint(portfolio = pspec, type = "group", groups = c(3, 1),
    group_min = c(0.1, 0.15), group_max = c(0.85, 0.55), group_labels = c("GroupA",
        "GroupB"))
```

A position limit constraint can be added to limit the number of assets with non-zero, long, or short positions. The ROI solver used for maximizing return and ETL/ES/cVaR objectives support position limit constraints for `max_pos` (i.e. using the glpk plugin). `max_pos` is not supported for the ROI solver using the quadprog plugin. Note that `max_pos_long` and `max_pos_short` are not supported for either ROI solver. Position limit constraints are fully supported for DEoptim and random solvers.

```
# Add position limit constraint such that we have a maximum number of
# three assets with non-zero weights.
pspec <- add.constraint(portfolio = pspec, type = "position_limit", max_pos = 3)

# Can also specify maximum number of long positions and short positions
# pspec <- add.constraint(portfolio=pspec, type='position_limit',
# max_pos_long=3, max_pos_short=3)
```

A target diversification can be specified as a constraint. Diversification is defined as $diversification = \sum_{i=1}^{N} w_i$ for N assets. The optimizers work by applying a penalty if the diversification value is more than 5% away from `div_target`.

```
pspec <- add.constraint(portfolio = pspec, type = "diversification", div_target = 0
```

A target turnover can be specified as a constraint. The turnover is calculated from a set of initial weights. The initial weights can be specified, by default they are the seed weights in the portfolio object. The optimizers work by applying a penalty if the turnover value is more than 5% away from `turnover_target`. Note that the turnover constraint is not currently supported for the ROI solvers.

```
pspec <- add.constraint(portfolio = pspec, type = "turnover", turnover_target = 0.2)
```

A target mean return can be specified as a constraint.

```
pspec <- add.constraint(portfolio = pspec, type = "return", return_target = 0.007)
```

This demonstrates adding constraints to the portfolio object. As an alternative to adding constraints directly to the portfolio object, constraints can be specified as separate objects.

3.1 specifying Constraints as Separate Objects

The following examples will demonstrate how to specify constraints as separate objects for all constraints types.

```
# full investment constraint
weight_constr <- weight_sum_constraint(min_sum = 1, max_sum = 1)

# box constraint
box_constr <- box_constraint(assets = pspec$assets, min = 0, max = 1)

# group constraint
group_constr <- group_constraint(assets = pspec$assets, groups = c(3, 1), group_min
    0.15), group_max = c(0.85, 0.55), group_labels = c("GroupA", "GroupB"))

# position limit constraint
poslimit_constr <- position_limit_constraint(assets = pspec$assets, max_pos = 3)

# diversification constraint
div_constr <- diversification_constraint(div_target = 0.7)

# turnover constraint
to_constr <- turnover_constraint(turnover_target = 0.2)

# target return constraint
ret_constr <- return_constraint(return_target = 0.007)
```

4 Adding Objectives

Business objectives can be added to the portfolio object with `add.objective`. The `add.objective` function is the main function for adding and/or updating business objectives to the portfolio object. This function allows the user to specify the portfolio to

add the objectives to, the type (currently 'return', 'risk', or 'risk_budget'), name of the objective function, arguments to the objective function, and whether or not to enable the objective. If updating an existing constraint, the indexnum argument can be specified.

Here we add a risk objective to minimize portfolio variance. Note that the name of the function must correspond to a function in R. Many functions are available in the PerformanceAnalytics package.

```
pspec <- add.objective(portfolio = pspec, type = "risk", name = "var", enabled = TRU
```

The portfolio object now has 1 object in the objectives list for the risk objective we just added.

```
print(pspec$objectives)

## [[1]]
## $name
## [1] "var"
##
## $target
## NULL
##
## $arguments
## $arguments$portfolio_method
## [1] "single"
##
##
## $enabled
## [1] TRUE
##
## $multiplier
## [1] 1
##
## $call
## add.objective(portfolio = pspec, type = "risk", name = "var",
##     enabled = TRUE)
##
## attr(,"class")
## [1] "portfolio_risk_objective" "objective"
```

We now have a portfolio object with the following constraints and objectives to pass to `optimize.portfolio`.

- Constraints
  - weight_sum: The weights sum to 1 (i.e. full investment constraint)
  - box: minimum weight of any asset must be greater than or equal to 0.05 and the maximum weight of any asset must be less than or equal to 0.4.
- Objectives
  - risk objective: minimize portfolio var(iance).

5 Optimization

Note that this currently does not work, but is how I envision the portfolio object replacing the current constraint object.

```
# out <- optimize.portfolio(R=returns, portfolio=pspec,
# optimize_method='ROI')
```