Creating a Portfolio Object with PortfolioAnalytics

Ross Bennett

June 24, 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
##
## Attaching package:  'PerformanceAnalytics'
## The following object(s) are masked from 'package:graphics':
##
## legend

library(PerformanceAnalytics)  # just for edhec data set
```

1.2 Data

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

```
data(edhec)

# Use the first 4 indices 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 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)

## assuming equal weighted seed portfolio

print(pspec)

## $assets
## Convertible Arbitrage            CTA Global Distressed Securities
##                  0.25                  0.25                  0.25
##      Emerging Markets
##                  0.25
##
## $weight_seq
## NULL
##
## $constraints
## list()
##
## $objectives
## list()
##
## $call
## portfolio.spec(assets = fund.names)
##
## attr(,"class")
## [1] "portfolio.spec" "portfolio"
```

3 Adding Constraints

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 (currently 'weight_sum', 'box', or 'group'), arguments for the constraint, and whether or not to enable the constraint. 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, enabled = TRUE)
```

Here we add box constraints for the asset weights. 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.

```
pspec <- add.constraint(portfolio = pspec, type = "box", min = 0.05, max = 0.4,
    enabled = TRUE)

## 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)
```

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$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] "v2_constraint" "constraint"
##
## [[2]]
## $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", enabled = TRUE,
##     min = 0.05, max = 0.4)
##
## attr(,"class")
```

```
## [1] "v2_constraint" "constraint"
```

Another common constraint that can be added is a group constraint. Group constraints are currently only supported by the ROI solvers, see the ROI vignette [still need to make this] for examples using group constraints. Box constraints and weight_sum constraints are required by `optimize.portfolio`. Other constraint types will be added.

4 Adding Objectives

Business objectives can be added to the portfolio object with `add.objective_v2`. The `add.objective_v2` 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_v2(portfolio = pspec, type = "risk", name = "var", enabled =
```

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_v2(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')
```