The randPort package

Mike Flynn, Angel Zhou, and Dave Kane

June 20, 2013

Introduction

Stuff about making random portfolios and why they are important.

Methodology

getWeights

getWeights takes equality constraints in the form of

$$Ex = Ex_0$$

where x_0 is the weights of an original portfolio that we desire to match. It also guarantees that all the output weights will be positive. Starting from the original portfolio, we do a random walk based in the null space of E. This guarantees that the equality constraints will still hold via:

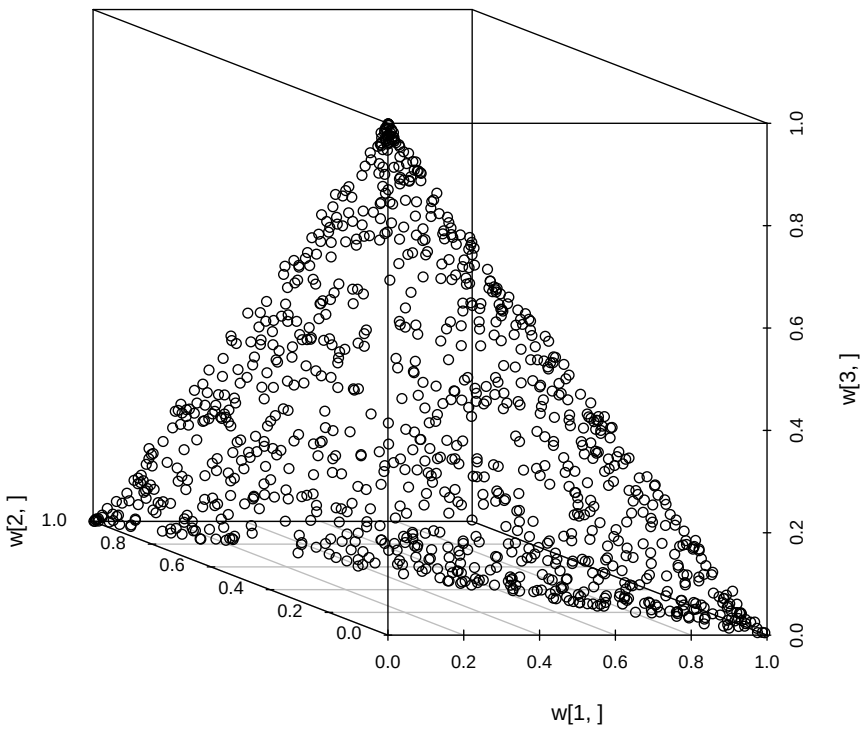
$$E(x_0 + v) = Ex_0 + Ev = Ex_0$$

if $Ev = 0$ (the definition of a vector in the null space). To fully sample the problem we have $v = Z * r$ where Z is a matrix whose columns span the null space, and r is a random vector whose coefficients $r_i \sim N(0, 1)$. This can be interpreted geometrically as moving in a random direction on the hyperplane that is the null-space. On each step, getWeights checks if any of the weights are negative, if they are, it projects the negative components onto the basis vectors in Z and subtracts those vectors from step. This guarantees that the equality constraints will still hold, while none of the weights will be less than zero.

Example

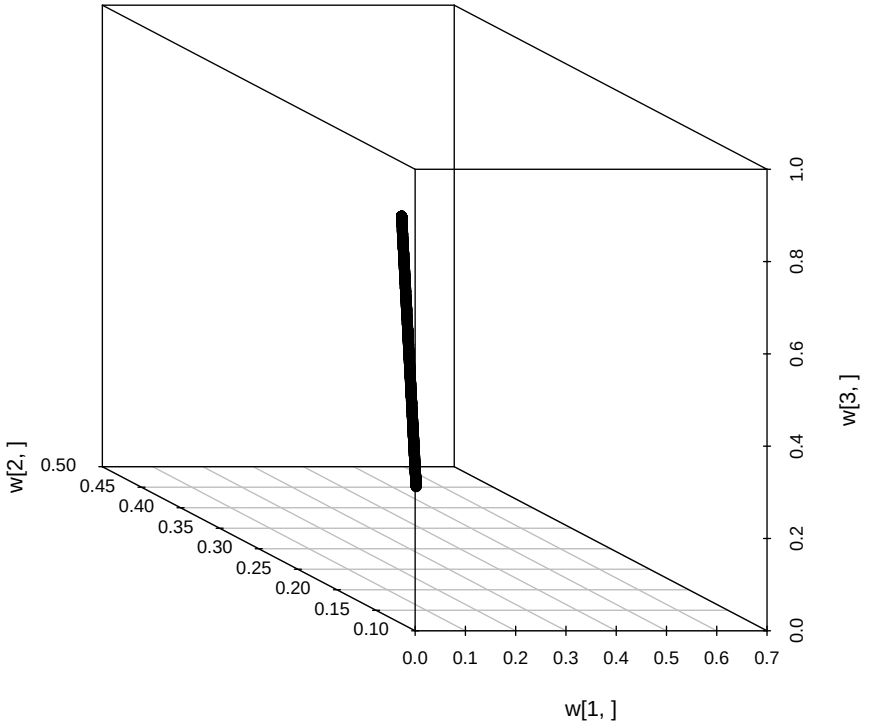
Consider a universe with only 3 stocks, and our only constraint is that their weights must add up to 1. Let our original portfolio be represented by the vector $x_0 = (.3, .3, .4)$. Our equality matrix E can be represented by the one-row matrix $E = [1, 1, 1]$. All our solutions are necessarily in the plane $x + y + z = 1$. To check if getWeights does this, we can look at a scatterplot of the results.

```
library(scatterplot3d)
E = matrix(c(1, 1, 1), 1, 3)
x0 = c(0.3, 0.3, 0.4)
w = getWeights(E, x0, 1000)
scatterplot3d(x = w[1, ], y = w[2, ], z = w[3, ], angle = 160)
```



Adding an additional constraint effectively reduces the dimensionality of the solution space by 1, because if E has one more row, the null space will have one less column. In our example, Z will only have on vector, so we will essentially be stepping back and forth in one direction on the surface. Demonstrating:

```
E = rbind(E, rnorm(3))
## x0 can be the same
w = getWeights(E, x0, 1000)
scatterplot3d(x = w[1, ], y = w[2, ], z = w[3, ], angle = 160)
```


w[2,]
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
w[1,]
0.0
0.2
0.4
0.6
0.8
1.0
w[3,]