Portfolio Efficient Set

Niraj Sardar

Loading libraries

```
library(quantmod)
```

```
## Warning: package 'quantmod' was built under R version 4.1.3
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(ggplot2)
```

```
## Warning in register(): Can't find generic `scale_type` in package ggplot2 to
## register S3 method.
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:xts':
##
##     first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
ticks <- c("STZ", "CAT", "LLY", "CBRL")

retout <- NULL
retout <- xts(retout)

for(i in 1:length(ticks)){
  prices = getSymbols(ticks[i], auto.assign = F)
  returns <- periodReturn(prices, period = "monthly",
                          type = "arithmetic")
  retout <- merge.xts(retout, returns)
}
colnames(retout) <- ticks
retout <- retout['2010/2015']
retout <- na.omit(retout)
head(retout)
```

```
##                              STZ         CAT         LLY        CBRL
## 2010-01-28 19:00:00  0.009416196 -0.08334795 -0.01428166 -0.02711248
## 2010-02-25 19:00:00 -0.064676617  0.09207498 -0.02443185  0.18181821
## 2010-03-30 20:00:00  0.093085173  0.10166519  0.05474668  0.06181321
## 2010-04-29 20:00:00  0.111313801  0.08337308 -0.03451132  0.06446740
## 2010-05-27 20:00:00 -0.088122605 -0.10765161 -0.06233915  0.00931746
## 2010-06-29 20:00:00 -0.062424970 -0.01135612  0.02165291 -0.06562314
```

B. Calculate means and covariance matrix

```
meanret <- colMeans(retout,na.rm = T)
x1 = round(meanret, 5)
cat("The mean vector :\n")
```

```
## The mean vector :
```

```
print(x1)
```

```
##     STZ     CAT     LLY    CBRL
## 0.03421 0.00581 0.01275 0.01885
```

```
covar <- var(retout)
x2 = round(covar, 8)
cat("The covariance matrix: \n")
```

```
## The covariance matrix:
```

```
print(x2)
```

```
##             STZ        CAT        LLY       CBRL
## STZ  0.00745920 0.00105944 0.00058250 0.00119845
## CAT  0.00105944 0.00699707 0.00031562 0.00100841
## LLY  0.00058250 0.00031562 0.00154920 0.00029047
## CBRL 0.00119845 0.00100841 0.00029047 0.00405279
```

C. Creating one portfolio as an example to showcase method

The weight vector contains weights for each security. Weights add to 1.

```
weight <- c(.2, .3, .3, .2)
```

Calculate portfolio variance Using matrix algebra

```
weight <- as.matrix(weight)
dim(weight)
```

```
## [1] 4 1
```

```
meanret <- as.matrix(meanret)
dim(meanret)
```

```
## [1] 4 1
```

```
mretp <- t(weight) %*% meanret
sretp <- sqrt(t(weight) %*% covar %*% weight)

cat("The mean and sigma of portfolio returns: ", mretp, sretp)
```

```
## The mean and sigma of portfolio returns:  0.01617978 0.0416561
```

D. Run simulation with random weights

First, for niter iterations, let's create random portfolio weights.

```
set.seed(12)
niter <- 500   # Set the number of iterations here
randomnums <- data.frame(replicate(4, runif(niter, 1, 10)))

head(randomnums)
```

```
##         X1       X2       X3       X4
## 1 1.624248 3.921433 1.545892 6.809350
## 2 8.359977 4.151757 2.406675 8.817655
## 3 9.483596 5.183035 8.274220 3.444403
## 4 3.424437 2.660027 5.040517 1.732409
## 5 2.524133 5.642371 5.675101 5.134248
## 6 1.305061 7.567241 6.264320 3.112291
```

```
wt_sim <- randomnums / rowSums(randomnums)
cat("The weights after normalization are in wt_sim...")
```

```
## The weights after normalization are in wt_sim...
```

```
head(wt_sim)
```

```
##           X1        X2        X3        X4
## 1 0.11684463 0.2820988 0.1112079 0.4898488
## 2 0.35220570 0.1749135 0.1013932 0.3714877
## 3 0.35942787 0.1964368 0.3135926 0.1305427
## 4 0.26633999 0.2068870 0.3920327 0.1347403
## 5 0.13301817 0.2973448 0.2990696 0.2705674
## 6 0.07151443 0.4146680 0.3432709 0.1705466
```

Initializing Variables.

```
# initialize weight and Results matrices
weight <- matrix(data = NA, nrow = length(ticks), ncol = 1)
Results <- matrix(data = NA, nrow = niter, ncol = 6)
```

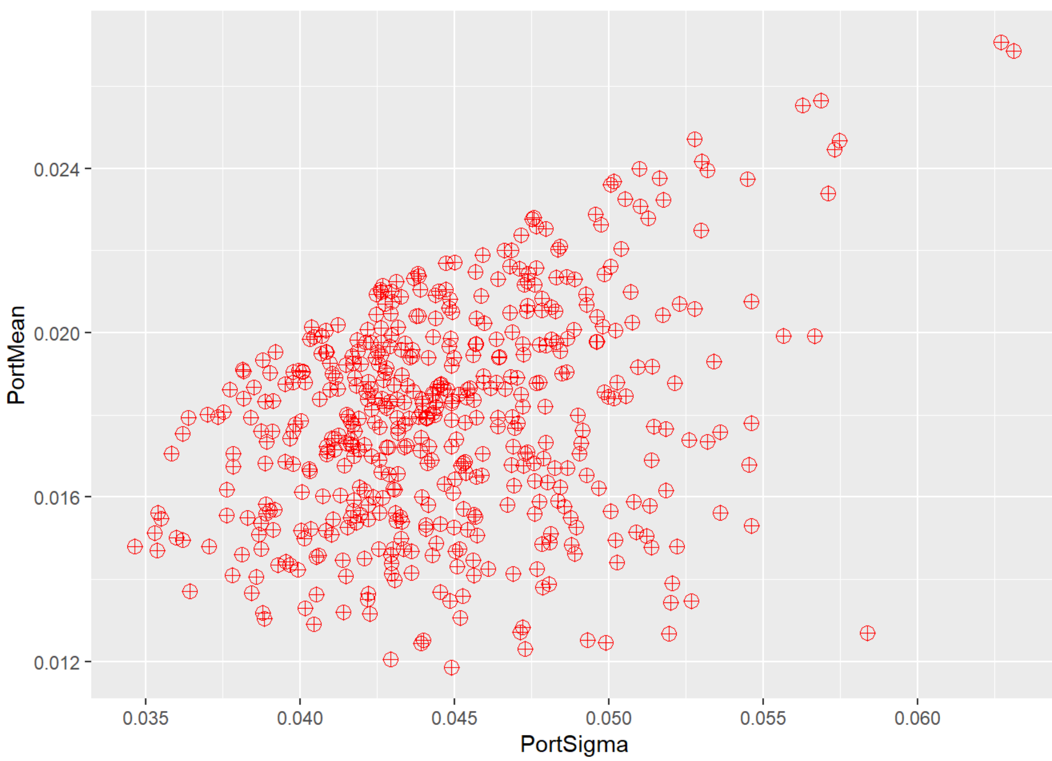
Run the simulations - this means, do portfolio calculations for each simulated portfolio.

```
# loop: each i is a portfolio
for (i in 1:niter){

    # inner loop places weights into Results
    for (k in 1:length(ticks)) {
             Results[i,k] = weight[k,1] = wt_sim[i,k]
    }

    Results[i,5] <- t(weight) %*% meanret                # portfolio mean
    Results[i,6] <- sqrt(t(weight) %*% covar %*% weight) # portfolio sigma
}

colnames(Results) <- c(ticks, "PortMean", "PortSigma")
Results <- as.data.frame(Results)
head(Results)
```

```
##          STZ       CAT       LLY      CBRL   PortMean  PortSigma
## 1 0.11684463 0.2820988 0.1112079 0.4898488 0.01628776 0.04693199
## 2 0.35220570 0.1749135 0.1013932 0.3714877 0.02136079 0.04862590
## 3 0.35942787 0.1964368 0.3135926 0.1305427 0.01989614 0.04430372
## 4 0.26633999 0.2068870 0.3920327 0.1347403 0.01785103 0.04003521
## 5 0.13301817 0.2973448 0.2990696 0.2705674 0.01519099 0.04083742
## 6 0.07151443 0.4146680 0.3432709 0.1705466 0.01244703 0.04391907
```

Plotting the results.

```
ggplot(data = Results , aes(x = PortSigma, y = PortMean)) +
    geom_point(pch = 10, colour = "red", size = 3)
```



E. Optimization

Again, risk is bad while return is good, so one way to select the best portfolio(s) is to identify a constraint based on one of these dimensions, and locate the best portfolio using the other.

```r
minmret = min(Results$PortMean)
maxmret = max(Results$PortMean)
seqmret = seq(round(minmret,3)-.001, maxmret+.001, .001)

optim <- Results %>% mutate(portnumber = index(Results)) %>%
    mutate(ints = cut(PortMean ,breaks = seqmret),
           lower = as.numeric( sub("\\((.+),.*", "\\1", ints) )) %>%
    group_by(ints) %>%
    summarise( lowerval = min(lower),
               sig_optim = min(PortSigma),
               retn_optim = PortMean[which.min(PortSigma)],
               numb = length(PortSigma),
               portID=portnumber[which.min(PortSigma)])

optim
```

```
## # A tibble: 17 x 6
##    ints          lowerval sig_optim retn_optim  numb portID
##    <fct>            <dbl>     <dbl>      <dbl> <int>  <int>
##  1 (0.011,0.012]    0.011    0.0449     0.0119     1    446
##  2 (0.012,0.013]    0.012    0.0405     0.0129    11    407
##  3 (0.013,0.014]    0.013    0.0364     0.0137    20    315
##  4 (0.014,0.015]    0.014    0.0347     0.0148    45    231
##  5 (0.015,0.016]    0.015    0.0353     0.0151    57    242
##  6 (0.016,0.017]    0.016    0.0376     0.0162    48    352
##  7 (0.017,0.018]    0.017    0.0358     0.0171    76    282
##  8 (0.018,0.019]    0.018    0.0370     0.0180    72    345
##  9 (0.019,0.02]     0.019    0.0382     0.0191    68    403
## 10 (0.02,0.021]     0.02     0.0404     0.0201    42    218
## 11 (0.021,0.022]    0.021    0.0426     0.0211    28    485
## 12 (0.022,0.023]    0.022    0.0466     0.0220    14     76
## 13 (0.023,0.024]    0.023    0.0500     0.0236    10    139
## 14 (0.024,0.025]    0.024    0.0528     0.0247     4    495
## 15 (0.025,0.026]    0.025    0.0563     0.0255     2     36
## 16 (0.026,0.027]    0.026    0.0631     0.0269     1    198
## 17 (0.027,0.028]    0.027    0.0627     0.0271     1    120
```

```r
ggplot(data = optim , aes(x = sig_optim, y = retn_optim)) +
    geom_point(pch = 10, colour = "red", size = 3)
```