Homework1

Jack Cheng

8/6/2017

R Markdown

This is the homework 1 for the second part of STA380 in Red McCombs business school.

Statistics Questions:

Question 1

From the question we know that:

P(RC)=0.3

P(TC)=1-P(RC)=0.7 since TC is the complement of RC

P(Y)=0.65

P(Y|RC)=0.5

Where RC denotes that the clicker is a random clicker, TC denotes the clicker is a truthful clicker and Y denotes the result is yes.

And we want to know P(Y|TC).

Solution:

P(Y,RC)=P(Y|RC)*P(RC)=0.5*0.3=0.15

P(Y,TC)=P(Y)-P(Y,RC)=0.65-0.15=0.5 since TC is the complement of RC

so P(Y|TC)=P(Y,TC)/P(TC)=0.5/0.7=0.7142857

Question 2

From the question we know that:

P(P|D)=0.993

P(N|Dc)=0.9999

P(D)=0.000025

Where D denotes with desease, Dc denotes no desease, P denotes positive and N denotes negative.

We want to know: P(D|P)

Solution:

since we know Dc is the complement of D

so P(Dc)=1-P(D)=0.999975 and P(P)=P(Dc,P)+P(D,P)

and N is the complement of P

so P(P|Dc)=1-P(N|Dc)=0.0001

P(D|P)=P(D,P)/P(P)

=(P(P|D)*P(D))/(P(D,P)+P(Dc,P))

=(P(P|D)*P(D))/(P(P|D)*P(D)+P(Dc,P)*P(Dc))

=0.993*0.000025/(0.993*0.000025+0.0001*0.999975)=0.1988824

Which is really high!

That is to say though the sensitivity and specificity of the test is really good, due to the fact that the prior probability of desease is so low as 0.000025, the false positive rate is still really high. This kind of implementing a universal testing policy for the disease will lead to panic and chaos.

Exploratory analysis: green buildings

Bootstrapping

```
library(mosaic)
```

```
## Loading required package: dplyr

##
## Attaching package: 'dplyr'

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

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

## Loading required package: lattice

## Loading required package: ggformula

## Loading required package: ggplot2

##
## New to ggformula?  Try the tutorials:
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")

## Loading required package: mosaicData

##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.
##
## Note: If you use the Matrix package, be sure to load it BEFORE loading mosaic.

##
## Attaching package: 'mosaic'

## The following objects are masked from 'package:dplyr':
##
##     count, do, tally
## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median,
##     prop.test, quantile, sd, t.test, var

## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

```r
library(quantmod)
```

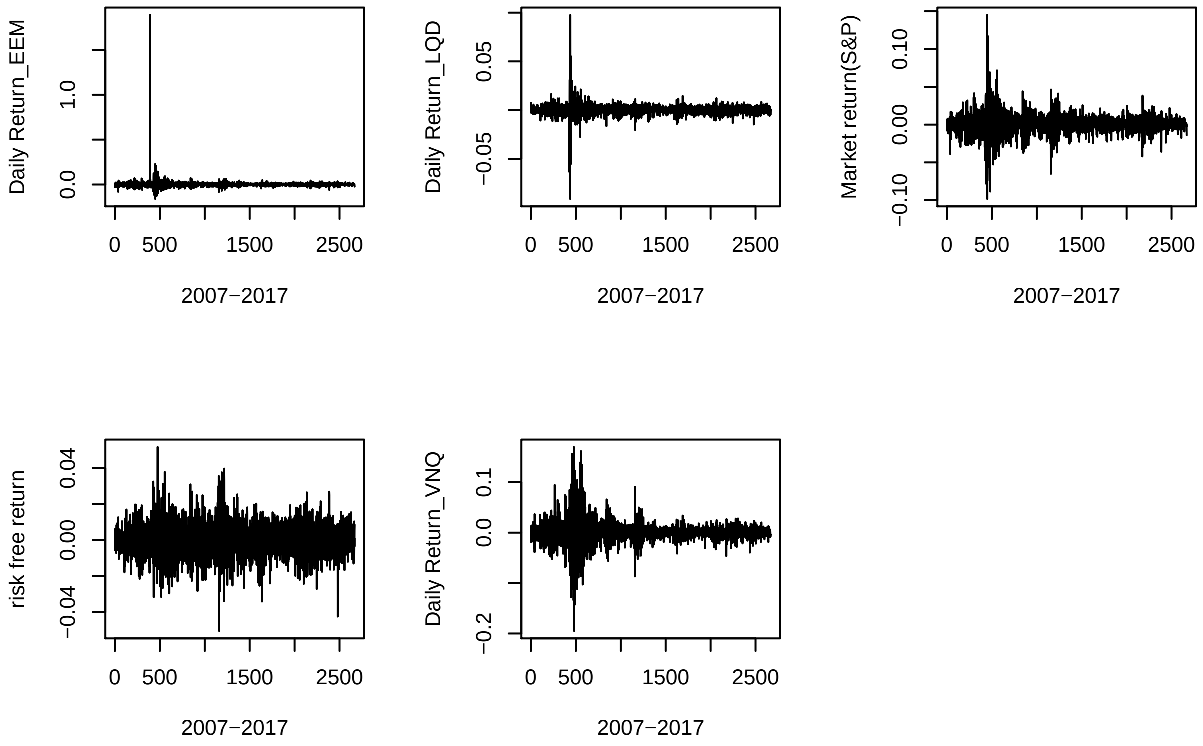
```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

##
## Attaching package: 'xts'

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

## Loading required package: TTR

## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```r
library(foreach)
mystocks = c("SPY", "TLT", "LQD","EEM","VNQ")
getSymbols(mystocks)
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).

## Warning: LQD contains missing values. Some functions will not work if
## objects contain missing values in the middle of the series. Consider using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.

## [1] "SPY" "TLT" "LQD" "EEM" "VNQ"
```

```r
EEMa = adjustOHLC(EEM)
LQDa = adjustOHLC(LQD)
```

```r
SPYa = adjustOHLC(SPY)
TLTa = adjustOHLC(TLT)
VNQa = adjustOHLC(VNQ)

all_returns = cbind(ClCl(EEMa),ClCl(LQDa),ClCl(SPYa),ClCl(TLTa),ClCl(VNQa))
all_returns = as.matrix(na.omit(all_returns))
par(mfrow=c(2,3))
plot(all_returns[,1], type='l', xlab="2007-2017",ylab="Daily Return_EEM")
plot(all_returns[,2], type='l', xlab="2007-2017",ylab="Daily Return_LQD")
plot(all_returns[,3], type='l',xlab="2007-2017",ylab="Market return(S&P)")
plot(all_returns[,4], type='l',xlab="2007-2017",ylab="risk free return")
plot(all_returns[,5], type='l',xlab="2007-2017",ylab="Daily Return_VNQ")
```



First let us just to explore the data by looking at the mean and variance of each asset to get a roughly idea about their risk return properties:

```r
library(plotly)
```

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:mosaic':
##
##     do

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter
## The following object is masked from 'package:graphics':
##
##     layout
```

```
plot_ly(type='box', yaxis= list(range = c(-0.5, 0.5))) %>%
  add_boxplot(y = all_returns[,3], name = 'SPY') %>%
  add_boxplot(y = all_returns[,4], name = 'TLT') %>%
  add_boxplot(y = all_returns[,2], name = 'LQD') %>%
  add_boxplot(y = all_returns[,1], name = 'EEM') %>%
  add_boxplot(y = all_returns[,5], name = 'VNQ') %>%
  layout(
    yaxis = list(range = c(-0.5,0.5)))
```

We can see through the graph that EEM and VNQ are with high volatility and rather high return and the TLT is just the most robust way of investing. Then let us see the sharp ratio for those assets first to get a more through measurement of these assets:

- There are many ways to measure the performance of a certain asset. Here we choose the sharp ratio, Jensen's alpha and treynor ratio as examples.

Note that sharp ratio is the ratio between extra mean return exceed risk free asset and the volitility of asset.

First use TLT as an approximation to the risk free asset. Calculate the extra return of each asset:

```
EEMa = adjustOHLC(EEM)
LQDa = adjustOHLC(LQD)
SPYa = adjustOHLC(SPY)
TLTa = adjustOHLC(TLT)
VNQa = adjustOHLC(VNQ)
EEM_extra_return=all_returns[,1]-all_returns[,4]
LQD_extra_return=all_returns[,2]-all_returns[,4]
Market_extra_return=all_returns[,3]-all_returns[,4]
VNQ_extra_return=all_returns[,5]-all_returns[,4]

EEM_SD=sd(EEM_extra_return)
LQD_SD=sd(LQD_extra_return)
Market_SD=sd(Market_extra_return)
VNQ_SD=sd(VNQ_extra_return)

SR_EEM=mean(EEM_extra_return)/EEM_SD
SR_LQD=mean(LQD_extra_return)/LQD_SD
SR_Market=mean(Market_extra_return)/Market_SD
SR_VNQ=mean(VNQ_extra_return)/VNQ_SD

SR_EEM
```

```
## [1] 0.01669493
```

```
SR_LQD
```

```
## [1] -0.009391961
```

```
SR_Market
```

```
## [1] 0.002428866
```

```
SR_VNQ
```

```
## [1] 0.0046315
```

Actually here we see that LQD have a lower mean return than risk free asset, that probably suggest us not to invest in this asset since this performance is really terrible. EEM got the highest sharp ratio.

Then we look at Jensen's alpha:

Jensen's alpha is a measurement of the return after adjusting by taking risk into account.

Fit the data with a CAPM model first:

```
lmEEM=lm(EEM_extra_return~Market_extra_return)
lmLQD=lm(LQD_extra_return~Market_extra_return)
lmVNQ=lm(VNQ_extra_return~Market_extra_return)
coef(lmEEM)
```

```
##         (Intercept) Market_extra_return
##         0.000688956         1.184530681
```

```
coef(lmLQD)
```

```
##         (Intercept) Market_extra_return
##       -9.757413e-05        3.552796e-01
```

```
coef(lmVNQ)
```

```
##         (Intercept) Market_extra_return
##        6.770747e-05        1.157167e+00
```

The alpha and beta are intercept and beta in this particular case.

We can see that EEM get the highest alpha while VNQ get the lowest.

Then let us look at the treynor ratio:

```
TR_EEM=mean(EEM_extra_return)/coef(lmEEM)[2]
TR_LQD=mean(LQD_extra_return)/coef(lmLQD)[2]
TR_VNQ=mean(VNQ_extra_return)/coef(lmVNQ)[2]

TR_EEM
```

```
## Market_extra_return
##        0.0006276331
```

```
TR_LQD
```

```
## Market_extra_return
##       -0.0002286351
```

```
TR_VNQ
```

```
## Market_extra_return
##        0.0001045167
```

We can see here that EEM is with the highest treynor ratio.

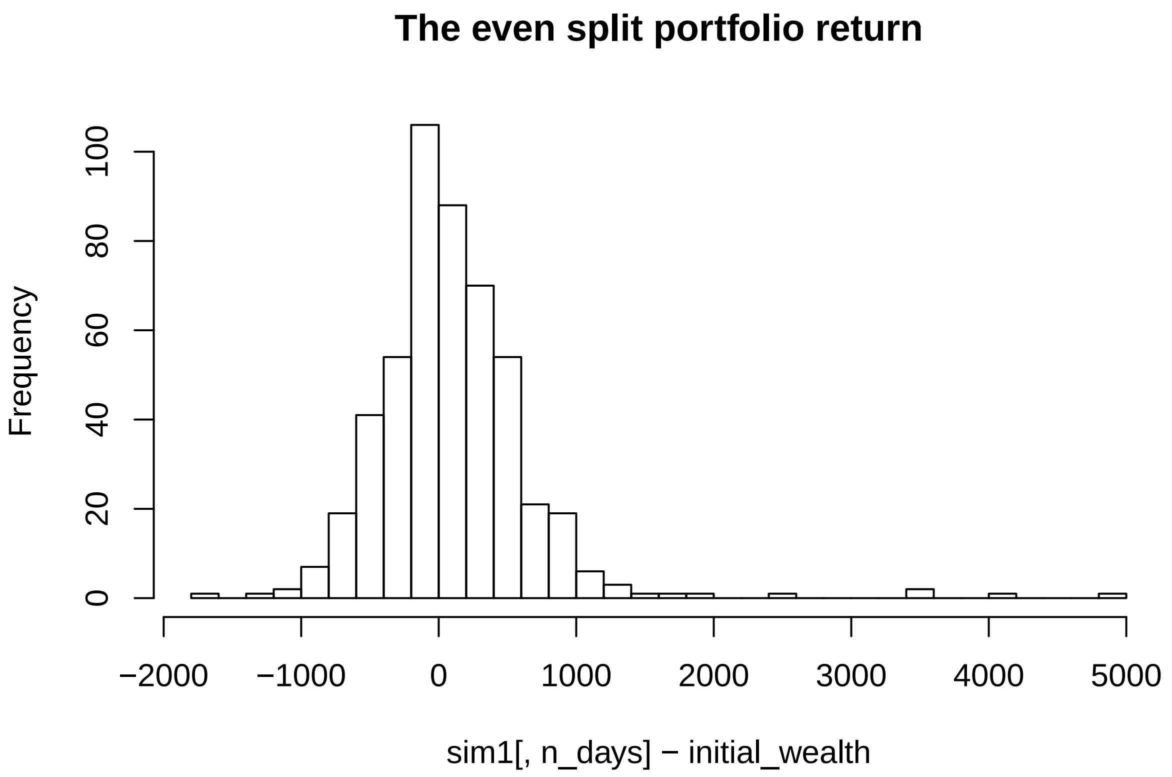
Now we begin the boostrapping simulation.

The even split one

```
set.seed(888)
initial_wealth = 10000
sim1 = foreach(i=1:500, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
    holdings = weights * total_wealth
    n_days = 20
    wealthtracker = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(all_returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
        wealthtracker[today] = total_wealth
    }
    wealthtracker
}
mean(sim1[,n_days]- initial_wealth)
```

```
## [1] 99.38072
```

```
hist(main = 'The even split portfolio return',sim1[,n_days]- initial_wealth, breaks=30)
```

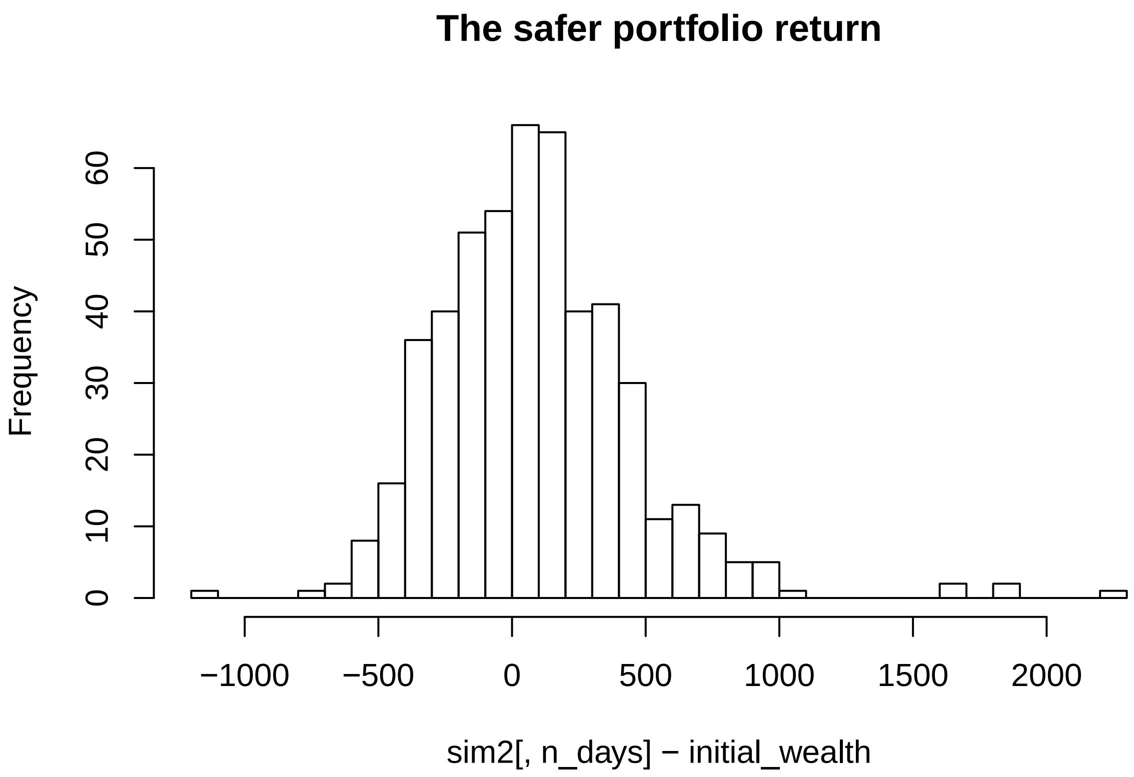


The safer portfolio Next we have our safe portfolio as we invest more on the TLT and since we analysis that LQD is inefficient so we do not invest in it.

```
set.seed(888)
initial_wealth = 10000
sim2 = foreach(i=1:500, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(0.1, 0, 0.1, 0.7, 0.1)
    holdings = weights * total_wealth
    n_days = 20
    wealthtracker = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(all_returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
        wealthtracker[today] = total_wealth
    }
    wealthtracker
}
mean(sim2[,n_days]- initial_wealth)
```

```
## [1] 87.44794
```

```
hist(main = 'The safer portfolio return',sim2[,n_days]- initial_wealth, breaks=30)
```



The more aggressive one.

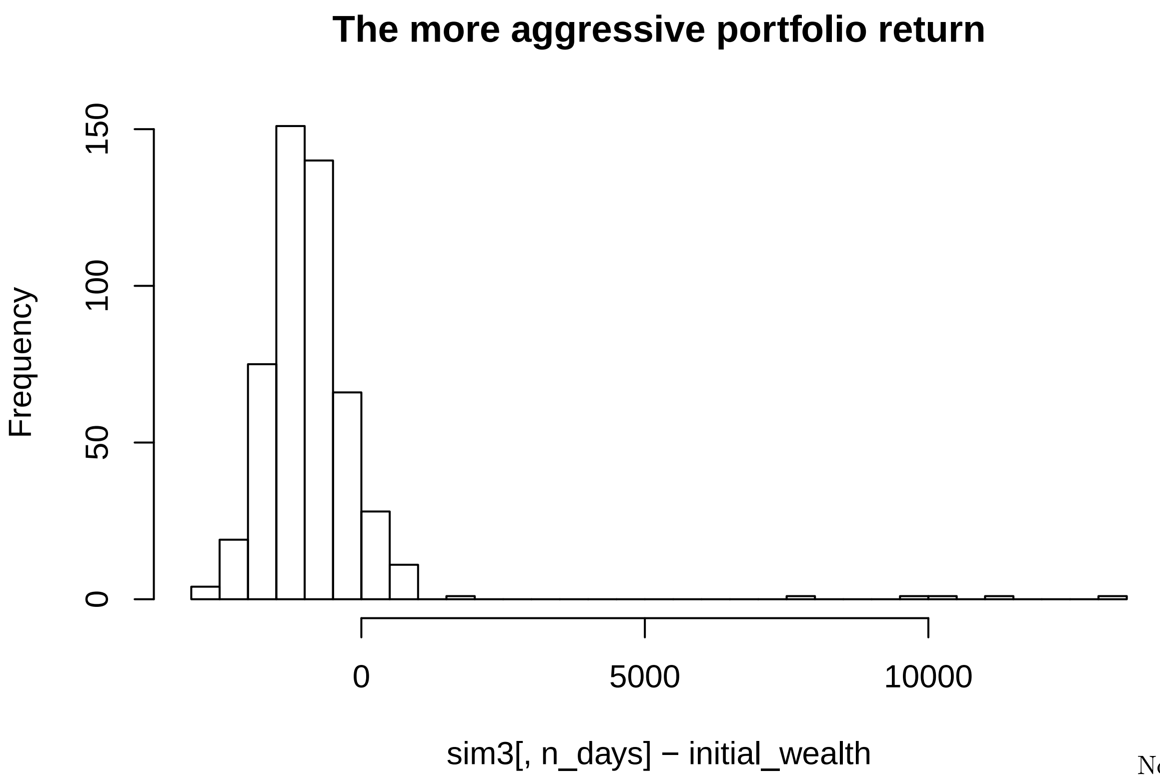
In this one we choose to invest in more EEM which is a more risky asset.

```
set.seed(888)
initial_wealth = 10000
sim3 = foreach(i=1:500, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(0.6, 0, 0.1, 0.1, 0.1)
    holdings = weights * total_wealth
    n_days = 20
    wealthtracker = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(all_returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
```

```r
        wealthtracker[today] = total_wealth
    }
    wealthtracker
}
mean(sim3[,n_days]- initial_wealth)
```

```
## [1] -845.7464
```

```r
hist(main = 'The more aggressive portfolio return',sim3[,n_days]- initial_wealth, breaks=30)
```



Now let us look at the value at risk at 5% level of these portfolios:

```r
quantile(sim1[,n_days], 0.05) - initial_wealth
```

```
##        5%
## -635.7145
```

```r
quantile(sim2[,n_days], 0.05) - initial_wealth
```

```
##        5%
## -409.0383
```

```r
quantile(sim3[,n_days], 0.05) - initial_wealth
```

```
##        5%
## -1961.649
```

As we can see, here we have the value at risk of 5% for these three different portfolio with different style, and these numbers make sence as the aggressive one have the highest risk and the safe one is with the lowest risk.