Beixi_Lei_RM_HW3

Beixi Lei

4/24/2018

Part 1 Collect 1-year daily stock prieces for 10 companies

```
#Get adjusted close price
stocks <-  c("CL","MGA","ABX","NLY","MAS","DISCA","DISCK","VIAB","VIA"
            ,"GT")
  pricedata.env <- new.env()

StartDate = as.Date("2017-04-01")
EndDate = as.Date("2018-04-01")
  ### here we use l_ply so that we don't double save the data
  ### getSymbols() does this already so we just want to be memory efficient
  ### go through every stock and try to use getSymbols()
  l_ply(stocks, function(sym) try(getSymbols(sym,env=pricedata.env,from = StartDate, to
 = EndDate),silent=T))
```

```
## '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).
```

```
  ### now we only want the stocks that got stored from getSymbols()
  ### basically we drop all "bad" tickers
  stocks <- stocks[stocks %in% ls(pricedata.env)]

  ### now we just loop through and merge our good stocks
  ### if you prefer to use an lapply version here, that is also fine
  ### since now we are just collecting all the good stock xts() objects
  pricedata <- xts()
  for(i in seq_along(stocks)) {
    symbol <- stocks[i]
    pricedata <- merge(pricedata, Ad(get(symbol,envir=pricedata.env)))
    #names(pricedata)[i] <- paste("close.price")
    #diff <- diff(pricedata)
  }

#Replace NA value to 0
#diff[is.na(diff)] <- 0
#diff
#pricedata
```

Part 2 Compute daily returns, and the mean and standard deviation

```
#sktpool is a data.frame
stkpool = c("CL","MGA","ABX","NLY","MAS","DISCA","DISCK","VIAB","VIA"
            ,"GT")

# Get data from Yahoo!Finance
for(stk in stkpool){
  getSymbols(stk, from = StartDate, to = EndDate)
  expr <- paste(stk, " = data.frame(dailyReturn(", stk, "))", sep="")
  na.omit(stk)
  eval(parse(text = expr))
  # CL = data.frame(date = index(dailyReturn(CL)), dailyReturn(CL))
}

stock <- cbind(CL,MGA,ABX,NLY,MAS,DISCA,DISCK,VIAB,VIA,GT)
#stock

stockmean <- matrix(rep(0),nrow = 10, ncol=1)
stockdeviation <- matrix(rep(0),nrow = 10, ncol=1)
for (i in 1:10){
  stockmean[i] <- mean(stock[i]$daily.returns)
  stockdeviation[i] <- sd(stock[i]$daily.returns)
}

#stockmean
#stockdeviation
```

Part 3 Collect current and long-term liabilities of each firm from finance.yahoo.com

```
currentliabilities <- data.frame(rep(0),nrow=1,ncol=10)
currentliabilities <- c(3408000,
                        7276000,
                        1747000,
                        0,
                        1628000,
                        1871000,
                        1871000,
                        3753000,
                        3753000,
                        5025000)

longtermliabilities <- data.frame(rep(0),nrow=11,ncol=10)
longtermliabilities <- c(6566000,
                         2346000,
                         6364000,
                         607854,
                         2969000,
                         14568000,
                         14568000,
                         11100000,
                         11100000,
                         5043000)
```

Part 4 Run the program and retrieve the default probability and expected recovery of each firm

```
#required packages
library(rootSolve)

#inputs
Ve = 5 #value of equity
vol = 0.3 #volatility of equity
r = 0.06 #risk-free rate
t = 5 #time horizon
D = 3 #value of debt

merton = function(x)
{c(f1 = (pnorm((log(x[1] / D) + (r + 0.5 * (x[2] ^ 2)) * t)/ (x[2] * sqrt(t))) * x[2] *
 x[1]) - (Ve * vol),
f2 = (x[1] * pnorm((log(x[1] / D) + (r + 0.5 * (x[2] ^ 2)) * t) / (x[2] * sqrt(t))) - D
 * exp(-r * t) *
pnorm(((log(x[1] / D) + (r + 0.5 * (x[2] ^ 2)) * t) / (x[2] * sqrt(t))) - x[2] * sqrt
(t))) - Ve)}

roots = multiroot(merton, c(15, 0.2))

#outputs
v0 = roots$root[1]  #value of assets
volv = roots$root[2]  #volatility of assets
v0
```

```
## [1] 7.219107
```

```
volv
```

```
## [1] 0.2083816
```

```
#calculate probability of default and recovery
market_val = v0 - Ve
payment = D * exp(-r * t)

prob_default = pnorm(-(((log(v0 / D) + (r + 0.5 * (volv ^ 2)) * t) / (volv * sqrt(t))) -
 volv * sqrt(t)))
prob_default
```

```
## [1] 0.01085465
```

```
expected_loss = (payment - market_val) / (payment)
recovery = 1 - (expected_loss / prob_default)
recovery
```

```
## [1] 0.8612303
```