final.R

Dipro

2021-05-20

```
# Question 1
# ====================

# Q 1.1

library(quantmod)
```

```
## Warning: package 'quantmod' was built under R version 4.0.4
```

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

```
## Warning: package 'xts' was built under R version 4.0.4
```

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

```
## Warning: package 'zoo' was built under R version 4.0.4
```

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

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

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

```
## Warning: package 'TTR' was built under R version 4.0.4
```

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

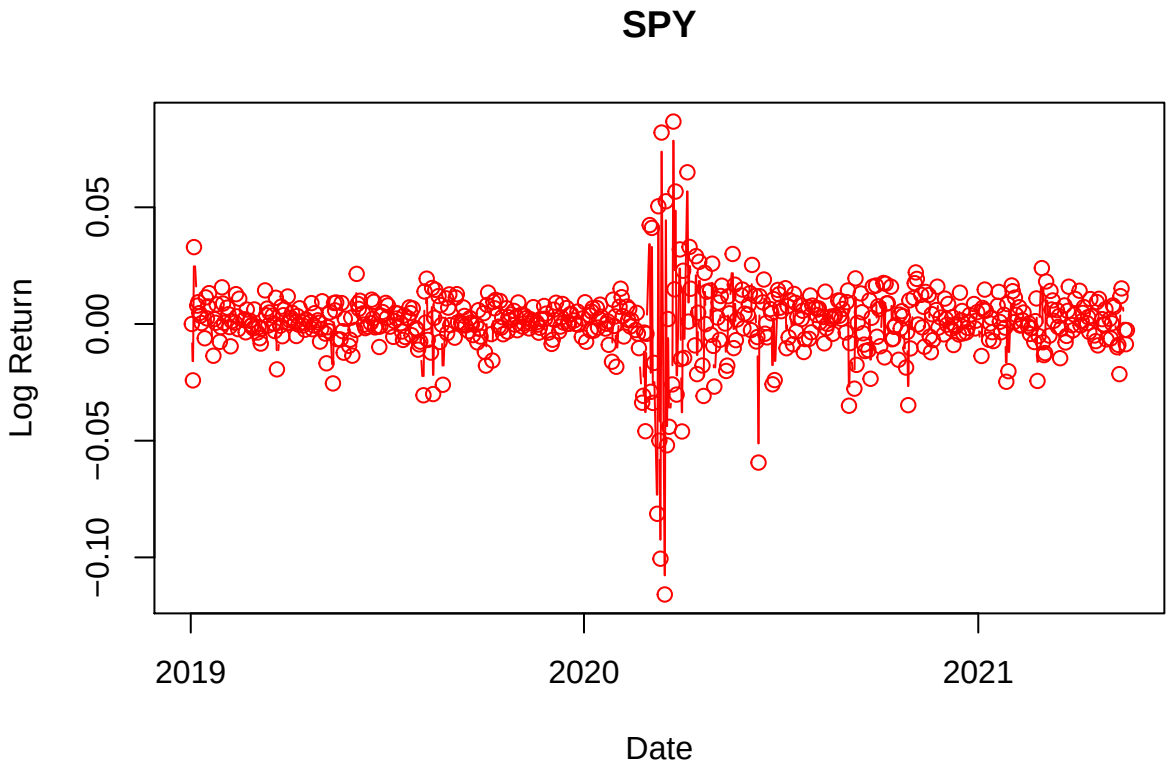
```
getSymbols(Symbols = "SPY", from = "2019-01-01")
```

```
## '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.
```

```
## [1] "SPY"
```

```
SPY <- data.frame(SPY)

#head(SPY)
#tail(SPY)
```

```r
# Q 1.2
SPY.price <- SPY$SPY.Adjusted
SPY.log.price <- log(SPY.price)
SPY.log.return <- diff(SPY.log.price)

# Q 1.3
#SPY
SPY$date <- as.Date(rownames(SPY))
SPY$log.return <- c(0 , SPY.log.return)

#head(SPY)

plot(SPY$date , SPY$log.return , type="b" ,
     xlim = c(as.numeric(min(SPY$date)),as.numeric(max(SPY$date))),
     ylim = c(min(SPY$log.return) , max(SPY$log.return)),
     main = "SPY" , xlab = "Date" ,
     ylab = "Log Return" , col = "red"
)
```




```r
# Question 2
# ==========================

# Q 2.1
sample.skewness <- function(x , adjusted){

  x.m3.origin <- mean(x^3)# 3rd sample moment about the origin
```

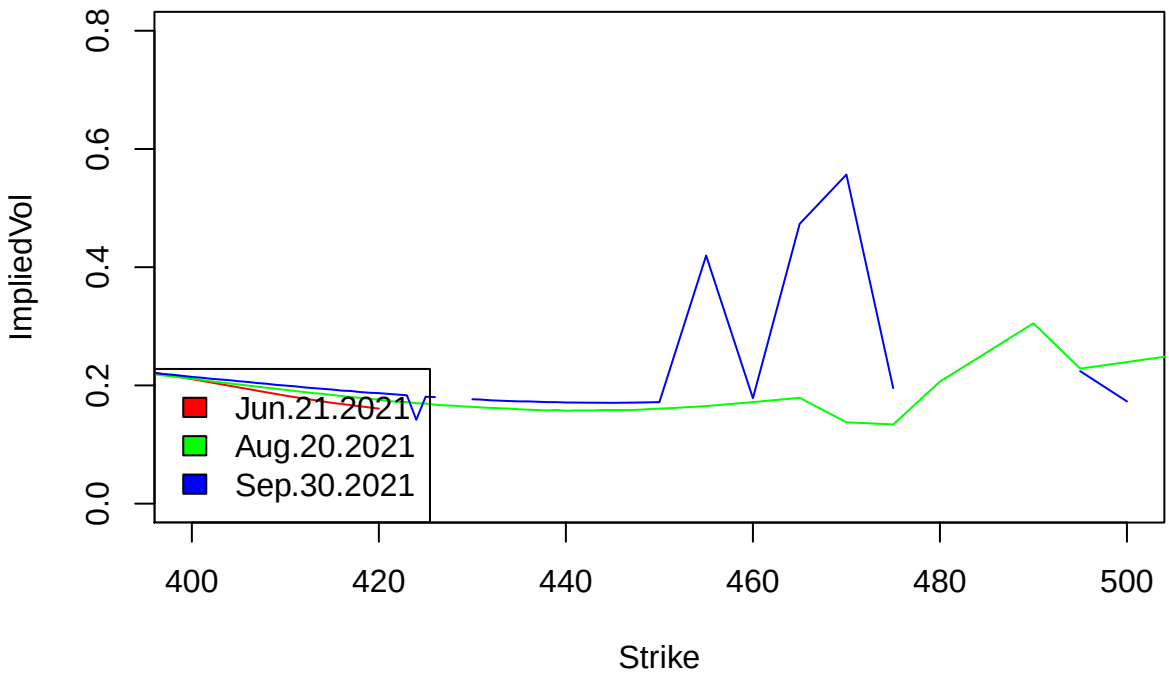
```
  x.m2.origin <- mean(x^2)
  x.sm3 <- x.m3.origin /  (x.m2.origin)^(3/2)

  n <- length(x)

  if(adjusted == TRUE){
    coff <- sqrt(n * (n-1)) / (n-2)
    x.sm3.adj <- coff * x.sm3
    return(x.sm3.adj)

  }else{
    return(x.sm3)
  }

}

sample.kurtosis <- function(x , adjusted){

  x.m4.origin <- mean(x^4)# 4th sample moment about the origin
  x.m2.origin <- mean(x^2)
  x.sm4 <- x.m4.origin /  (x.m2.origin)^(4/2)
  n <- length(x)

  if(adjusted == TRUE){
    coff <- (n-1) / ((n-2) * (n-3))
    x.sm4.adj <- coff * ((n+1)*x.sm4 - 3*(n-1)) + 3
    return(x.sm4.adj)
  }else{
    return(x.sm4)
  }
}

spy.skewness <- sample.skewness(SPY.log.return , FALSE)
spy.skewness.adj <- sample.skewness(SPY.log.return , TRUE)

spy.kurtosis <- sample.kurtosis(SPY.log.return , FALSE)
spy.kurtosis.adj <- sample.kurtosis(SPY.log.return , TRUE)


# Q 2.2

first_column <- c(spy.skewness , spy.skewness.adj)
second_column <- c(spy.kurtosis , spy.kurtosis.adj)
result <- data.frame(first_column , second_column)

colnames(result) <- c("SPY.skewness" , "SPY.kurtosis")
rownames(result) <- c("Unadjusted" , "Adjusted")
result
```

```
##            SPY.skewness SPY.kurtosis
## Unadjusted   -0.9259976     17.26423
## Adjusted     -0.9283239     17.39410
```

```r
# Question 3
# ========================================


# Q 3.1

SPY <- getOptionChain("SPY" , NULL)

# Q 3.2

for(i in 1:length(SPY)){
  SPY[[i]]$calls$Price <- 0.5*(SPY[[i]]$calls$Bid + SPY[[i]]$calls$Ask)
  SPY[[i]]$puts$Price <- 0.5 * (SPY[[i]]$puts$Bid + SPY[[i]]$puts$Ask)
}

# Q 3.3

bisection.new <- function(f, a, b, tol = 0.001, N.max = 100){
  f.a <- f(a)
  f.b <- f(b)
  if(is.na(f.a*f.b) || f.a*f.b > 0){
    return(NA)
  }else if(f.a == 0){
    return(a)
  }else if(f.b == 0){
    return(b)
  }
  for(n in 1:N.max){
    c <- (a+b)/2
    f.c <- f(c)
    if(f.c == 0 || abs(b - a) < tol){
      break
    }
    if(f.a*f.c < 0){
      b <- c
      f.b <- f.c
    }else{
      a <- c
      f.a <- f.c
    }
  }
  return(c)
}


bs.call <- function(S0, K, T1, sigma, r , type){

  d1 <- (log(S0/K) + (r+0.5*sigma^2)*T1)/(sigma*sqrt(T1))
  d2 <- d1 - sigma*sqrt(T1)

  if(type == "call"){
    return(S0*pnorm(d1) - exp(-r*T1)*K*pnorm(d2))
  }
```

```
  return(-S0*pnorm(-d1) + exp(-r*T1)*K*pnorm(-d2)) # put option
}

implied.vol <- function(S0, K, T1, r, price , type){
   price.diff <- function(sigma)bs.call(S0, K, T1, sigma, r , type) - price
   return(bisection.new(price.diff, 0.01, 5))
}

SPY.S0 <- getQuote("SPY")$Last
r <- 0.07 * 0.01
SPY.expiration <- names(SPY)# all expiration dates
T.vec <- (as.Date(SPY.expiration,"%b.%d.%Y")-Sys.Date())/365# calendar day
T.vec <- as.numeric(T.vec)# all time to maturities

for(i in 1:length(SPY)){

  for(j in 1:nrow(SPY[[i]]$calls)){
    SPY[[i]]$calls$ImpliedVol[j] <-
      implied.vol(SPY.S0,
                  SPY[[i]]$calls$Strike[j],
                  T.vec[i],
                  r,
                  SPY[[i]]$calls$Price[j],
                  "call")
  }
  SPY[[i]]$calls <-
    SPY[[i]]$calls[c("Bid", "Ask", "Strike","Price","ImpliedVol")]

  for(j in 1:nrow(SPY[[i]]$puts)){
    SPY[[i]]$puts$ImpliedVol[j] <-
      implied.vol(SPY.S0,
                  SPY[[i]]$puts$Strike[j],
                  T.vec[i],
                  r,
                  SPY[[i]]$puts$Price[j],
                  "put")
  }
  SPY[[i]]$puts <-
    SPY[[i]]$puts[c("Bid", "Ask", "Strike","Price","ImpliedVol")]
}

# Q 3.4

# Initializing figure manually
plot(NA, xlim = c(400,500), ylim = c(0,0.8), xlab = "Strike",
     ylab = "ImpliedVol") # xlim: range of strike, ylim: range of vol
# Add lines
lines(SPY[[14]]$puts$Strike,
      SPY[[14]]$puts$ImpliedVol,col = "red")
lines(SPY[[19]]$puts$Strike,
      SPY[[19]]$puts$ImpliedVol,col = "green")
lines(SPY[[21]]$puts$Strike,
      SPY[[21]]$puts$ImpliedVol,col = "blue")
```

```
legend("bottomleft", SPY.expiration[c(14,19,21)],
       fill = c("red","green","blue"))
```




```
# Q 3.5

for(i in 1:length(SPY)){
  SPY[[i]]$calls <- SPY[[i]]$calls[c("Strike" , "Bid" , "Ask" , "Price" , "ImpliedVol")]
  SPY[[i]]$puts <- SPY[[i]]$puts[c("Strike" , "Bid" , "Ask" , "Price" , "ImpliedVol")]
}

today <- format(Sys.Date(), "%Y-%m-%d")
Exp <- names(SPY)
Exp <- as.Date(Exp, format = "%b.%d.%Y") # convert to date object
Exp <- format(Exp, "%Y_%m_%d") # convert to chars with certain format


for (i in 1:length(Exp)){
  write.csv(SPY[[i]]$calls, file = paste("SPYdata" , today, "Exp" , Exp[i] , "calls.csv" , sep = ""))
  write.csv(SPY[[i]]$puts, file = paste("SPYdata" , today, "Exp" , Exp[i] , "puts.csv" , sep = ""))
}
```