PSTAT 276 Final Project

Ron Kinel, Rebecca He (176), Lingyu Zhou, Wenjing Li, Samran Khan

June 12, 2020

Problem 1

```
StockVol <- function(histoPrice){
  R <- array(0 , length(histoPrice)-1)
  for (i in 1:(length(R))){
    R[i] <- log(histoPrice[i+1]/histoPrice[i])
  }
  Daily_vol <- sqrt(var(R))
  Annual_Vol <- Daily_vol*sqrt(252)
  return(Annual_Vol)
}
```

Problem 2

```
#n: number of paths
#sigma: volatilty/variance of stock, not sd
#S0: current stock price
#t: terminal time in yearly unit
#np: number of periods
#r: interest rate
#delta: continuous dividend yield of the stock
StockPath <- function(n, sigma, S0, t, np=t, r, delta=0){
  S <- array(0, dim=c(n,np+1))
  S[,1] <- S0
  Z <- matrix( rnorm(n*(np)), n, (np))
  h <- t/np
  #h: length of each period = t/np

  for(i in 1:n){
    for (j in 1:np){
      S[i,j+1] <- S[i,j]*exp((r-delta-((sigma^2)/2))*h + sqrt(h*(sigma^2))*Z[i,j])
    }
  }

  mypath <- S
  return(mypath)
}
```

Problem 3

```
EurOptPrice <- function(mypath, t, r, k){
  n <- nrow(mypath)
  np <- ncol(mypath) - 1
  disc_payoff <- array(0, dim=c(n,1)) #discounted payoff vector
```

```
  for (i in 1:n){
    disc_payoff[i,1] <- exp(-r*t)*pmax(k-mypath[i,(np+1)] ,0)
  }

  option_price <- mean(disc_payoff)
  option_var <- var(disc_payoff)

  return(list( option_price,disc_payoff, option_var))
}
```

Problem 4

```
AmeOptPrice <- function(C, t, r, k = C[1,1]){
  #t is time in years until maturity -- needs to match t in stock path function?
  degree0 <- function (x) rep(1,length(x))
  degree1 <- function (x) x
  degree2 <- function (x) x^2
  model <- list(degree0,degree1,degree2)
  S0 <- C[1,1]
  Paths <- nrow(C)
  Steps <- ncol(C)-1
  dt <- t/Steps #time interval for each step
  df <- exp(-r*dt*(1:Steps)) #all discounting factors
  num_model <-length(model) #number of functions used in model
  C <- C[,-1] #first column is simply S0, so remove it
  Value_t <- pmax(k - C[,Steps], 0) #value at time t
  exercise_time <- Steps*rep(1,Paths)
  discounted_payoff <- c(rep(0,20))
  for (i in seq(from=Steps-1, to=1, by=-1)){ #make comparisons starting from the last two column
    Payoff_Value <- pmax(k - C[,i], 0)
    gain <- which(Payoff_Value > 0) #find rows with positive gains
   # discounted_payoff[-gain,i] <- 0
    paths_gain <- C[gain,i] #generate a matrix with current row and column
    regress_Matrix <- matrix(nrow=length(paths_gain), ncol= num_model)
    for (j in 1:num_model) {
      regress_Matrix[,j] <- sapply(paths_gain,model[[j]]) #apply regression on current matrix
    }
    temp <- Value_t[gain]*df[exercise_time[gain]-i]
    discounted_payoff[gain] <- temp
    regress_model <- lm.fit(regress_Matrix,temp)
    Vector <- as.numeric(regress_model$coefficients)
    Holding_Value <- regress_Matrix%*%Vector
    perform_time <- which(Payoff_Value[gain]>Holding_Value)
    exercise_Paths <- gain[perform_time]
    Value_t[exercise_Paths] <- Payoff_Value[exercise_Paths]
    exercise_time[exercise_Paths] <- i #in i-th col, we exercise
  }
  price <- pmax(pmax(k-S0, 0),mean(Value_t*df[exercise_time]))
  variance <- var(discounted_payoff)
  return(list(price,discounted_payoff,variance))
}
```

Problem 5

```
ContVariate <- function(Unknown_Vec, Known_Vec, Known_Value){
  B <- cov(Unknown_Vec, Known_Vec)/var(Known_Vec)
  Ca <- Unknown_Vec + as.numeric(B)*(Known_Vec - Known_Value)
  return(mean(Ca))
}
```

Problem 6

Choose your favorite underlying stock and a corresponding put option that expires in 1 year. Use 1 year libor yield as the interest rate. You also need to compute the continuous dividend yield. Apply the ContVariate function to the American put option price you computed. Do the analysis as if you want to convince your manager that this is the right way to estimate the price of American style option:

We chose Microsoft(MSFT) to do our analysis on. We first import the last year of data for Microsoft. There are 252 trading days between 06/13/19 and 06/11/20 which are the start and end dates of our data.

According to nasdaq.com, MSFT Dividend History in this period is as follows: 05/20/2020: $0.51 02/19/2020: $0.51 11/20/2019: $0.51 08/14/2019: $0.46

According to macrotrends.net, the current 1 year LIBOR rate as of June 04, 2020 is 0.63%. Although it is better to use advanced intrest rate modeling, the 1 year LIBOR is a global "benchmark" for short-term interest and so we will use it instead.

Below we input MSFT last year stock information and extract the closing prices. We will use the function from problem 1 to compute MSFT's annual volatility. We also assign the 1-year LIBOR rate and compute the continuous dividend yield using the Dividend History above. All Data was extracted from Yahoo! Finance.

```
#Input Prices, Spreadsheet included in pproject submission
MSFT <- read.csv(file = 'C:/Users/ron12/OneDrive/Desktop/PSTAT 276/MSFT.csv',
                 header = T, sep = ",")

MSFT_Price <- MSFT$Close

LIBOR <- 0.0063

MSFT_Vol <- StockVol(MSFT_Price)

MSFT_Div <- log((1 + (0.51/MSFT$Close[MSFT$Date == "5/20/2020"]))*(1 + (0.51/MSFT$Close[MSFT$Date == "2/
  (1 + (0.51/MSFT$Close[MSFT$Date == "11/20/2019"]))*(1 + (0.46/MSFT$Close[MSFT$Date == "8/14/2019"])))

cat("The annual volatility for MSFT is:", MSFT_Vol, "\n\n")
```

```
## The annual volatility for MSFT is: 0.4031884
```

```
cat("The continuous dividend yield is:", MSFT_Div)
```

```
## The continuous dividend yield is: 0.01229299
```

Next, we will simulate the next year's MSFT prices by weeks using our StockPath function from Problem 2. This simulation is done using standard Monte Carlo method. For more information, see the description below Problem 2 chunk.

We use weeks because it splits a year into a large number of time periods but its not as computationally heavy as using days. Additionally, it is common to see 1-year stock analysis done in weeks. We will use 100

simulations. This is not a very large amount but should still be sufficient for our analysis while remaining computationally quick.

```
set.seed(34)

n = 20
T = 1
np = 52
r <- LIBOR

MSFT_Path <- StockPath(n, MSFT_Vol , MSFT_Price[length(MSFT_Price)] , T, np, r, MSFT_Div)
```

For our analysis, we will use the 1-year Put: MSFT|20210618|190.00P with Last = 27.25, Bid = 24.1, and Ask = 28.15.
This put expires in almost exactly a year and should be a good comparison point for our analysis.

First, we will use the American Put function from problem 4 to find the Monte-Carlo Machine-Learning Adjusted price. This function... For more information on how the function works, see the explanation following Problem 4 chunk above.

```
K <- 190

Simulated_MSFT_Prices <- as.matrix(MSFT_Path)

MSFT_put <- AmeOptPrice(Simulated_MSFT_Prices ,T ,r, K)

MSFT_put[[1]]
```

```
## [1] 37.30819
```

The result above preforms...

Next, we will use the Black-Scholes models and our European Put Option Function from problem 4 to incorporate in the control variates function from Problem 5 with our result from the previous part. The Control Variates method is a variance reduction technique... For more information about the Control Variate technique read the the discription following the chunk in Problem 5.

We obtain the Black-Scholes price using the bsput function from the 'derivmkts' library and the payoffs from our simulated payoffs using the European Option function from Problem 3. For more information on how this funtion works, see the explanation following Problem 3 chunk.

```
library(derivmkts)
```

```
## Warning: package 'derivmkts' was built under R version 3.5.3
```

```
MSFT_Euro_True <- bsput(MSFT_Price[length(MSFT_Price)], K, MSFT_Vol, r, T, MSFT_Div)

cat("The true European BS Put price is:" , MSFT_Euro_True, "\n\n")
```

```
## The true European BS Put price is: 32.24696
```

```
MSFT_Euro_Sim <- EurOptPrice(MSFT_Path, T, r, K)

cat("The Simulated Put price is:", MSFT_Euro_Sim[[1]])
```

```
## The Simulated Put price is: 34.16288
```

Using our results above with control variates:

```
Cont_Variate_put <- ContVariate(MSFT_put[[2]], MSFT_Euro_Sim[[2]], MSFT_Euro_True)

Cont_Variate_put
```

```
## [1] 39.14475
```

This result...