Monte Carlo Simulation in Option Pricing

Niraj Sardar

Introduction

Pricing financial options is amongst the most important and challenging problems in the modern financial industry. Except in the simplest cases, the prices of options do not have a simple closed form solution and efficient computational methods are needed to determine them. Monte Carlo methods have increasingly become a popular computational tool to price complex financial options, especially when the underlying space of assets has a large dimensionality,as the performance of other numerical methods typically suffer from the 'curse of dimensionality'.

Monte Carlo Simulation

Monte Carlo Option Price is a method often used in Mathematical finance to calculate the value of an option with multiple sources of uncertainties and random features, such as changing interest rates, stock prices or exchange rates, etc.This method is called Monte Carlo simulation, naming after the city of Monte Carlo, which is noted for its casinos.

The basis of a Monte Carlo simulation is that the probability of varying outcomes cannot be determined because of random variable interference. Therefore, a Monte Carlo simulation focuses on constantly repeating random samples to achieve certain results. In this assignment we will be running 1000 iterations

The below function is used to generate random stock price with the stock price as 20 , maturity as 0.25, steps as 20 , deviation of 0.4 , interest rate of 3 percent and 1000 iterations .

Generate Stock Price

The first function will return matrix of stock prices with rows forming price path , mean ending price, mean max price and mean min price.

```r
generateStockPrice <- function( stockPrice = 20, maturity = 0.25 , steps = 20 , sigma = 0.4  , interestRate = 0.03 , iterati
ons =1000 ,
                                seed = 12) {

S0 <- stockPrice
r <- interestRate
sig <- sigma
T <- maturity
m <- steps
t <-  T/m

n <- iterations
nvars <- m*n
set.seed(seed)
e <- as.vector(rnorm(n=nvars, m=0, sd=1))
E <- matrix(e,nrow = n,ncol = m)
head(E, n=2)


fac <- exp((r-.5*sig^2)*t+sig*sqrt(t)*E)
head(fac, n=1)

fac1 <- t(apply(fac,1,cumprod))     # cumulate returns
head(fac1, n=1)

St <- fac1*S0                       # multiply with price
head(St, n=1)

MeanSt <- apply(St,1,mean)

 endPrice <- St[,ncol(St)]
 endPriceMean <- mean(endPrice)

 maxPrice <- apply(St, 1, max)
 maxMean <- mean(maxPrice)

 minPrice <- apply(St, 1, min)
 minMean <- mean(minPrice)


 finalResult <- list(St, endPriceMean, maxMean, minMean) #returning multiple arguments
 return(finalResult)

}
```

The stock price has been generated and the above function returns the value of stock price path along with end price mean , maximum mean , minimum mean. The values are listed down below .

```r
result <- generateStockPrice()

# The first two price path :

head(result[[1]],2)
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
## [1,] 18.70693 16.70236 17.43784 16.95179 19.34127 18.78898 18.94030 18.97158
## [2,] 21.44819 21.93510 23.12699 22.27332 22.35005 20.57222 19.18193 20.07497
##          [,9]    [,10]    [,11]    [,12]    [,13]    [,14]    [,15]    [,16]
## [1,] 20.53006 21.30370 19.69631 18.01130 16.09071 16.65925 15.79453 15.67380
## [2,] 20.68644 20.83912 20.27498 18.71168 17.89366 15.90486 16.30661 15.50645
##         [,17]    [,18]    [,19]    [,20]
## [1,] 14.92605 14.56841 13.32745 14.19706
## [2,] 16.94518 16.98628 16.41634 16.42661
```

```r
# End Price Mean :

print(result[[2]])
```

```
## [1] 20.20476
```

```
# Maximum Mean :

print(result[[3]])
```

```
## [1] 22.8527
```

```
# Minimum Mean :

print(result[[4]])
```

```
## [1] 17.55858
```

Option Price Estimation

In this function we are required to calculate the option price estimation for different type of exotic options . We will be using the output from the first function inside this function and do some calculations in order to get the desired option value .

The 5 different type of exotic options used are :

1. Float Lookback Option : A floating strike lookback option is an option with its strike price set equal to the optimal value thatis achieved by the underlying asset over the option´s life. In the case of a call, that optimal value is the lowest value achieved by the underlier during the life of the option, so it pays off the difference between the final value of the underlying asset and that lowest value. In the case of a put, the option pays off the difference between the highest value achieved and the value of the underlying asset at expiration.

2. Fixed Lookback Option : A fixed strike lookback option has a strike price set in advance. Its exercise depends on the optimal value achieved by the underlying asset during the life of the option. In the case of a call, the optimal value is the highest value the underlier achieves, so the call pays off the difference between that value and the strike price, if positive, and zero otherwise. In the case of a put, the optimal value is the lowest value achieved by the underlying asset and the put pays off the difference between the strike price and that lowest value, if positive, and zero otherwise.

3. Asian Arithmetic Option : An arithmetic average strike option is a specific Asian option. Its payoff depends on a strike price which is set equal to the arithmetically made up mean of the asset price during the life of the option.

4. Asian Geometric : The geometric average rate option is a specific Asian option and therefore depends on an average price of the underlier. Here this average is calculated geometrically. The sampling is carried out between 0 and t, for which period you have to enter the arithmetic average stock price.

5. Asset ot Nothing : An asset-or-nothing call is a type of digital option whose payout is fixed after the underlying asset exceeds the predetermined threshold or strike price. The payout depends only on whether or not the underlying asset closes above the strike price—in the money (ITM)—at the expiration date.

```r
optionPriceEstimation <- function( stockPrice = 20, maturity = 0.25 , steps = 20 , sigma = 0.4  , interestRate = 0.03 , iter
ations =1000 ,
                                seed = 12 , callorput = "call" , strike = 20 , exotic )
{

S0 <- stockPrice
r <- interestRate
sig <- sigma
T <- maturity
m <- steps
t <-  T/m

getStPrice <- generateStockPrice(S0, T, m , sig, r , iterations, seed)

stockPrice  <- getStPrice[[1]] #extracting stock price matrix from function 1 return values.
endPrice <- stockPrice[,ncol(stockPrice)]
maxPrice <- apply(stockPrice, 1, max)
minPrice <- apply(stockPrice, 1, min)
t <- T/m #time interval per step

fac <- exp(-r*T)

# Float Lookback Option (floating look back where the payoff depends on the difference between the last price and the max/mi
n price; there is no strike price)

    if (exotic == "floatlookback") {
      if (callorput == "call") {
        payoff1 <- ifelse(endPrice > minPrice, endPrice - minPrice, 0)
        floatlookback <- fac * payoff1
        optionValue <- mean(floatlookback)}

      else if (callorput == "put") {
        payoff1 <- ifelse(endPrice < maxPrice, maxPrice - endPrice, 0)
        floatlookback <- fac * payoff1
        optionValue <- mean(floatlookback)}
    }
# Fixed Lookback Option (fixed look back where the payoff depends on the difference between the max/min price and the exerci
se price)
    if (exotic == "fixedlookback") {
      if (callorput == "call") {
        payoff2 <- ifelse(maxPrice > strike, maxPrice - strike, 0)
        fixedlookback <- fac * payoff2
        optionValue <- mean(fixedlookback)}

      else if (callorput == "put") {
        payoff2 <- ifelse(strike > minPrice, strike - minPrice, 0)
        fixedlookback <- fac * payoff2
        optionValue <- mean(fixedlookback)}
    }

# Asian arithmetic option
    if (exotic == "asianarithmetic") {
      meanStkPrice <- apply(stockPrice, 1, mean)
      if (callorput == "call") {
        payoff3 <- ifelse(meanStkPrice > strike, meanStkPrice - strike, 0)
        AsianArithmetic <- fac * payoff3
        optionValue <- mean(AsianArithmetic)}

      else if (callorput == "put") {
        payoff3 <- ifelse(strike > meanStkPrice, strike - meanStkPrice, 0)
        AsianArithmetic <- fac * payoff3
        optionValue <- mean(AsianArithmetic)}
    }

# Asian geometric option (the geometric mean is used in the payoff function)
    if (exotic == "asiangeometric") {
      geoMean <- function(x, na.rm=TRUE){
        exp(sum(log(x[x > 0]), na.rm=na.rm) / length(x))}
      geoMeanStk <- apply(stockPrice, 1, geoMean)
      if (callorput == "call") {
        payoff4 <- ifelse(geoMeanStk > strike, geoMeanStk - strike, 0)
        AsianGeo <- fac * payoff4
        optionValue <- mean(AsianGeo)}

      else if (callorput == "put") {
        fac4 <- ifelse(strike > geoMeanStk, strike - geoMeanStk, 0)
        AsianGeo <- fac * fac4
        optionValue <- mean(AsianGeo)}
    }

# Asset or nothing option

    if (exotic == "assetornothing") {
      if (callorput == "call") {
        d1 <- (log(S0/strike) + (r + sigma^2/2)*T-t) / (sigma*sqrt(T-t))
        optionValue <- S0 * pnorm(d1)}

      else if (callorput == "put") {
        d1 <- (log(S0/strike) + (r + sigma^2/2)*T-t) / (sigma*sqrt(T-t))
        optionValue <- S0 * pnorm(-d1)}
    }


returnlist2 <- list(S0, T, steps, sigma, r , iterations, seed, callorput, strike, exotic, optionValue)
return(returnlist2)
}
```

The opiton price estimation function will require the below input parameters. We will exectue this to get the desired output.

```
callorput <- list("floatlookback", "fixedlookback", "asianarithmetic", "asiangeometric", "assetornothing")
exotic <- list("call","put")
finalResult <- data.frame("Stock_Price", "Time to Maturity", "Steps", "Standard Deviation", "Interest Rate", "Iterations",
"Seed_Value", "Option", "Strike Price", "Category", "Option Price")
for (i in callorput){
  for(j in exotic){
    output <- optionPriceEstimation(, , , , , , , j, , i)
    finalResult[nrow(finalResult) + 1,] <- output
  }
}
names(finalResult) <- NULL

# Final option value with differenct exotic values
head(finalResult)
```

```
##
## 1 Stock_Price Time to Maturity Steps Standard Deviation Interest Rate
## 2          20             0.25    20                0.4          0.03
## 3          20             0.25    20                0.4          0.03
## 4          20             0.25    20                0.4          0.03
## 5          20             0.25    20                0.4          0.03
## 6          20             0.25    20                0.4          0.03
##
## 1 Iterations Seed_Value Option Strike Price        Category     Option Price
## 2       1000         12   call           20   floatlookback 2.62640506381602
## 3       1000         12    put           20   floatlookback 2.62815962362561
## 4       1000         12   call           20   fixedlookback 2.91661212073614
## 5       1000         12    put           20   fixedlookback 2.49285184917642
## 6       1000         12   call           20 asianarithmetic 1.01511826435883
```

Limitations and Improvements:

1. Monte Carlo financial-planning software is based on a normal distribution for returns, with inputs including expected asset-class returns, standard deviations and correlations. Outcomes at the tails may not be precise, because we cannot be certain about both the accuracy of our inputs and the underlying distribution for the returns. Those probabilities of success or failure for a plan calculated through Monte Carlo simulation are only estimates.

2)A problem with Monte Carlo tools that is that they can often paint an unrealistic picture of returns.For example, an average expected return for cash of 2% is unrealistic (since the return on cash today is closer to 0%).

3)It is fairly complex and can only be carried out using specially designed software that may be expensive.

4)If bad distributions and parameters are given as input, the output will be garbage.

5. One method to incorporate information about today’s return environment (e.g., the yield on bonds) is to use what’s called an autoregressive model. An autoregressive model captures a relationship between the previous value and the next value. This is important when projecting things like bond yields versus stocks, because bond returns (and interest rates) are auto-correlated. However there are other metrics (e.g., dividend yields and Shiller’s Cyclically Adjusted Price-to-Earnings ratio) that have historically been useful when projecting the future return on equities.
6. There are alternative ways to incorporate returns into a Monte Carlo tool. One method is to use historical series of returns (e.g., the returns for the U.S. stock market over a long time period). One significant problem with this approach is that is assumes what has happened historically will happen in the future. It implicitly incorporates the recurrence of historical events and allows for only a limited amount of data. Two ways to make using historical series of returns more attractive are to reduce returns to more conservative levels and/or use international data, when available.

Conclusion :

Created two functions which used Monte Carlo Simulation to generate stock prices and do option price estimation for different exotic options such as lookback, asian geometric, asian arithmetic and asset or nothing ,and how these value will change if we either call or put have been calculated. The use cases of this model have been discussed along with the limitations. This model can be best build if the user has clear understanding of the input parameters and try to understand the limitations of this model to build proper simulation model. Hence, there are very few limitations when running a Monte Carlo simulation, although you may be limited based on the tools you have available or your ability to build such tools yourself.