Monte Carlo Option Pricing

Ishan Ambike

12/02/2021

Introduction:

Options are financial derivatives that give buyers the right, but not the obligation, to buy or sell an underlying asset at an agreed-upon price and date. Call options and put options form the basis for a wide range of option strategies designed for hedging, income, or speculation. Although there are many opportunities to profit with options, investors should carefully weigh the risks. In this assignment, we will be generating price paths for options using the Weiner Process, and then using Monte Carlo simulation we will be generating option values for a few exotic options.

Function 1 - Stock Price Generation:

The use of function 1 is to generate stock prices. To generate the stock prices, I am using the below inputs to the function. Default values are mentioned in parentheses.

- Stock price - S (20)
- Maturity - T (0.25)
- Steps - m (20)
- Sigma - sig (0.4)
- Interest rate - r (0.03)
- Iterations - n (1000)
- Seed (12)

I am then generating time intervals per step(t) by dividing T by n. By multiplying m and n we get the number of variables(nvars). I have then generated random numbers equal to nvars by using rnorm() function and then stored them in the matrix E. I am using the Weiner process to generate a sequence of values.

Weiner Process : $S(t+\delta t)=S(t)\exp[(r-\sigma^2/2)\delta t+\sigma\epsilon\text{sqrt}(\delta t)]$

The factor is calculated using Weiner Process. I am then calculating the cumulative product of each row and the result is multiplied by stock price(S) to create price path. Price paths are then stored in the matrix St. I have then calculated ending price, max price, and min price for each price path. Then mean is calculated of all the three prices to get the mean values. I have created a list to return all the required values.

Function 1 code:

```
options(warn = -1)
library(psych)
```

```
fun1 <- function(S = 20, T = .25, m = 20, sig = .4, r = .03, n = 1000, seed =
12)
{
  t <- T/m
  set.seed(seed)
  nvars <- m * n

  e <- as.vector(rnorm(n=nvars, m=0, sd=1))
  E <- matrix(e,nrow = n,ncol = m)
  fac <- exp((r-.5*sig^2)*t+sig*sqrt(t)*E)
  fac1 <- t(apply(fac,1,cumprod))
  St <- fac1 * S

  endingPrice <- St[,ncol(St)]
  meanEndingPrice <- mean(endingPrice)

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

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

  results <- list(St, meanEndingPrice, meanMaxPrice, meanMinPrice)
  return(results)
}
```

Calling Function fun1 with default values and displaying results:

```
returnValues <- fun1()
St <- returnValues[[1]]
meanEndingPrice <- returnValues[[2]]
meanMaxPrice <- returnValues[[3]]
meanMinPrice <- returnValues[[4]]

cat("First 2 price paths:\n")

## First 2 price paths:

St[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

cat("\n Mean Ending Price: ",meanEndingPrice)

##
##  Mean Ending Price:  20.20476

cat("\n Mean Max Price: ",meanMaxPrice)

##
##  Mean Max Price:  22.8527

cat("\n Mean Min Price: ",meanMinPrice)

##
##  Mean Min Price:  17.55858
```

Hence we get the results as above for the specified inputs.

Function 2 - Option Price Estimation:

This function uses all the same inputs as fun1 in addition to these three:

- callorput ('call')
- Strike price - X (20)
- exotic

The argument 'exotic' is a string mentioning the type of exotic option. No default value is set and it should be mentioned explicitly during the function call.

For this assignment we have to implement the following types of options:

1. Float Look Back: The option's strike price is floating and determined at maturity. The floating strike is the optimal value of the underlying asset's price during the option life. The payoff is the maximum difference between the market asset's price at maturity and the floating strike.

Payoff formula:

Call: max(Send - Smin, 0) | Put: max(Smax - Send, 0)

2. Fixed Look Back: This option's strike price is fixed. This option is not exercised at the price at maturity. The payoff is the maximum difference between the optimal underlying asset price and the strike.

Payoff formula:

Call: max(Smax - X, 0) | Put: max(X - Smin, 0)

3. Asian Arithmetic: Asian options are path-dependent. One advantage of Asian options is that they reduce the risk of market manipulation of the underlying instrument at maturity. The payoff is the maximum difference between the arithmetic mean and the strike price.

Payoff formula:

Call: max(Smean - X, 0) | Put: max(X - Smean, 0)

4. Asian Geometric: This is similar to the Asian arithmetic option except we are using geometric mean instead of arithmetic mean.

Payoff formula:

Call: max(Sgeo_mean - X, 0) | Put: max(X - Sgeo_mean, 0)

5. Asset or 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. Call options settle with the physical delivery of the underlying asset if the option expires in the money. These options are digital or binary, meaning they pay a predetermined payout or zero.

Payoff formula:

$d1 <- (log(S/X) + (r + sig^2/2) * T\text{-}t) / (sig * sqrt(T\text{-}t))$

Call: $S * e^{-qT} * N(d1)$ | Put: $S * e^{-qT} * N(-d1)$

Function 2 code:

```
fun2 <- function(S = 20, T = .25, m = 20, sig = .4, r = .03, n = 1000, seed =
12,
                 callorput = 'call', X = 20, exotic)
{

  fun1values <- fun1(S, T, m, sig, r, n, seed)

  St2 <- fun1values[[1]]

  t <- T/m

  endingPrice <- St2[,ncol(St2)]
  minPrice <- apply(St2,1,min)
  maxPrice <- apply(St2,1,max)

  PV <- exp(-r*T)
  optionValue <- 0

  if (exotic == 'floatlookback'){

    if(callorput == 'call'){
      payoff <- ifelse(endingPrice > minPrice, endingPrice - minPrice, 0)
```

```
    LCfloat <- PV * payoff
    optionValue <- mean(LCfloat)
  }
  else if(callorput == 'put'){
    payoff <- ifelse(endingPrice < maxPrice, maxPrice - endingPrice, 0)
    LPfloat <- PV * payoff
    optionValue <- mean(LPfloat)
  }
}
else if (exotic == 'fixedlookback'){

  if(callorput == 'call'){
    payoff <- ifelse(maxPrice > X, maxPrice - X, 0)
    LCfix <- PV * payoff
    optionValue <- mean(LCfix)
  }
  else if(callorput == 'put'){
    payoff <- ifelse( X > minPrice, X - minPrice, 0)
    LPfix <- PV * payoff
    optionValue <- mean(LPfix)
  }
}
else if (exotic == 'asianarithmetic'){
 arithmeticMean <- apply(St2,1,mean)

 if(callorput == 'call'){
   payoff <- ifelse(arithmeticMean > X, arithmeticMean - X, 0)
   asianArithmeticCall <- PV * payoff
   optionValue <- mean(asianArithmeticCall)
 }
 else if(callorput == 'put'){
   payoff <- ifelse(X > arithmeticMean , X - arithmeticMean, 0)
   asianArithmeticPut <- PV * payoff
   optionValue <- mean(asianArithmeticPut)
 }
}
else if (exotic == 'asiangeometric'){
  geometricMean <- apply(St2,1,geometric.mean)

  if(callorput == 'call'){
    payoff <- ifelse(geometricMean > X, geometricMean - X, 0)
    asianGeometricCall <- PV * payoff
    optionValue <- mean(asianGeometricCall)
  }
  else if(callorput == 'put'){
    payoff <- ifelse(X > geometricMean, X - geometricMean, 0)
    asianGeometricPut <- PV * payoff
    optionValue <- mean(asianGeometricPut)
  }
}
```

```
  else if (exotic == 'assetornothing'){
    d1 <- (log(S/X) + (r + sig^2/2)*T-t) / (sig*sqrt(T-t))

    if(callorput =='call'){
      optionValue <- S * pnorm(d1)
    }
    else if(callorput == 'put'){
      optionValue <- S * pnorm(-d1)
    }
  }

  retList <- list(optionValue, S, T, m, sig, r, n, seed, callorput, X,
exotic)
  return(retList)
}
```

Calling Function fun2 for all the option types and storing the results in a data frame:

```
exoticList <- c('floatlookback', 'fixedlookback', 'asianarithmetic',
'asiangeometric', 'assetornothing')
mat = matrix(ncol = 11, nrow = 0)
df=data.frame(mat)

# calling fun2() to get call option values
for(i in exoticList){
  op <- fun2(exotic = i)
  df <- rbind(df, op)
}

# calling fun2() to get put option values
for(i in exoticList){
  op <- fun2(exotic = i,callorput='put')
  df <- rbind(df, op)
}
names(df) <- c('Option Value', 'Stock price', 'Maturity', 'Steps', 'Sigma',
'Int rate', 'Iterations', 'Seed', 'callorput', 'Strike price', 'exotic')

df <-df[order(df$exotic),]
df
```

```
##    Option Value Stock price Maturity Steps Sigma Int rate Iterations Seed
## 3     1.0151183          20     0.25    20   0.4     0.03       1000   12
## 8     0.9281440          20     0.25    20   0.4     0.03       1000   12
## 4     0.9794763          20     0.25    20   0.4     0.03       1000   12
## 9     0.9580069          20     0.25    20   0.4     0.03       1000   12
## 5    10.6133539          20     0.25    20   0.4     0.03       1000   12
## 10    9.3866461          20     0.25    20   0.4     0.03       1000   12
## 2     2.9166121          20     0.25    20   0.4     0.03       1000   12
## 7     2.4928518          20     0.25    20   0.4     0.03       1000   12
## 1     2.6264051          20     0.25    20   0.4     0.03       1000   12
## 6     2.6281596          20     0.25    20   0.4     0.03       1000   12
```

```
##    callorput Strike price          exotic
## 3       call           20 asianarithmetic
## 8        put           20 asianarithmetic
## 4       call           20  asiangeometric
## 9        put           20  asiangeometric
## 5       call           20  assetornothing
## 10       put           20  assetornothing
## 2       call           20   fixedlookback
## 7        put           20   fixedlookback
## 1       call           20   floatlookback
## 6        put           20   floatlookback
```

The table above gives values for all the types of option values and the inputs used. Default inputs were used to generate values for all the options.

Use cases:

- We can generate multiple scenarios and analyze them.
- We can use this simulation to make the decision-making process easier.
- Can be used to understand the impact of risk and uncertainty in prediction and forecasting models.
- Investors can get to know the risk factor of their investment portfolio.
- Financial planners use this simulation to determine optimal investment strategies for their clients.

Limitations:

- We only get statistical numbers and not the final decisive output.
- Complex processes and prone to errors which can give wrong results.
- Pseudo-random numbers are drawn from a normal distribution, hence, usually, the uniform normal random variables used in the model, will not be able to capture the abnormal risk events.
- Requires excessive computation.

Conclusion:

In this assignment, we understood and implemented the Monte Carlo simulation. First using the default inputs, we generated price paths and summary measures like mean ending price, mean max price, and mean min price. We learned about five different exotic options and generated price paths and calculated the option value for each type. We also discussed the use cases and the limitations of the model.

References:

- https://en.wikipedia.org/wiki/Lookback_option
- https://en.wikipedia.org/wiki/Asian_option
- https://quant.stackexchange.com/questions/15489/derivation-of-the-formulas-for-the-values-of-european-asset-or-nothing-and-cash