Using R: European Option Pricing Using Monte Carlo Simulation

Clifford S. Ang, CFA
February 3, 2015

In this article, I demonstrate how to estimate the price of a European call option using Monte Carlo (MC) simulation. The point of this example is to show how to price using MC simulation something we already know how to price using the Black-Scholes options pricing model (OPM). This example allows us to focus on a new technique rather than understanding a new technique *and* new concept.

In the Black-Scholes OPM, once we have the required inputs, the rest of the calculation is a matter of plugging those numbers into a formula. However, the Black-Scholes OPM does not allow us much flexibility in terms of modeling different types of options. This is where the benefit of pricing options using MC simulation comes in.

For our example, we will consider pricing a hypothetical Amazon.com option with a strike price of $300 as of the end of 2013 and 2.5 years to maturity. On 12/30/13, the Amazon.com stock price was $398.79 and the annualized standard deviation of its daily returns was approximately 32%. As of that date, the annualized risk-free rate was 1%.

```
> stock=398.79
> sigma=0.32
> strike=300
> TTM=2.5
> rf=0.01
```

Above is the output we would see from the `R Console` if we constructed the variables for `stock`, `sigma`, `strike`, `TTM`, and `rf`. Note that to input the code above into the `R Editor`, which I suggest you do, we exclude the `>` that is part of the `R Console` output.

Next, we determine how many paths we want to simulate. I suggest having a minimum of 10,000 paths, but 100,000 is typically more than sufficient. The time it takes to complete 100,000 runs for this particular program is *de minimis* compared to 10,000 runs. However, note that in more complex situations, the difference can be substantial.

```
> num.sim<-100000
```

We now setup the parameters for our model of stock price behavior. For our current purpose, we use a normal distribution. The variable `TTM.price` calculates the last price in each sample path based on the `R` and `SD` values. The output below shows that `R` is equal to -0.103 and `SD` is equal to 0.506. Note that the output is in the line that begins with `[1]`. I do not output the value for `num.sim` because that would generate output with 100,000 observations. This is because using `num.sim` with the `rnorm` function generates 100,000 sample paths in this example. The last two arguments in `rnorm` are for the mean and standard deviation, which is 0 and 1, respectively, in our example (i.e., we generate a standard normal random variable).

```
> R<-(rf-0.5*sigma^2)*TTM
> R
[1] -0.103
> SD<-sigma*sqrt(TTM)
> SD
[1] 0.5059644
> TTM.price<-stock*exp(R+SD*rnorm(num.sim,0,1))
```

We can now calculate the value of the call option for each sample path. We know that a call option's intrinsic value at maturity is equal to $\max[0, S_T - K]$, where S_T is the stock price at time to maturity T and K is the strike price of the option. The `pmax` function calculates the maximum of the 0 or the $S_T - K$ for each value at the end of the sample path. We then discount each of the 100,000 values we calculate to the present, which is represented by the variable `PV.call`. The average of the 100,000 values of `PV.call` is the price of the European call option. We arrive at this using the `mean` function. This yields a call option value of $134.27.

```
> TTM.call<-pmax(0,TTM.price-strike)
> PV.call<-TTM.call*(exp(-rf*TTM))
> mean(PV.call)
[1] 134.2697
```

Similarly, we can calculate the value of a put option in the same manner with the only change is in the way we calculate the intrinsic value at maturity. Specifically, the value of a put option is equal to $\max[0, K - S_T]$, where K is the strike price of the option and S_T is the stock price at time to maturity T. This calculation yields a put option value of $28.07.

```
> TTM.put<-pmax(0,strike-TTM.price)
> PV.put<-TTM.put*(exp(-rf*TTM))
> mean(PV.put)
[1] 28.07119
```

We can then compare the call and put option values we just calculated to the Black-Scholes call and put option values for the same option.

```
> d1<-(log(stock/strike)+(rf+0.5*sigma^2)*TTM)/(sigma*sqrt(TTM))
> d2<-d1-(sigma*sqrt(TTM))
>
> BS.call<-stock*pnorm(d1,mean=0,sd=1)-strike*exp(-rf*TTM)*pnorm(d2,mean=0,sd=1)
> BS.call
[1] 134.2938
>
> BS.put<-BS.call-stock+strike*exp(-rf*TTM)
> BS.put
[1] 28.09674
```

As the above output shows, the Black-Scholes call option value is $134.29, which is pretty close to the MC call option value of $134.27. Then, we calculate the value of the put option under Black-Scholes via put-call parity. The Black-Scholes put option value is $28.10, which is also very close to the MC put option value of $28.07.