Math 458 - Financial Mathematics for Actuaries II - Project # 2

Due: Midnight on Monday, November 13, 2023.

Objective: Use R and the numerical implementation of the lognormal model of stock prices to estimate the price of put and call stock options.

Directions:

1. Introduction: This model is a good representation of how simulation within R can be used to provided a probabilistic estimate of the *fair price* of a stock option. In this project you will write an R script that will generate thousands of possible stock prices in the future, use these prices to compute all option payoffs at the end of 1 year, and then evaluate basic descriptive statistics of the generated data to arrive at a box-whisker plot and a good estimate for the fair market option price at time $t = 0$. Finally you will compare your numerical analysis with a theoretical model for the correct option price.

2. Models:

   - Theory: The BSM theory of stock option pricing gives the price C_E of a European call option and the P_E of a European put option with the aforementioned parameters. They are

     $$C_E = N(d_1) S_0 e^{-\delta T} - N(d_2) K e^{-rT}$$

     and

     $$P_E = N(-d_2) K e^{-rT} - N(-d_1) S_0 e^{-\delta T}.$$

   - Simulation: Recall from class the lognormal pricing model for stocks where the price at time T for a stock with price S_0 at time $t = 0$ is given by the random variable

     $$S_T = S_0 e^{\left(r-\delta-\sigma^2/2\right)T+\sigma\sqrt{T}\,Z}. \tag{1}$$

     It follows that the payoff of any derivative based on S_T is also a random variable. Statistical analysis of these payoffs can then be used to estimate the arbitrage-free price of a derivative.

3. Method:

A) Open RStudio and create a new script that starts by loading the libraries "`stringr`" and "`magrittr`" which we will use to format boxplots later in this project.

B) Write an R script that

i) uses a `for` loop and the `append` command to iteratively compute 1000 different stock prices at time $T = 1$ year and, in the same loop, constructs two vectors, one each for the corresponding payoffs of a 1-year European put option and a 1-year European call option. To find each stock price after 1 year, use formula (1) with a risk free rate of 7%, a dividend rate of 2%, a stock volatility of 25%, an initial stock price of \$100, and the command `rnorm(1,0,1)` to generate single normally distributed random numbers Z. Your code should use variable names for everything but Z in (1) so that you only have to change the model parameters once to run your code for a different stock option.

ii) computes the means μ_P and μ_C for the 1000 elements of the separate put and call payoff vectors in i) and then find the estimated option prices using the present value formulae

$$P_E \approx e^{-rT}\mu_P \quad \text{and} \quad C_E \approx e^{-rT}\mu_C;$$

iii) repeats the above process 50 times using an outer `for` loop and the `append` command to create a vector of 50 estimated option prices, each of which is based on 1000 simulated stock prices;

iv) uses the wrapping function

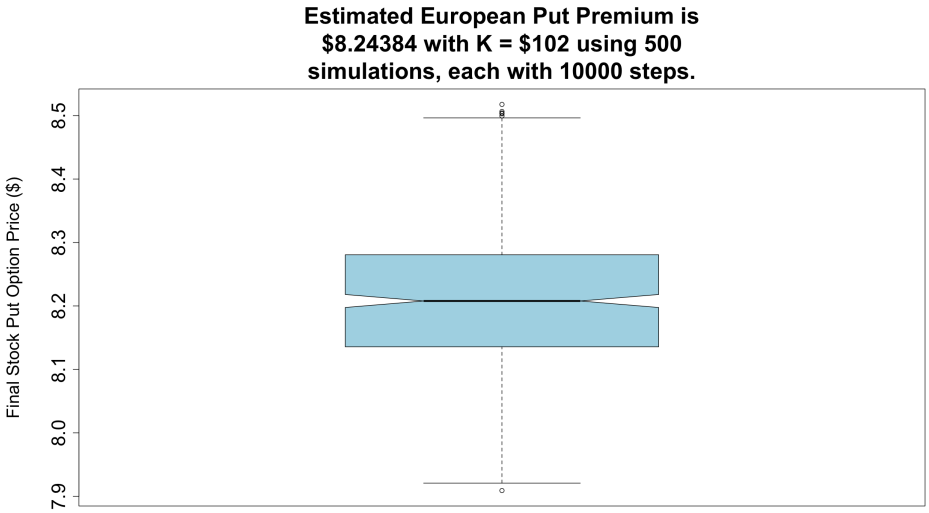
```
wrap.it <- function(x, len)
                    {
sapply(x, function(y) paste(strwrap(y, len),
              collapse ="\n"),
            USE.NAMES = FALSE)
                    }
```

and the boxplot command

```
boxplot(optionvalueestimates, xlab=" ", ylab="final
stock price",col="lightblue",notch=T,main=str_wrap(paste("Mean
European put premium is $", round(optionprice,4), "
with K=$",K, " using ", optionsteps, " estimates, each
with ", stocksteps, " stock price simulations.", sep=""),60))
```

to create a plot of the form shown below for the case of a put option, and a similar plot for the case of a call option. In the above command, `K` is the strike price, `optionsteps` is the number of option payoff

estimates (50 in so far), and `stocksteps` is the number of stock prices (1000 so far).



C) How long does it take your code to run? You can find the runtime of your script by inserting the line `timer<-proc.time()` at the beginning of your code and `proc.time() -timer` at the end of your code. The time under "elapsed" will be the time your code took to run.

D) Rerun your code from parts B) when you compute 500 estimates of the option price using 10000 simulated stock prices, instead of 50 estimates of the option price using 1000 simulated stock prices. How long does it take your code to run compared to the original problem runtime you computed in part C)?

E) Repeat parts B)-D) above using an R script that *does not* use a `for` loop to create a vector of stock prices but instead uses the vector processing power of R. Use the command `rnorm(`N`,0,1)` to generate a vector of $N > 0$ values of Z and the vector capabilities of `exp()` to quickly compute a vector of N simulated stock prices using formula (1). Your code should only include a single `for` loop and one use of the `append` command to create a vector of $M > 0$ estimated option prices, each of which is based on $N > 0$ stock prices.

F) For the final part of this project, use BSM formulas for a put and a call option to compute C_E and P_E exactly. Then compare this theoretical result with the prediction of your simulations from the earlier parts of the project. Are they close? Assuming that BSM is perfect, what is the relative error between your simulated prices and the theoretical prices? Explain.