Option Pricing Model

Mayur

30/10/2019

# INTRODUCTION

Options are derivative contracts which give the bearer of that contract the right, but not obligation, to sell or buy the underlying asset. The right can be exercised either at expiration date of the contract (European Option) or at any time till the expiration date of the contract (American Option). A call option gives the bearer of the contract the right to buy, and a put a option gives the bearer of the contract the right sell, the underlying asset. The option contracts are either traded in an exchange in standardized form or they are traded over-the-counter in customized form. When the underlying asset is a stock then the options are known as stock options. Options can either be used to earn money by speculating about the future direction of the value of underlying asset based on market opinion or they can be used to limit losses through hedging. Usually the bearer of the option contract has to pay a premium to the writer of the option contract. The premium is usually paid at the time the contract is signed. The premium is also known as the value of the option or the option price.

The value of option consist of two parts: intrinsic value and extrinsic value. The intrinsic value is the amount of payoff earned if the option is exercised and the underlying asset is disposed off immediately at the market price. Extrinsic value is intrinsic value plus a premium based on the time to expiration and the volatility of the underlying asset. While the value of option decreases with decreasing time to maturity, the option value increases with increasing volatility. The value of option can be calculated using mathematical models which take into account both the intrinsic as well as extrinsic value of an option.

Our purpose in this assignment is to estimate option value using two such mathematical models for non-dividend paying european options: the Black-Scholes-Merton model and the Binomial model. We will then compare these estimated values with actual market price of the option. We will also examine the Put-Call Parity relationship in all the three cases.

# DATA ANALYSIS : SOURCES AND METHODOLOGY

For the purpose of our analysis we will do the case study of Netflix stocks listed on NASDAQ and its widely traded 3 month stock options. We will use an option chain with an expiration date of “08-11-2019” as our unit of analysis. All the data analysis, modelling, and visualization will be done using “R software”. Specifically we will use the “quantmod” package which allows us to import option chain data for currently traded options from “Yahoo! Finance”.

Following are the R packages we require:

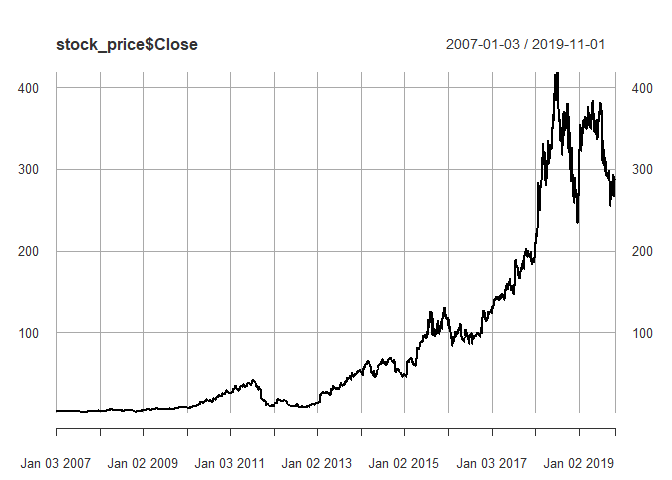
> library(quantmod)  
> library(lubridate)  
> library(dplyr)  
> library(ggplot2)  
> library(tidyr)

Following is the external customizable input for the code: It includes the ticker for the specific stock, the expiration date for the option, the risk free rate of interest, the number of past years’ data to be used to calculate volatility, and the number of steps to used in Binomial tree. We will use the interest rate on a three-month U.S. Treasury bill as our risk free rate of interest.

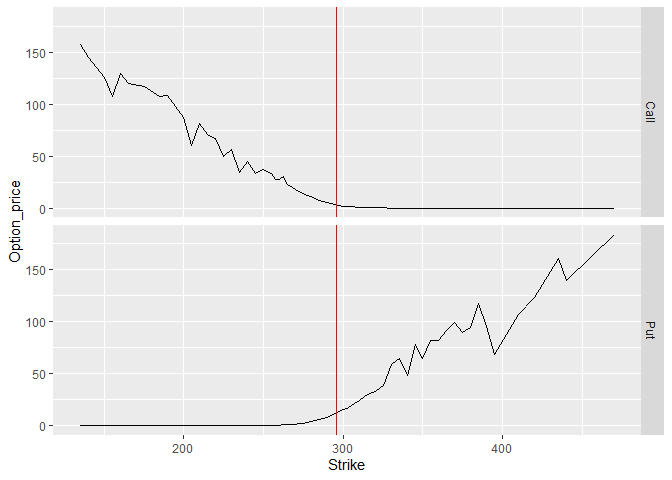
> # NOTE: check beforehand whether the option chain data for that particular ticker and expiration date is available on Yahoo Finance  
> ticker <- 'NFLX'  
> expiration\_date <- as.Date('2019-11-08')  
> riskfree\_rate <- 0.0193  
> # number of past years to calculate annualized volatility  
> Y <- 5  
> # number of steps in Binomial Tree  
> N <- 3

blank\_empty\_space

> options(timeout= 4000000)  
> options\_chain\_data <- getOptionChain(ticker, Exp = expiration\_date)  
> # separate call & put; select only strike, volume, last price which we need to calculate intrinsic value of an option  
> calls <- options\_chain\_data$calls[,c('Strike', 'Last', 'Vol')]  
> puts <- options\_chain\_data$puts[,c('Strike', 'Last', 'Vol')]  
> colnames(calls)[2] <- 'Market\_price'  
> colnames(puts)[2] <- 'Market\_price'  
>   
> stock\_price <- getSymbols(ticker, src = 'yahoo', auto.assign = F)  
> colnames(stock\_price)[4] <- 'Close'  
> colnames(stock\_price)[6] <- 'Adjusted'  
> stock\_price$Returns <- diff(log(stock\_price$Adjusted))  
>   
> contract\_date <- expiration\_date  
> month(contract\_date) <- month(expiration\_date) - 3  
> maturity\_time <- as.numeric(expiration\_date - contract\_date)/365  
>   
> So <- as.numeric(stock\_price$Close[contract\_date])  
>   
> plot(stock\_price$Close)



> calls %>%   
+ select(Strike, Call = Market\_price) %>%   
+ inner\_join(puts %>% select(Strike, Put = Market\_price), by = 'Strike') %>%   
+ gather(key = 'Option\_type', value = 'Option\_price', -Strike) %>%   
+ ggplot(aes(x = Strike, y = Option\_price)) +  
+ geom\_line() +  
+ geom\_vline(xintercept = So, col = 'red') +  
+ facet\_grid(rows = vars(Option\_type))



blank\_empty\_space

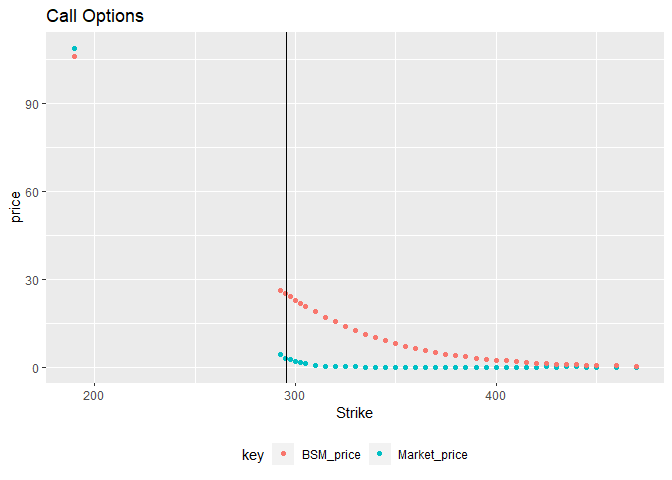
> # price of the option should not be less than intrinsic value  
> # the intrinsic value of a call option is equal to max[St − K, 0]  
> calls$intrinsic <- So - calls$Strike  
> calls$dummy <- ifelse(calls$intrinsic <= calls$Market\_price, 1, 0)  
> number\_of\_inconsistent\_calls <- sum(!calls$dummy)  
> calls <- calls %>% filter(dummy == 1)  
>   
> # the intrinsic value of a put option is equal to max[K −St , 0]  
> puts$intrinsic <- puts$Strike - So  
> puts$dummy <- ifelse(puts$intrinsic <= puts$Market\_price, 1, 0)  
> number\_of\_inconsistent\_puts <- sum(!puts$dummy)  
> puts <- puts %>% filter(dummy == 1)

blank\_empty\_space

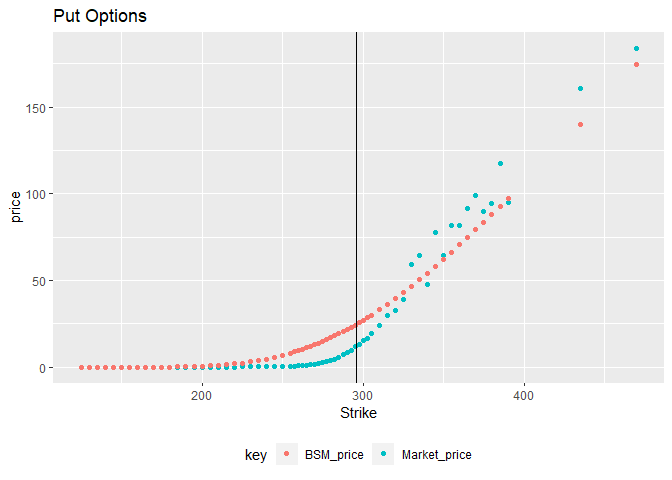
# BLACK-SCHOLES OPTION PRICING MODEL

blank\_empty\_space

> # multiply volatility of returns by 252 to annualize it  
> returns\_annualized\_volatility <- sd(stock\_price$Returns[paste(max(year(index(stock\_price))) - Y, '/')], na.rm = T)\*sqrt(252)  
>   
> # creating a function to calculate BSM option price  
> bs.opm <- function(S,K,T,r,sigma,type){  
+ d1<-(log(S/K)+(r+0.5\*sigma^2)\*T)/(sigma\*sqrt(T))  
+ d2<-d1-sigma\*sqrt(T)   
+ if(type=="Call"){   
+ opt.val<-S\*pnorm(d1)-K\*exp(-r\*T)\*pnorm(d2)   
+ }  
+ if(type=="Put"){   
+ opt.val<-K\*exp(-r\*T)\*pnorm(-d2)-S\*pnorm(-d1)   
+ }  
+ opt.val  
+ }  
>   
> calls$BSM\_price <- vapply(X = calls$Strike, FUN = bs.opm, FUN.VALUE = double(1), S = So, T = maturity\_time, r = riskfree\_rate, sigma = returns\_annualized\_volatility, type = 'Call')  
>   
> puts$BSM\_price <- vapply(X = puts$Strike, FUN = bs.opm, FUN.VALUE = double(1), S = So, T = maturity\_time, r = riskfree\_rate, sigma = returns\_annualized\_volatility, type = 'Put')  
>   
>   
> calls %>%   
+ select(Strike, Market\_price, BSM\_price) %>%   
+ gather(key = 'key', value = 'price', -Strike) %>%   
+ ggplot(aes(x = Strike, y = price, col = key)) +  
+ geom\_point() +  
+ geom\_vline(xintercept = So) +  
+ ggtitle('Call Options') +  
+ theme(legend.position="bottom")



> puts %>%   
+ select(Strike, Market\_price, BSM\_price) %>%   
+ gather(key = 'key', value = 'price', -Strike) %>%   
+ ggplot(aes(x = Strike, y = price, col = key)) +  
+ geom\_point() +  
+ geom\_vline(xintercept = So) +  
+ ggtitle('Put Options') +  
+ theme(legend.position="bottom")

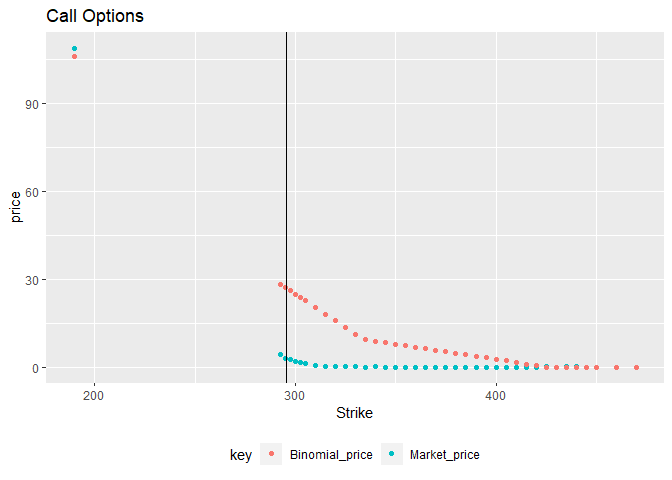


blank\_empty\_space

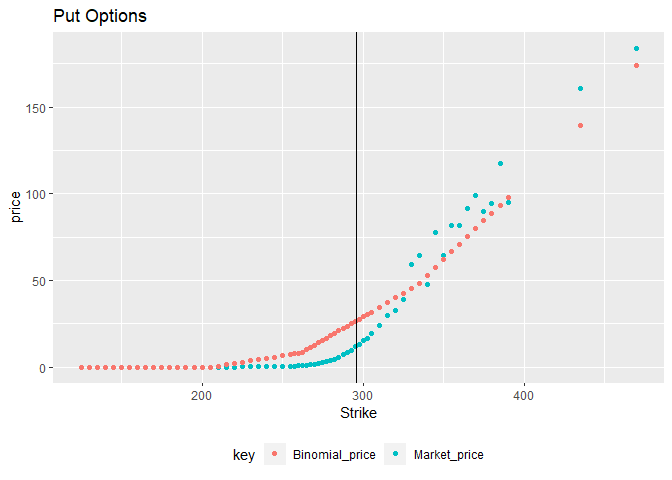
# BINOMIAL OPTION PRICING MODEL

blank\_empty\_space

> # Binomial OPM function  
> Binomial\_OPM <- function(S,K,T,r,sigma,n,type) {  
+ x=NA  
+ if (type=="Call") x=1  
+ if (type=="Put") x=-1  
+ if (is.na(x)) stop("Option Type can only be call or put")  
+ dt=T/ n  
+ u=exp(sigma\*sqrt(dt))  
+ d=1/ u  
+ p=((1+r\*dt)-d)/ (u-d)  
+ disc<- (1+r\*dt)  
+ OptVal<- x\*(S\*u^(0:n)\*d^(n:0)-K)  
+ OptVal=ifelse(OptVal<0,0,OptVal)  
+ for (j in seq(from=n-1,to=0,by=-1))  
+ for (i in 0:j)  
+ OptVal[i+1]=(p\*OptVal[i+2]+(1-p)\*OptVal[i+1])/disc  
+ value=OptVal[1]  
+ value  
+ }  
>   
> calls$Binomial\_price <- vapply(calls$Strike, FUN = Binomial\_OPM, FUN.VALUE = double(1), S = So, T = maturity\_time, r = riskfree\_rate, sigma = returns\_annualized\_volatility, n = N, type = 'Call')  
>   
> puts$Binomial\_price <- vapply(puts$Strike, FUN = Binomial\_OPM, FUN.VALUE = double(1), S = So, T = maturity\_time, r = riskfree\_rate, sigma = returns\_annualized\_volatility, n = N, type = 'Put')  
>   
> calls %>%   
+ select(Strike, Market\_price, Binomial\_price) %>%   
+ gather(key = 'key', value = 'price', -Strike) %>%   
+ ggplot(aes(x = Strike, y = price, col = key)) +  
+ geom\_point() +  
+ geom\_vline(xintercept = So) +  
+ ggtitle('Call Options') +  
+ theme(legend.position="bottom")



> puts %>%   
+ select(Strike, Market\_price, Binomial\_price) %>%   
+ gather(key = 'key', value = 'price', -Strike) %>%   
+ ggplot(aes(x = Strike, y = price, col = key)) +  
+ geom\_point() +  
+ geom\_vline(xintercept = So) +  
+ ggtitle('Put Options') +  
+ theme(legend.position="bottom")

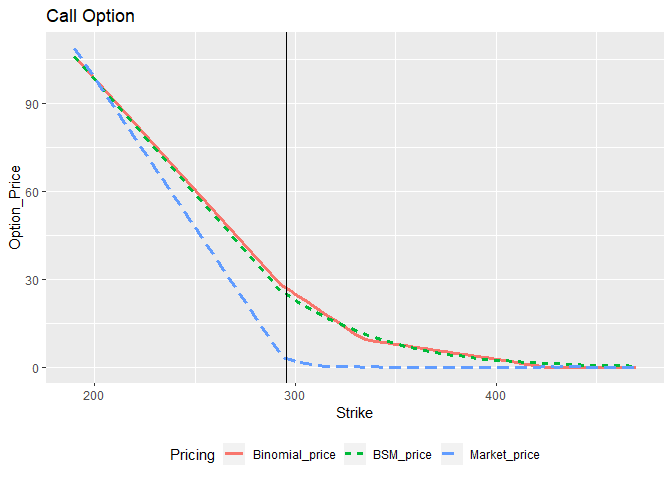


blank\_empty\_space

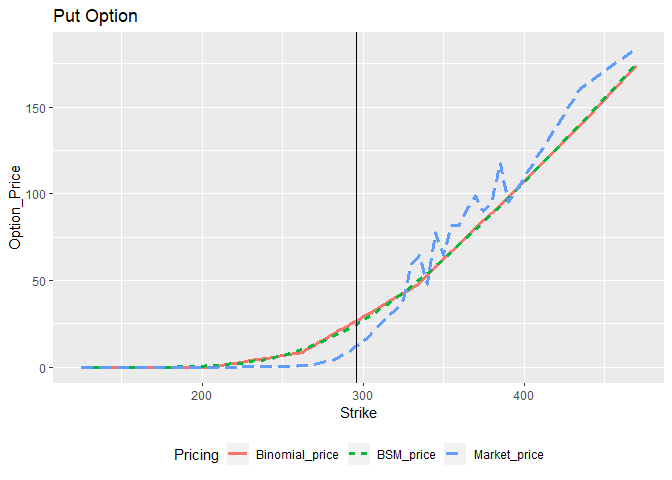
# COMPARING ALL THREE PRICES

blank\_empty\_space

> calls %>%   
+ select(Strike, Market\_price, BSM\_price, Binomial\_price) %>%   
+ gather(key = 'Pricing', value = 'Option\_Price', -Strike) %>%   
+ ggplot(aes(x = Strike, y = Option\_Price, col = Pricing, linetype = Pricing)) +  
+ geom\_line(size = 1.2) +  
+ geom\_vline(xintercept = So) +  
+ ggtitle('Call Option') +  
+ theme(legend.position="bottom")



> puts %>%   
+ select(Strike, Market\_price, BSM\_price, Binomial\_price) %>%   
+ gather(key = 'Pricing', value = 'Option\_Price', -Strike) %>%   
+ ggplot(aes(x = Strike, y = Option\_Price, col = Pricing, linetype = Pricing)) +  
+ geom\_line(size = 1.2) +  
+ geom\_vline(xintercept = So) +  
+ ggtitle('Put Option') +  
+ theme(legend.position="bottom")



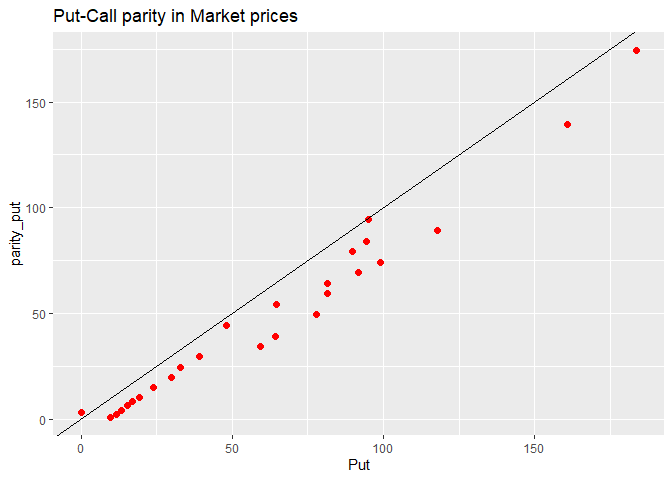
blank\_empty\_space

# EVALUATING PUT-CALL PARITY

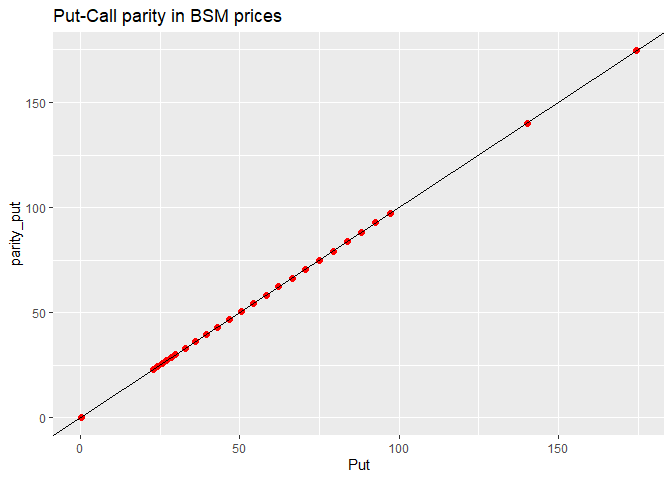
blank\_empty\_space

p + S = c + K exp (−rf ∗ T )

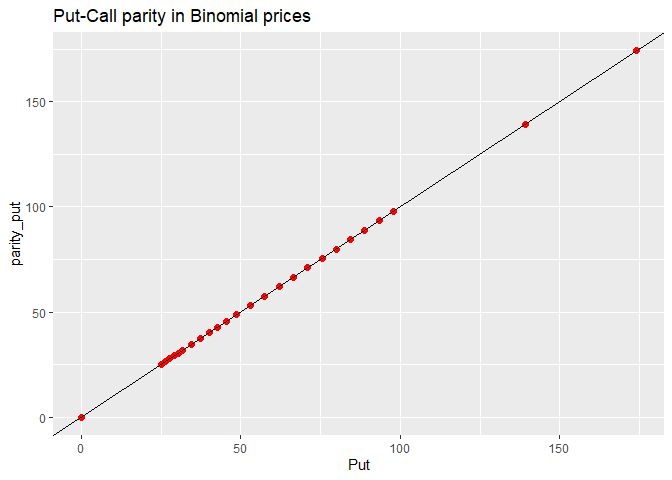
> # for Market price  
> parity\_data\_market <- puts %>%   
+ select(Strike, Put = Market\_price) %>%   
+ inner\_join(calls %>% select(Strike, Call = Market\_price), by = 'Strike')  
>   
> parity\_data\_market$parity\_put <- parity\_data\_market$Call + parity\_data\_market$Strike\*exp(-riskfree\_rate\*maturity\_time) - So  
>   
> parity\_data\_market %>%   
+ ggplot(aes(x = Put, y = parity\_put)) +  
+ geom\_point(col = 'red', size = 2) +  
+ geom\_abline() +  
+ ggtitle('Put-Call parity in Market prices')



> # for BSM price  
> parity\_data\_BSM <- puts %>%   
+ select(Strike, Put = BSM\_price) %>%   
+ inner\_join(calls %>% select(Strike, Call = BSM\_price), by = 'Strike')  
>   
> parity\_data\_BSM$parity\_put <- parity\_data\_BSM$Call + parity\_data\_BSM$Strike\*exp(-riskfree\_rate\*maturity\_time) - So  
>   
> parity\_data\_BSM %>%   
+ ggplot(aes(x = Put, y = parity\_put)) +  
+ geom\_point(col = 'red', size = 2) +  
+ geom\_abline() +  
+ ggtitle('Put-Call parity in BSM prices')



> # for Binomial price  
> parity\_data\_Binomial <- puts %>%   
+ select(Strike, Put = Binomial\_price) %>%   
+ inner\_join(calls %>% select(Strike, Call = Binomial\_price), by = 'Strike')  
>   
> parity\_data\_Binomial$parity\_put <- parity\_data\_Binomial$Call + parity\_data\_Binomial$Strike\*exp(-riskfree\_rate\*maturity\_time) - So  
>   
> parity\_data\_Binomial %>%   
+ ggplot(aes(x = Put, y = parity\_put)) +  
+ geom\_point(col = 'red', size = 2) +  
+ geom\_abline() +  
+ ggtitle('Put-Call parity in Binomial prices')



blank\_empty\_space

# CONCLUSION

blank\_empty\_space

# REFERENCES

blank\_empty\_space