Assignment2 - Napat L

Napat L

February 27, 2018

Problem 1: Binomial/Trinomial Tree Basics

1. Construct an additive binomial tree to calculate the values of the European Call option and one for the European Put option.

```
additiveBinomial <- function (K, T, S0, sig, r, N, type, isAmerican){
    #Time steps
    dt=T/N
    #Drift term
    nu=r-0.5*sig^2
    #Step siz up and down
    dxu=sqrt(sig^2*dt+(nu*dt)^2)
    dxd=-dxu
    #prob go up
    pu=0.5+0.5*(nu*dt/dxu)
    #discount
    disc=exp(-r*dt)
    dpu=disc*pu
    dpd=disc*(1-pu)

    cp = ifelse(type == "calls", 1, -1)

    edxu=exp(dxu)
    edxd=exp(dxd)

    M=N+1

    V <- matrix(0, nrow=M, ncol=M, dimnames=list( paste("S:", 1:(N+1),sep=""), paste("T=",0:N,
sep="")))
    S <- matrix(0, nrow=M, ncol=M, dimnames=list( paste("S:", 1:(N+1),sep=""), paste("T=",0:N,
sep="")))

    S[1,1] = S0

    for (j in 2:M) {
        S[1, j] <- S[1, j-1]*edxu
        for(i in 2:j) {
            S[i, j] <- S[i-1, j-1]*edxd
        }
    }

    for (j in 1:M) {
        V[M-j+1, M] <- max(0,cp*(S[M-j+1, M]-K))
    }

    for (j in (M-1):1) {
        for (i in 1:j) {
            V[i, j] <- dpu*V[i, j+1] + dpd*V[i+1, j+1]
```

```
            if(isAmerican == TRUE) {
                V[i, j] <- max(V[i, j], cp*(S[i, j]-K))
            }
        }
    }
    #print(S)
    #print(V)
    return(V[1,1])

}
```

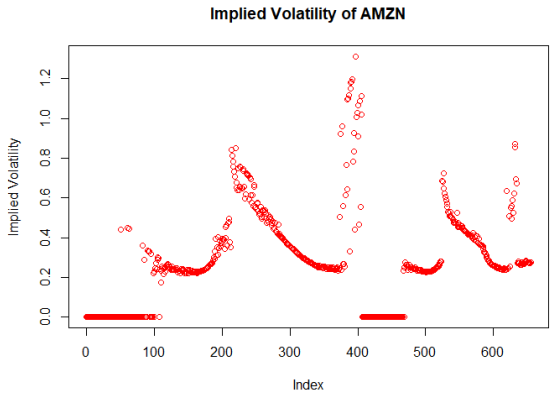
2.Download Option prices (you can use the Bloomberg Terminal, Yahoo! Finance, etc.) for an equity, for 3 different maturities (1 month, 2 months, and 3 months) and 20 strike prices. For each strike price in the data, calculate the corresponding implied volatility. You may use the data and calculations you have done for Homework 1

Downloading the AMZN Option data from Yahoo Finance, I got 3 maturities which have 406 strike prices for '2018-03-16', 228 strike prices for '2018-04-20' and, 21 strike prices for '2018-05-18'. The total rows are 655.

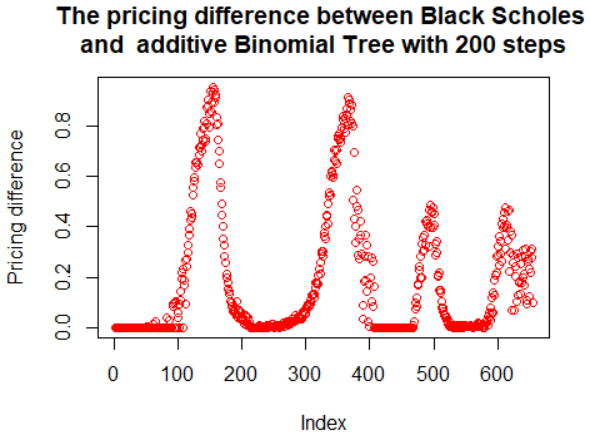
```
> summaryBy(Strike~maturity, data=DATA, FUN = length)
    maturity Strike.length
1 2018-03-16           406
2 2018-04-20           228
3 2018-05-18            21
```

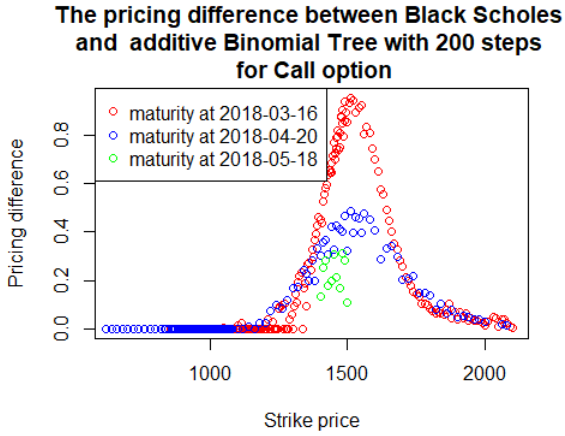
```
dim(DATA)

## [1] 655  15

unique(DATA$maturity)

## [1] "2018-03-16" "2018-04-20" "2018-05-18"
```

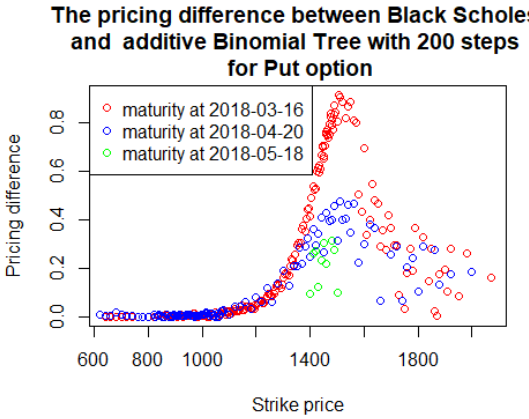
The figure below shows the implied volatilities. One smile represents one maturity of one option type. For example, the first smile is the implied volatilities of “2018-03-16” maturity of call option type.



From the comparisons of difference between BSM model and Binomial tree model, we can see the differences are not even over 1.00 meaning that with 200 tree steps, the option price from BSM model and Binomial tree model come very close to each other.







Code:

```
library(quantmod)

## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: TTR

## Warning: package 'TTR' was built under R version 3.4.2

## Version 0.4-0 included new data defaults. See ?getSymbols.

getOptEqtPrice <- function (sec, source = "yahoo", date = Sys.Date()){

    df <- data.frame()

    for( i in sec){

        data <- getOptionChain(i, Exp = NULL, src="yahoo")
        names(data)

        for (j in 1:length(names(data))){
            if(names(data[[j]])[1] == "calls" && !is.null(data[[j]]$calls)){
                df.temp <- data.frame(security = i ,data[[j]]$calls)
                df.temp$maturity <- as.Date(names(data)[j], "%b.%d.%Y")
                df.temp$type <- "calls"
                df.temp$time.stamp <- date
                df.temp$T.mat <- as.double((df.temp$maturity - Sys.Date())/252)

                df <- rbind(df, df.temp)
            }
            if(names(data[[j]])[2] == "puts" && !is.null(data[[j]]$puts)){

                df.temp <- data.frame(security = i, data[[j]]$puts)
                df.temp$maturity <- as.Date(names(data)[j], "%b.%d.%Y")
                df.temp$type <- "puts"
                df.temp$time.stamp <- date
                df.temp$T.mat <- as.double((df.temp$maturity - Sys.Date())/252)

                df <- rbind(df, df.temp)
            }
        }
    }

    df.price <- data.frame()
    for(i in sec){
        price.temp <- getSymbols(i, src = source, auto.assign = FALSE, from = date, to = date
+ 1)
        df.price.temp <- data.frame(security = i, equity.price = as.double(price.temp[1,4]))
        df.price <- rbind(df.price, df.price.temp)
    }
```

```
    df$equity.price <- 0

    for(i in 1:length(sec)){
        df[df$security == df.price$security[i],]$equity.price <- df.price$equity.price[i]
    }
    df$interest <- 1.59/100

    return(df)
}
setwd("C:/Users/nackz/Desktop/Stevens Institute/Subjects/FE621 - Computational Methods in Fina
nce/Assignments/Assignment 2")
#DATA <- getOptEqtPrice("AMZN")
#write.csv(DATA, file = "DATA.csv")
DATA <- read.csv("DATA.csv")
DATA <- subset(DATA, maturity == "2018-03-16" | maturity == "2018-04-20" | maturity == "2018-0
5-18")
DATA$maturity <- as.Date(DATA$maturity)

BS <- function (K, S, t, r, v, type){

    d1 <- (log(S/K)+(r+(v^2/2)*t))/(v*sqrt(t))
    d2 <- d1 - v * sqrt(t)

    OptPrice <- NULL

    if(tolower(type) == "calls"){
        OptPrice <- S*pnorm(d1) - K*exp(-r*t)*pnorm(d2)
    }else{
        OptPrice <- K*exp(-r*t) * pnorm(-d2) - S*pnorm(-d1)
    }

    return(OptPrice)
}

ImpVolBisection <- function (K, S, t, r, OptVal, type, start, end, epsilon = 1e-6){
    n = 1

    while(abs((BS(K, S, t, r, end, type) - OptVal) - (BS(K, S, t, r, start, type) - OptVal)) >
epsilon){

        if(((BS(K, S, t, r, end, type) - OptVal) * (BS(K, S, t, r, start, type) - OptVal)) > 0
){ mid = 0;break;}
        mid = (start + end)/2
        if(((BS(K, S, t, r, mid, type) - OptVal) * (BS(K, S, t, r, start, type) - OptVal)) < 0
){
            end = mid
        }else if(((BS(K, S, t, r, end, type) - OptVal) * (BS(K, S, t, r, mid, type) - OptVal))
< 0){
            start = mid
        }
        if(n > 10000){break;}
        n = n + 1
    }
    return(mid)
}
bi.vol.matrix <- matrix(, nrow = nrow(DATA), ncol = 1)
for(i in 1:nrow(DATA)){

    bi.vol <- ImpVolBisection(DATA$Strike[i], DATA$equity.price[i], DATA$T.mat[i], DATA$intere
st[i],
```

```
                                (DATA$Bid[i] + DATA$Ask[i])/2, DATA$type[i], 0.001, 800)
    bi.vol.matrix[i,1] <- bi.vol
}
DATA <- cbind(DATA[,1:15], data.frame(imp.vol = bi.vol.matrix[,1]))

#BS calculation
bs.vector <- vector(mode="numeric", length=nrow(DATA))

for( i in 1:nrow(DATA)){
    bs.vec <- BS(DATA$Strike[i], DATA$equity.price[i],
                 DATA$T.mat[i], DATA$interest[i], DATA$imp.vol[i], DATA$type[i])
    bs.vector[i] <- bs.vec
}
DATA <- cbind(DATA[,1:16], data.frame(bs.price = bs.vector))


#Binomial caluclation
bm.vector <- vector(mode="numeric", length=nrow(DATA))

for( i in 1:nrow(DATA)){
    bm.vec <- additiveBinomial(DATA$Strike[i], DATA$T.mat[i], DATA$equity.price[i],
                               DATA$imp.vol[i], DATA$interest[i], N=200, DATA$type[i], isAmeri
can = FALSE)
    bm.vector[i] <- bm.vec
}
DATA <- cbind(DATA[,1:17], data.frame(bm.price = bm.vector))

DATA$bs.bm.diff <- ifelse(abs(DATA$bs.price - DATA$bm.price) < 10e-5, 0, abs(DATA$bs.price - D
ATA$bm.price))

plot(DATA$bs.bm.diff, main = "The pricing difference between Black Scholes \nand  additive Bin
omial Tree with 200 steps",
     xlab = "Index", ylab = "Pricing difference", type = "p", col = "red")

plot(subset(DATA, type == "calls" & maturity == "2018-03-16")$Strike ,subset(DATA, type == "ca
lls" & maturity == "2018-03-16")$bs.bm.diff,
     main = "The pricing difference between Black Scholes \nand  additive Binomial Tree with 2
00 steps \n for Call option",
     xlab = "Strike price", ylab = "Pricing difference", type = "p", col = "red")
points(subset(DATA, type == "calls" & maturity == "2018-04-20")$Strike ,subset(DATA, type == "
calls" & maturity == "2018-04-20")$bs.bm.diff, col = "blue")
points(subset(DATA, type == "calls" & maturity == "2018-05-18")$Strike ,subset(DATA, type == "
calls" & maturity == "2018-05-18")$bs.bm.diff, col = "green")
legend("topleft", c("maturity at 2018-03-16", "maturity at 2018-04-20", "maturity at 2018-05-1
8"),
       col=c("red", "blue", "green"), pch = c(1,1,1),ncol = 1, cex = 1)

plot(subset(DATA, type == "puts" & maturity == "2018-03-16")$Strike ,subset(DATA, type == "put
s" & maturity == "2018-03-16")$bs.bm.diff,
     main = "The pricing difference between Black Scholes \nand  additive Binomial Tree with 2
00 steps \n for Put option",
     xlab = "Strike price", ylab = "Pricing difference", type = "p", col = "red")
points(subset(DATA, type == "puts" & maturity == "2018-04-20")$Strike ,subset(DATA, type == "p
uts" & maturity == "2018-04-20")$bs.bm.diff, col = "blue")
points(subset(DATA, type == "puts" & maturity == "2018-05-18")$Strike ,subset(DATA, type == "p
uts" & maturity == "2018-05-18")$bs.bm.diff, col = "green")
legend("topleft", c("maturity at 2018-03-16", "maturity at 2018-04-20", "maturity at 2018-05-1
8"),
       col=c("red", "blue", "green"), pch = c(1,1,1),ncol = 1, cex = 1)
```

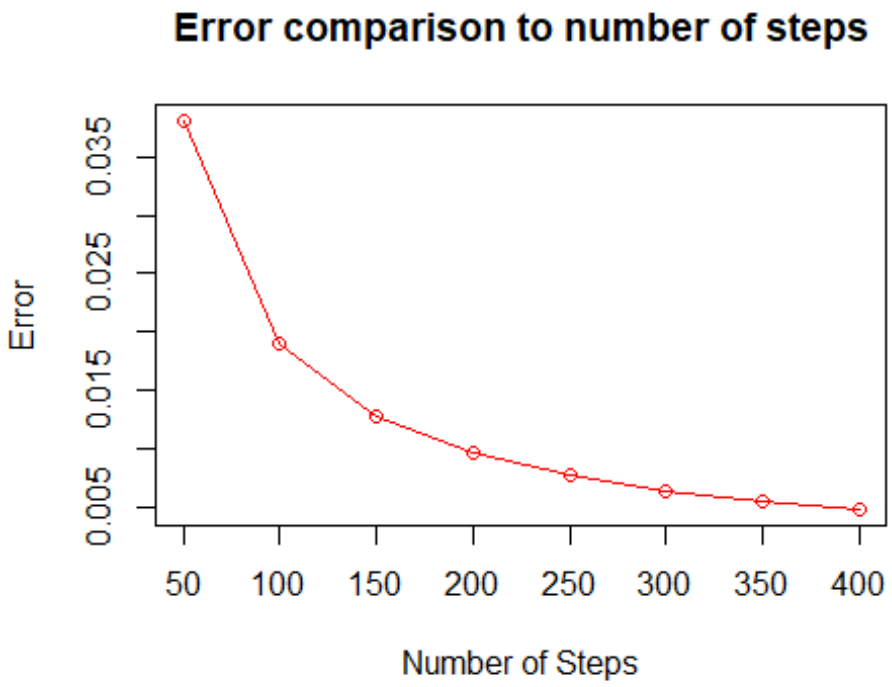
3.Compute and plot the absolute error

$$\varepsilon_N = \left| C^{BSM}(S_0, K, T, r; \sigma) - C_N^{BT}(S_0, K, T, r; \sigma) \right|,$$

From the figure below, when we increase the tree steps, the difference is clearly reducing.

```
N <-c(50, 100, 150, 200, 250, 300, 350, 400)
errorEstimation(N)
```

```
## [1] "Error 50  :  0.0381028119807709"   "Error 100  :  0.0190740048479867"
## [3] "Error 150  :  0.0127209312426828"  "Error 200  :  0.00954253316552922"
## [5] "Error 250  :  0.00763490380053611" "Error 300  :  0.00636290602328771"
## [7] "Error 350  :  0.00545421661736967" "Error 400  :  0.00477263432105346"
```



Code:

```
errorEstimation <- function(N){
    bsm.price <- vector(mode="numeric", length=length(N))
    bt.price <-vector(mode="numeric", length=length(N))
    error <-vector(mode="numeric", length=length(N))


    for(i in 1:length(N)){
        bt <-additiveBinomial(K=100,T=1,S=100,sig=0.2,r=0.06, N[i], type = "calls", isAmerican
=FALSE)
        bsm <-BS(100, 100, 1, 0.06, 0.2, "calls")
        bt.price[i] <- bt
```

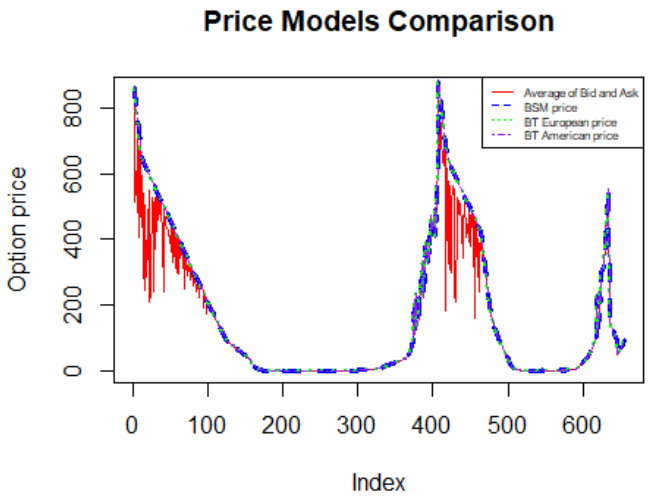
```
        bsm.price[i] <- bsm
        error[i] <- bsm.price[i] - bt.price[i]
    }
    print(paste("Error", N, " : ", error))
    plot(N,error,xlab = "Number of Steps",ylab = "Error", type="o",col="red", main = "Error co
mparison to number of steps")
}
```

4.Implement an additive binomial tree to calculate the American option, both Call and Put. Repeat the steps in part 2) and calculate the respective option prices as if they are American.

```
bm.american.vector <- vector(mode="numeric", length=nrow(DATA))

for( i in 1:nrow(DATA)){
    bm.american.vec <- additiveBinomial(DATA$Strike[i], DATA$T.mat[i], DATA$equity.price[i],
                                DATA$imp.vol[i], DATA$interest[i], N=200, DATA$type[i], isAmeri
can = TRUE)
    bm.american.vector[i] <- bm.american.vec
}
DATA <- cbind(DATA[,1:19], data.frame(bm.american.price = bm.american.vector))
```

5.Create a table which contains the following columns: Bid and Ask values, Black Scholes price, European and American prices calculated using the binomial tree. Then write comments about the observed differences between the various option valuations and how they compare with the actual bid/ask values.

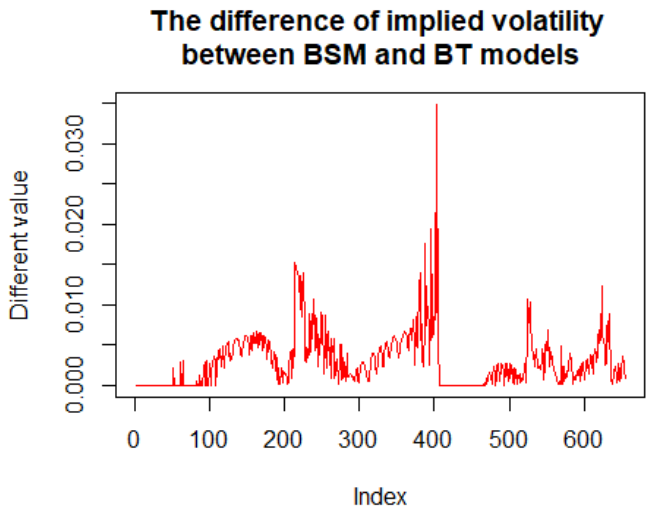
Using the average of Bid and Ask price to approximate the option price, as we can see; the option prices from 3 models are very close to each other. However, average of Bid and Ask price fluctuates in the high option price.



```
table <- DATA[,c(6,7,17,18,20)]
table$avg.price <- (table$Bid + table$Ask)/2


plot(table$avg.price , type = "l", col = "red", xlab = "Index", ylab = "Option price", main =
"Price Models Comparison")
lines(table$bs.price, col = "blue", lwd = 3, type = "l", lty = 2)
lines(table$bm.price, col = "green", lwd = 2, type = "l", lty = 3)
lines(table$bm.american.price, col = "purple", type = "l", lty = 4)

legend("topright", c("Average of Bid and Ask", "BSM price", "BT European price", "BT American
price"),
       col=c("red", "blue", "green", "purple"), lty = c(1,2,3,4),ncol = 1, cex = 0.5)
```

6.Using the binomial tree for American Calls and Puts, calculate the implied volatility corresponding to the data you have downloaded in part (2). You will need to use the bisection or Newton/secant method of finding roots with the respective binomial trees. Compare these values of the implied volatility with the volatilities from Homework 1, Problem 1c (or with the ones obtained in part 2 of this problem). Write detailed observations.

The difference of implied volatilities that we got from BSM model and Binomial tree model with 40 tree steps are very close to each other as we can see the differences are not even over 0.04.

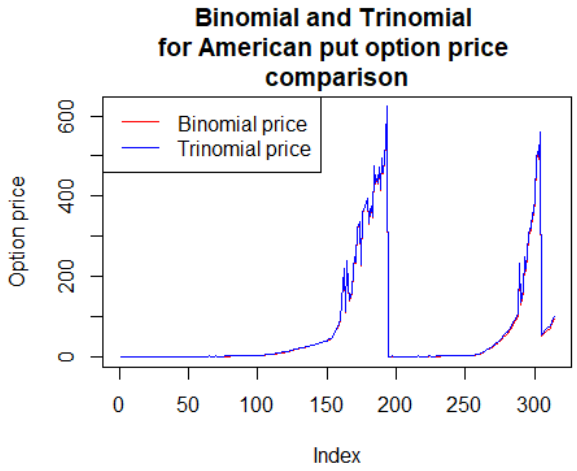


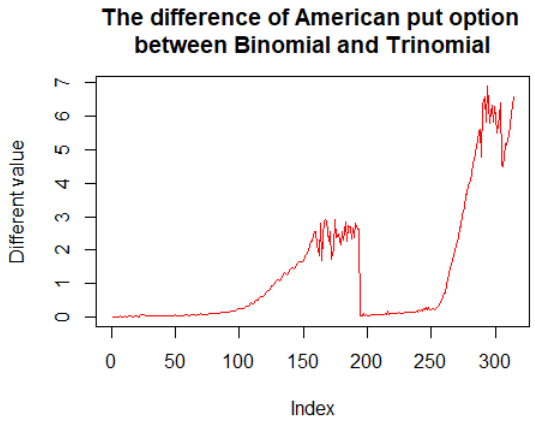

```
ImpVolBisectionTree <- function (K, S, t, r, OptVal, type, start, end, epsilon = 1e-6){
    n = 1
    while(abs((additiveBinomial(K,t,S,end,r,40,type, TRUE) - OptVal) - (additiveBinomial(K,t,S
,start,r,40,type, TRUE) - OptVal)) > epsilon){

        if(((additiveBinomial(K,t,S,end,r,40,type, TRUE) - OptVal) * (additiveBinomial(K,t,S,s
tart,r,40,type, TRUE) - OptVal)) > 0){ mid = 0;break;}

        mid = (start + end)/2
```

```
        if(((additiveBinomial(K,t,S,mid,r,40,type, TRUE) - OptVal) * (additiveBinomial(K,t,S,s
tart,r,40,type, TRUE) - OptVal)) < 0){
            end = mid
        }else if(((additiveBinomial(K,t,S,end,r,40,type, TRUE) - OptVal) * (additiveBinomial(K
,t,S,mid,r,40,type, TRUE) - OptVal)) < 0){
            start = mid
        }
        if(n > 10000){break;}
        n = n + 1
    }
    return(mid)
}
bi.tree.vol.matrix <- matrix(, nrow = nrow(DATA), ncol = 1)

for(i in 1:nrow(DATA)){

    bi.tree.vol <- ImpVolBisectionTree(DATA$Strike[i], DATA$equity.price[i], DATA$T.mat[i], DA
TA$interest[i],
                               (DATA$Bid[i] + DATA$Ask[i])/2, DATA$type[i], 0.001, 10)
    bi.tree.vol.matrix[i,1] <- bi.tree.vol
}
DATA <- cbind(DATA[,1:20], data.frame(tree.vol = bi.tree.vol.matrix))

plot(abs(DATA$imp.vol - DATA$tree.vol), type = "l", col = "red", main = "The difference of imp
lied volatility \nbetween BSM and BT models",
     xlab = "Index", ylab = "Different value")
```

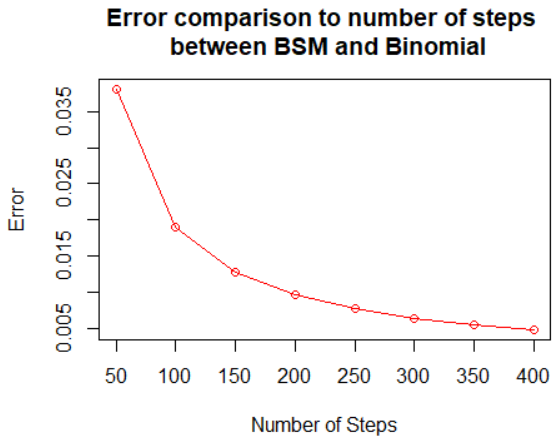
7.Implement a trinomial tree to price an American Put option. Hint. See Chapter 3 in [1]. Use this tree with the data in this problem and compare with the results obtained using the binomial tree.

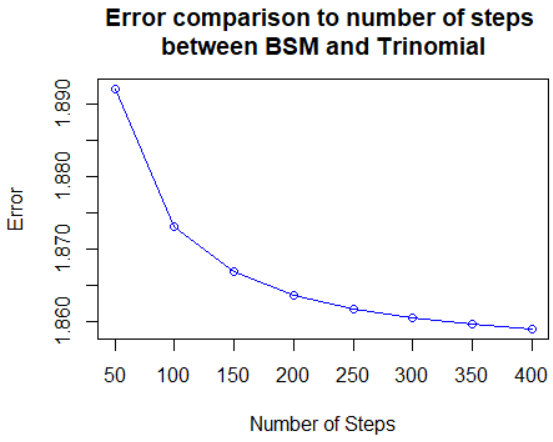
The option prices from Binomial tree and Trinomial tree models with 200 tree steps are pretty close to each other.





When we look at the absolute errors, it seems like Binomial tree performs closer to BSM model than Trinomial tree model.






```
N <-c(50, 100, 150, 200, 250, 300, 350, 400)
errorEstimationBt(N)

## [1] "Error Binomial  50  :  0.0381028119807709"
## [2] "Error Binomial  100  :  0.0190740048479867"
## [3] "Error Binomial  150  :  0.0127209312426828"
## [4] "Error Binomial  200  :  0.00954253316552922"
## [5] "Error Binomial  250  :  0.00763490380053611"
## [6] "Error Binomial  300  :  0.00636290602328771"
## [7] "Error Binomial  350  :  0.00545421661736967"
## [8] "Error Binomial  400  :  0.00477263432105346"
```

```
errorEstimationTt(N)
```

```
## [1] "Error Trinomial  50  :  1.8920574314598"
## [2] "Error Trinomial  100  :  1.87315374563835"
## [3] "Error Trinomial  150  :  1.86687381654589"
## [4] "Error Trinomial  200  :  1.86373777370857"
## [5] "Error Trinomial  250  :  1.86185739795815"
## [6] "Error Trinomial  300  :  1.86060432859028"
## [7] "Error Trinomial  350  :  1.85970953044575"
## [8] "Error Trinomial  400  :  1.85903857059594"
```

Code:

```
Trinomial <- function (K, T, S0, sig, r, div, N, type, isAmerican){
    #Time steps
    dt <- T/N
    dx <- sig*sqrt(3*dt)

    #Drift term
    nu <- r-div- 0.5*sig^2

    #prob to go up
    pu <- 0.5*((sig^2*dt+nu^2*dt^2)/dx^2+nu*dt/dx)
    #prob to go middle
    pm <- 1.0-(sig^2*dt+nu^2*dt^2)/dx^2
    #prob to go down
    pd <- 0.5*((sig^2*dt+nu^2*dt^2)/dx^2 -nu*dt/dx )
    #print(paste(pu,pm,pd))
    #compute discount factor
    disc <- exp(-r*dt)

    cp = ifelse(type == "calls", 1, -1)

    Row <- 2*N+1
    Col <- N+1

    S <- matrix(0, nrow = Row, ncol = Col,
                dimnames=list( paste("S", N:-N, sep=":"), paste("T", 0:N, sep="=")))
    V <- matrix(0, nrow = Row, ncol = Col,
                dimnames=list( paste("S", N:-N, sep=":"), paste("T", 0:N, sep="=")))
    S[Col, 1] = S0

    for (j in 1:N){
        for(i in (Col-j+1):(Col+j-1)){
            S[i-1, j+1] = S[i, j] * exp(dx)
            S[i , j+1] = S[i, j]
            S[i+1, j+1] = S[i, j] * exp(-dx)
        }
    }
    for (i in 1:Row){
        V[i, Col] = max( 0, cp * (S[i,Col]-K))
    }
    for (j in (Col-1):1){
```

```
        for(i in (Col-j+1):(Col+j-1)){
            V[i, j] = disc * (pu*V[i-1,j+1] + pm*V[i, j+1] + pd*V[i+1,j+1])

            if(isAmerican){
                V[i, j] = max(V[i, j], cp * (S[i, j] - K))
            }
        }
    }
    #print(V)
    #print(S)
    return(V[Col,1])
}
tt.american.vector <- vector(mode="numeric", length=nrow(DATA))

for( i in 1:nrow(DATA)){

    tt.american.vec <- Trinomial(DATA$Strike[i], DATA$T.mat[i], DATA$equity.price[i],
                                        DATA$imp.vol[i], DATA$interest[i],div=0.03, N=200, DAT
A$type[i], isAmerican = TRUE)

    tt.american.vector[i] <- tt.american.vec
}

DATA <- cbind(DATA[,1:21], data.frame(tt.american.price = tt.american.vector))
DATA$tt.american.price <- ifelse(is.nan(DATA$tt.american.pric), 0, DATA$tt.american.price)

plot(subset(DATA, type == "puts")$bm.american.price, col = "red", type = "l",
     main = "Binomial and Trinomial \nfor American put option price \ncomparison",
     xlab = "Index", ylab = "Option price")
lines(subset(DATA, type == "puts")$tt.american.price, col = " blue")
legend("topleft", c("Binomial price", "Trinomial price"),
       col=c("red", "blue"), lty = c(1,1), ncol = 1, cex = 1)

plot(abs(subset(DATA, type == "puts")$bm.american.price - subset(DATA, type == "puts")$tt.amer
ican.price)
     , type = "l", col = "red", main = "The difference of American put option \nbetween Binomi
al and Trinomial",
     xlab = "Index", ylab = "Different value")

errorEstimationBt <- function(N){
    bsm.price <- vector(mode="numeric", length=length(N))
    bt.price <- vector(mode="numeric", length=length(N))
    tt.price <- vector(mode="numeric", length=length(N))
    error.bt <-vector(mode="numeric", length=length(N))
    error.tt <-vector(mode="numeric", length=length(N))


    for(i in 1:length(N)){
        bt <- additiveBinomial(K=100,T=1,S=100,sig=0.2,r=0.06, N[i], type = "calls", isAmerica
n=TRUE)
        bsm <-BS(100, 100, 1, 0.06, 0.2, "calls")
        bt.price[i] <- bt
        bsm.price[i] <- bsm
        error.bt[i] <- bsm.price[i] - bt.price[i]
    }
    print(paste("Error Binomial ", N, " : ", error.bt))

    plot(N,error.bt,xlab = "Number of Steps",ylab = "Error", type="o", col="red"
         , main = "Error comparison to number of steps \n between BSM and Binomial")
}
```

```
errorEstimationTt <- function(N){
    bsm.price <- vector(mode="numeric", length=length(N))
    tt.price <- vector(mode="numeric", length=length(N))
    error.tt <-vector(mode="numeric", length=length(N))


    for(i in 1:length(N)){
        tt <- Trinomial(K=100,T=1, S0=100, sig=0.2,r=0.06,div=0.03,N[i], type="calls", isAmeri
can=TRUE)
        bsm <-BS(100, 100, 1, 0.06, 0.2, "calls")
        tt.price[i] <- tt
        bsm.price[i] <- bsm
        error.tt[i] <- bsm.price[i] - tt.price[i]
    }
    print(paste("Error Trinomial ", N, " : ", error.tt))

    plot(N,error.tt,xlab = "Number of Steps",ylab = "Error", type="o", col="blue"
         , main = "Error comparison to number of steps \nbetween BSM and Trinomial")
}
```

Problem 2. Pricing Exotic Options

1.Construct a binomial tree to calculate the price of an European Up and-Out call option. Use S0 = 10, strike K = 10, maturity T = 0:3, volatility sigma = 0:2, short rate r = 0:01, dividends div = 0, and barrier H = 11. Use as many steps in your tree as you think are necessary.

With pre-conditions from the question, we get the option prices as 0.05569624.

```
BinomialExo(K=10, T=0.3, S0=10, sig=0.2, r=0.01,
                N = 800, otype = "calls", isAmerican = FALSE,
                div = 0, mtype = "UO", H=11)

## [1] 0.05569624
```

Code:

```
BinomialExo <- function(K, T, S0, sig, r, N, otype, isAmerican, div, mtype, H=11)
{
    dt <- T/N
    nu <- r-div-0.5*sig*sig
    dxu <- sqrt(sig*sig*dt+((nu*dt)^2))
    dxd <- -dxu
    pu <- 0.5+0.5*(nu*dt/dxu)
    pd <- 1-pu
    disc <- exp(-r*dt)

    Row <-  2*N+1
    Col = N+1
    cp = ifelse(otype == "calls", 1, -1)


    V <- matrix(0, nrow=Row, ncol = Col, dimnames=list(
        paste("S", N:-N, sep=":"), paste("T", 0:N, sep="=")))
    S <- matrix(0, nrow=Row, ncol= Col, dimnames=list(
        paste("S", N:-N, sep=":"), paste("T", 0:N, sep="=")))

    S[Col, 1] <- S0

    for (j in 1:N) {
        for(i in (Col-j+1):(Col+j-1)) {
            S[i-1, j+1] <- S[i, j]*exp(dxu)
            S[i+1, j+1] <- S[i, j]*exp(dxd)
        }
    }
    for (i in 1:Row) {
        if(mtype == "UO"){
            V[i, Col] <- ifelse((S[i,Col] < H),max(0,cp * (S[i, Col]-K)),0)
        }
        else if(mtype == "DO")
            V[i, Col] <- ifelse((S[i,Col] > H),max(0,cp * (S[i, Col]-K)),0)
        else if(mtype == "UI")
            V[i, Col] <- ifelse((S[i,Col] > H),max(0,cp * (S[i, Col]-K)),0)
    }
    for (j in (Col-1):1) {
        for(i in (Col-j+1):(Col+j-1)) {
```

```
            V[i, j] = disc * (pu*V[i-1,j+1] + pd*V[i+1,j+1])
            if(isAmerican){
                V[i, j] <- max(V[i, j], cp * (S[i, j] - K))
            }
            if(mtype == "UO" && S[i,j] >= H){
                V[i, j] <- 0
            }
            else if(mtype == "DO" && S[i,j] <= H){
                V[i, j] <- 0
            }
            else if(mtype == "UI" && S[i,j] >= H){
                V[i, j] <- 0
            }
        }
    }
    #print(V)
    return(V[Col,1])
}
```

2.To price the European Up-and-Out Call option, explicit formulae can be found. Implement the formula (5.2) from [2] and compare your results with part (1). Use the same parameters as before. What can you observe?

The option prices from analytical method and Binomial tree method are close to each other. As we can see, the price from analytical method is 0.05243523 and from Binomial tree is 0.05569624.

```
BS.uo(10,10,0.3,0.01,0.2,200,11,0)

## [1] 0.05243523

BinomialExo(K=10, T=0.3, S0=10, sig=0.2, r=0.01,
                N = 800, otype = "calls", isAmerican = FALSE,
                div = 0, mtype = "UO", H=11)

## [1] 0.05569624
```

Code:

```
BS.uo <- function(S0,K,T,r,sig,N,H,d, type = "calls") {

    dbs<-function(S0,K) {
        return((log(S0/K)+v*T)/(sig*sqrt(T)))
    }
    v=r-d-sig^2/2
    UOBS <- BS(K,S0,T,r,sig,type) - BS(H,S0,T,r,sig,type) - (H-K)*exp(-r*T)*pnorm(dbs(S0,H))-
            (H/S0)^((2*v)/sig^2)*(BS(K,H^2/S0,T,r,sig,type)-BS(H,H^2/S0,T,r,sig,type)-
                                      (H-K)*exp(-r*T)*pnorm(dbs(H,S0)))
    return(UOBS)
```

```
}
```

3.Price an European Up-and-In call option, using the same parameters as before. Hint. Two methods can be employed: the analytical solution in (5.1) or the In-Out parity. Use both methods in order to verify your results.

For Up-and-In call option with binomial tree implanted, by using analytical method; the price is 0.3974818. and using In-Out parity; the price is 0.3979651 which are very close to each other. Moreover, using exotic binomial tree to calculate up-and-in call option, the price is 0.425381. It is also nearly to the other methods.

```
BS.uo(10,10,0.3,0.01,0.2,200,11,0)

## [1] 0.05243523

BS.ui(10,10,0.3,0.01,0.2,11,0)

## [1] 0.3974818

UO.price <- BS(10, 10, 0.3, 0.01, 0.2, "calls") - BS.ui(10,10,0.3,0.01,0.2,11,0)
UO.price

## [1] 0.05291854

UI.price <- BS(10, 10, 0.3, 0.01, 0.2, "calls") - BS.uo(10,10,0.3,0.01,0.2,200,11,0)
UI.price

## [1] 0.3979651

BinomialExo(K=10, T=0.3, S0=10, sig=0.2, r=0.01,
            N = 800, otype = "calls", isAmerican = TRUE,
            div = 0, mtype = "UI", H=11)

## [1] 0.4253817

BinomialExo(K=10, T=0.3, S0=10, sig=0.2, r=0.01,
            N = 800, otype = "calls", isAmerican = TRUE,
            div = 0, mtype = "DO", H=11)

## [1] 0
```

Code:

```
BS.ui <- function(S0,K,T,r,sig,H,d, type = "puts") {

    dbs<-function(S0,K) {
        return((log(S0/K)+v*T)/(sig*sqrt(T))) }
    v = r-d-sig^2/2
    UIBS <- ((H/S0)^((2*v)/sig^2))*((BS(K,H^2/S0,T,r,sig,type)-BS(H,H^2/S0,T,r,sig,type)+
                                        (H-K)*exp(-r*T)*pnorm(-dbs(H,S0))))+
        BS(H,H^2/S0,T,r,sig,type)+(H-K)*exp(-r*T)*pnorm(dbs(S0,H))
    return(UIBS)
}
```

4.Construct a binomial tree to calculate the price of an American Up and-In call option.

Using Binomial tree model, we get the Up and In-call option price as 0.397186 and 0.3992622 for analytical method and In-Out parity method, respectively. From the previous question, using BSM model, we get the Up and In-call option price as 0.3974818 and 0.3979651 for analytical method and In-Out parity method, respectively. Therefore, we can see that the prices are not much different.

```
BS.uo.bt(10,10,0.3,0.01,0.2,50,11,0)

## [1] 0.05148125

BS.ui.bt(10,10,0.3,0.01,0.2,11,0, 50)

## [1] 0.3971856

UO.bt.price <- Binomial(10,0.3,10,0.2,0.01,200,"calls",TRUE) - BS.ui.bt(10,10,0.3,0.0
1,0.2,11,0, 50)
UO.price

## [1] 0.05291854

UI.bt.price <- Binomial(10,0.3,10,0.2,0.01,200,"calls",TRUE) - BS.uo.bt(10,10,0.3,0.0
1,0.2,50,11,0)
UI.bt.price

## [1] 0.3992622
```

Code:

```
Binomial <- function (K, T, S0, sig, r, N, type, isAmerican){
    #Time steps
    dt=T/N
    #Drift term
    nu=r-0.5*sig^2
    #Step siz up and down
    dxu=sqrt(sig^2*dt+(nu*dt)^2)
    dxd=-dxu
    #prob go up
    pu=0.5+0.5*(nu*dt/dxu)
    #discount
    disc=exp(-r*dt)
    dpu=disc*pu
    dpd=disc*(1-pu)

    cp = ifelse(type == "calls", 1, -1)

    edxu=exp(dxu)
    edxd=exp(dxd)
```

```
    M=N+1

    V <- matrix(0, nrow=M, ncol=M, dimnames=list( paste("S:", 1:(N+1),sep=""), paste("T=",0:N,
sep="")))
    S <- matrix(0, nrow=M, ncol=M, dimnames=list( paste("S:", 1:(N+1),sep=""), paste("T=",0:N,
sep="")))

    S[1,1] = S0

    for (j in 2:M) {
        S[1, j] <- S[1, j-1]*edxu
        for(i in 2:j) {
            S[i, j] <- S[i-1, j-1]*edxd
        }
    }

    for (j in 1:M) {
        V[M-j+1, M] <- max(0,cp*(S[M-j+1, M]-K))
    }

    for (j in (M-1):1) {
        for (i in 1:j) {
            V[i, j] <- dpu*V[i, j+1] + dpd*V[i+1, j+1]
            if(isAmerican == TRUE) {
                V[i, j] <- max(V[i, j], cp*(S[i, j]-K))
            }
        }
    }
    #print(S)
    #print(V)
    return(V[1,1])

}
BS.uo.bt <- function(S0,K,T,r,sig,N,H,d, type = "calls") {

    dbs<-function(S0,K) {
        return((log(S0/K)+v*T)/(sig*sqrt(T)))
    }
    v=r-d-sig^2/2
    UOBS <- Binomial(K,T,S0,sig,r,N,type,TRUE) - Binomial(H,T,S0,sig,r,N,type,TRUE) - (H-K)*ex
p(-r*T)*pnorm(dbs(S0,H))-
        (H/S0)^((2*v)/sig^2)*(Binomial(K,T,H^2/S0,sig,r,N,type,TRUE)-Binomial(H,T,H^2/S0,sig,r
,N,type,TRUE)-
                                    (H-K)*exp(-r*T)*pnorm(dbs(H,S0)))
    return(UOBS)
}
BS.ui.bt <- function(S0,K,T,r,sig,H,d, N, type = "puts") {

    dbs<-function(S0,K) {
        return((log(S0/K)+v*T)/(sig*sqrt(T)))
    }
    v = r-d-sig^2/2
    UIBS <- ((H/S0)^((2*v)/sig^2))*((Binomial(K,T,H^2/S0,sig,r,N,type,TRUE)-Binomial(H,T,H^2/S
0,sig,r,N,type,TRUE)+
                                        (H-K)*exp(-r*T)*pnorm(-dbs(H,S0))))+
        Binomial(H,T,H^2/S0,sig,r,N,type,TRUE)+(H-K)*exp(-r*T)*pnorm(dbs(S0,H))
    return(UIBS)
}
```

Problem 3. Adapting The Binomial Model to The Varying Volatility

1.Report the results in a table

When we don't put the exact delta x on the function, the function will approximate the mean of delta t and then approximate the delta x which is 0.06202726 and it is in the range between 0.02 and 0.08. However, with delta x equals to 0.06202726, it cannot make the sum of delta t equal to 1.

```
> delta.t.bar <- Tm/N
> delta.t.bar
[1] 0.09090909
> delta.x <- sqrt((sig.bar^2)*delta.t.bar + (v.bar^2)*(delta.t.bar^2))
> delta.x
[1] 0.06202726
```

```
> sum(delta.t/N)
[1] 0.09925766
> sum(delta.t)
[1] 1.091834
>
```

```
PrintBtTimeVaryingVol(0)
```

```
##      i      r   sig         v    delta.t         p
## 1    0 0.0500 0.200 0.0300000 0.09597727 0.5232101
## 2    1 0.0505 0.201 0.0302995 0.09502467 0.5232091
## 3    2 0.0510 0.202 0.0305980 0.09408621 0.5232063
## 4    3 0.0515 0.203 0.0308955 0.09316161 0.5232017
## 5    4 0.0520 0.204 0.0311920 0.09225061 0.5231953
## 6    5 0.0525 0.205 0.0314875 0.09135294 0.5231872
## 7    6 0.0530 0.206 0.0317820 0.09046833 0.5231774
## 8    7 0.0535 0.207 0.0320755 0.08959654 0.5231661
## 9    8 0.0540 0.208 0.0323680 0.08873732 0.5231531
## 10   9 0.0545 0.209 0.0326595 0.08789043 0.5231387
## 11  10 0.0550 0.210 0.0329500 0.08705563 0.5231228
## 12  11 0.0555 0.211 0.0332395 0.08623269 0.5231054
```

Code:

```
PrintBtTimeVaryingVol <- function(delta.x){

    N <- 11
    Tm <- 1

    r <- 0.05*(1 + 0.01*(0:N))
    sig <- 0.2*(1 + 0.005*(0:N))
    v <- r - (0.5*(sig^2))

    sig.bar <- mean(sig)
    sig.bar <- mean(sig)
    delta.t.bar <- Tm/N
    v.bar <- mean(v)
```

```
    if(delta.x == 0){
        delta.x <- sqrt((sig.bar^2)*delta.t.bar + (v.bar^2)*(delta.t.bar^2))
    }

    delta.t <- (1/(2*(v^2)))*((-1*(sig^2)) + sqrt((sig^4) + (4*(v^2)*(delta.x^2))))
    p <- 0.5 + ((v*delta.t)/(2*delta.x))
    df <- data.frame(i = 0:N,r = r, sig = sig, v = v, delta.t = delta.t, p = p)

    return(df)

}
```

2.Apply one of the numerical methods learned in the previous sections to determine the optimal value

Using bisection method, we get that with delta x equal to 0.059235611, it makes the sum of delta t equal to 1.000002 which is very close to 1.

```
bisecVV(fx, 1, 0.0001)

## [1] 0.05935611

df <- PrintBtTimeVaryingVol(0.05935611)
df

##     i      r   sig         v    delta.t         p
## 1   0 0.0500 0.200 0.0300000 0.08790483 0.5222146
## 2   1 0.0505 0.201 0.0302995 0.08703235 0.5222137
## 3   2 0.0510 0.202 0.0305980 0.08617281 0.5222110
## 4   3 0.0515 0.203 0.0308955 0.08532598 0.5222065
## 5   4 0.0520 0.204 0.0311920 0.08449159 0.5222004
## 6   5 0.0525 0.205 0.0314875 0.08366941 0.5221927
## 7   6 0.0530 0.206 0.0317820 0.08285920 0.5221833
## 8   7 0.0535 0.207 0.0320755 0.08206072 0.5221724
## 9   8 0.0540 0.208 0.0323680 0.08127375 0.5221600
## 10  9 0.0545 0.209 0.0326595 0.08049807 0.5221462
## 11 10 0.0550 0.210 0.0329500 0.07973346 0.5221310
## 12 11 0.0555 0.211 0.0332395 0.07897971 0.5221144

sum(df$delta.t)

## [1] 1.000002
```

Code:

```
fx <- function(delta.x){

    N <- 11
    Tm <- 1
    r <- 0.05*(1 + 0.01*(0:N))
    sig <- 0.2*(1 + 0.005*(0:N))
    v <- r - (0.5*(sig^2))

    sig.bar <- mean(sig)
```

```
    delta.t.bar <- Tm/N
    v.bar <- mean(v)

    delta.t <- (1/(2*(v^2)))*((-1*(sig^2)) + sqrt((sig^4) + (4*(v^2)*(delta.x^2))))

    return(sum(delta.t)-1)
}
bisecVV <- function(fx, a, b, tol = 10e-6){
    n = 1
    while(abs(fx(b) - fx(a)) > tol){

        if(fx(b)*fx(a) > 0){c <- 0; break;}
        c = (a+b)/2
        if(fx(c)*fx(a) < 0){
            b <- c
        }else if(fx(b)*fx(c) < 0){
            a <- c
        }
        if(n > 1000){break;}
        n <- n + 1
    }
    return(c)
}
```