621_Homework 2

Liting Hu

Contents

Question 1

a)

Construct function to calculate option values using an additive binomial tree:

```
Option_BT <- function(isCall=T, isAmerican=F, S0=100, K=100,
                      Tm=1, sigma=0.2, r=0.0075, N=500) {
    # The additive binomial tree
    dt <- Tm/N
    nu <- r - 0.5*sigma^2
    dx <- sqrt(sigma^2*dt + (nu*dt)^2)
    p <- 0.5 + 0.5*(nu*dt/dx)

    disc <- exp(-r*dt)
    St <- rep(0, N + 1)
    C <- rep(0, N + 1)
    P <- rep(0, N + 1)
    St[1] <- S0*exp(-N*dx)
    for (i in 1:N) {St[i+1] <- St[i]*exp(2*dx)}
    C <- pmax(C, St - K)
    P <- pmax(P, K - St)
    for (i in N:1) {
        for (j in 1:i) {
            C[j] = disc*(p*C[j + 1] + (1 - p)*C[j])
            P[j] = disc*(p*P[j + 1] + (1 - p)*P[j])
            St[j] <- St[j]/exp(-dx)
            if (isAmerican==T) {
                C[j] = max(C[j], St[j] - K)
                P[j] = max(P[j], K - St[j])}
        }
    }
```

```
    if (isCall) ans <- C[1]
    else ans <- P[1]
    return(ans)
}
```

b)

This is BSM formula:

```
Option_BSM <- function(isCall = T, S0=100, K=100, Tm=1, sigma=0.2, r=0.0075, div=0) {
    d1 <- (log(S0/K) + (r - div + sigma^2/2)*Tm)/sigma/sqrt(Tm)
    d2 <- d1 - sigma*sqrt(Tm)
    if (isCall) {p <- S0*exp(-div*Tm)*pnorm(d1) - K*exp(-r*Tm)*pnorm(d2)}
    else {p <- K*exp(-r*Tm)*pnorm(-d2) - S0*exp(-div*Tm)*pnorm(-d1)}
    return(p)
}
```

Then download options data and stock data from yahoo finance:

```
setwd("/Users/apple/Desktop/621/HW2")
readfile <- try(read.csv("Calls.csv"), TRUE)
if (inherits(readfile, "try-error")) {
    # Get option chains from Yahoo finance
    AAPL.OPTS <- getOptionChain("AAPL", NULL)
    C1 <- AAPL.OPTS$Apr.21.2017$calls
    P1 <- AAPL.OPTS$Apr.21.2017$puts
    C1$Ave.Price <- (C1$Bid + C1$Ask)/2
    P1$Ave.Price <- (P1$Bid + P1$Ask)/2
    C1 <- C1[, c(1, 4, 5, 8)]
    P1 <- P1[, c(1, 4, 5, 8)]
    C2 <- AAPL.OPTS$May.19.2017$calls
    P2 <- AAPL.OPTS$May.19.2017$puts
    C2$Ave.Price <- (C2$Bid + C2$Ask)/2
    P2$Ave.Price <- (P2$Bid + P2$Ask)/2
    C2 <- C2[, c(1, 4, 5, 8)]
    P2 <- P2[, c(1, 4, 5, 8)]
    C3 <- AAPL.OPTS$Jun.16.2017$calls
    P3 <- AAPL.OPTS$May.19.2017$puts
    C3$Ave.Price <- (C3$Bid + C3$Ask)/2
    P3$Ave.Price <- (P3$Bid + P3$Ask)/2
    C3 <- C3[, c(1, 4, 5, 8)]
    P3 <- P3[, c(1, 4, 5, 8)]

    temp <- merge(C1, C2, by = "Strike")
    calls <- merge(temp, C3, by = "Strike")
    temp <- merge(P1, P2, by = "Strike")
    puts <- merge(temp, P3, by = "Strike")
    coln <- c("Strike", "Bid1", "Ask1", "Apr.21.2017", "Bid2", "Ask2",
              "May.19.2017", "Bid3", "Ask3", "Jun.16.2017")
    colnames(calls) <- coln
    colnames(puts) <- coln
    calls <- calls[3:22, ]
    puts <- puts[-19, ]
    print(calls)
```

```
    print(puts)
    write.csv(calls, file = "Calls.csv", row.names = F)
    write.csv(puts, file = "Puts.csv", row.names = F)
} else {
    calls <- read.csv("Calls.csv")
    puts <- read.csv("Puts.csv")
}

todaystock <- getQuote("AAPL")
# S_0 <- todaystock[, 2] # The value of underlying
# When the option data are downloaded, the stock price is 141.205
S_0 <- 141.205
```

As what we do in HW1, calculate the implied vols for all strikes and all maturities. I will skip this part but just show the result.

Implied vols are showed below:

```
knitr::kable(IV.Calls, caption = "Implied vols for Call Option")
```

Table 1: Implied vols for Call Option

Strike	Apr.21.2017	May.19.2017	Jun.16.2017
85	NaN	NaN	NaN
90	NaN	NaN	NaN
95	NaN	NaN	NaN
100	NaN	NaN	NaN
105	NaN	NaN	0.2377950
110	NaN	NaN	0.2158607
115	NaN	0.2647837	NaN
120	0.1983132	0.2103294	0.2020325
125	0.1921144	0.2047982	0.1909700
130	0.1716106	0.1937357	0.1853434
135	0.1323196	0.1818148	0.1768558
140	0.1265977	0.1737087	0.1738994
145	0.1245950	0.1698940	0.1718967
150	0.1362297	0.1709431	0.1725643
155	0.1561613	0.1755207	0.1749485
160	0.1850573	0.1841990	0.1795260
165	0.2074684	0.1940218	0.1855341
170	0.2286398	0.2051796	0.1906839
175	0.2496204	0.2139533	0.1986947
180	0.3062681	0.2417050	0.2083267

```
knitr::kable(IV.PUTS, caption = "Implied vols for Put Option")
```

Table 2: Implied vols for Put Option

Strike	Apr.21.2017	May.19.2017	Jun.16.2017
65	0.6047652	0.4346314	0.3665397
70	0.4767834	0.3891416	0.3282024
75	0.4230920	0.3719756	0.3137067
80	0.3719756	0.3280117	0.2767045

Strike	Apr.21.2017	May.19.2017	Jun.16.2017
85	0.3230526	0.2651651	0.2237761
90	0.2759416	0.2342664	0.1977410
95	0.2464733	0.2016511	0.1702755
100	0.1997437	0.1771419	0.1496763
105	0.1615018	0.1491995	0.1261208
110	0.1190637	0.1167749	0.0988460
115	0.0771978	0.0810125	0.0688056
120	0.0230297	0.0343783	0.0298007
125	NaN	NaN	NaN
130	NaN	NaN	NaN
135	NaN	NaN	NaN
140	NaN	NaN	NaN
150	NaN	NaN	NaN
155	NaN	NaN	NaN
170	NaN	NaN	NaN
200	2.5983060	NaN	NaN
P.S. Sorr	y about losing	put options d	ata. When I noticed that I could just donwload them beyond trading time.

Using these vols, calculate option prices by Binomial Tree and BSM:

```
Calls.Prices <- merge(IV.Calls, IV.Calls, by = "Strike") # Define data frames for options prices
Puts.Prices <- merge(IV.PUTS, IV.PUTS, by = "Strike")
coln2 <- c("Strike", "Apr.21.2017.BT", "Apr.21.2017.BSM", "May.19.2017.BT",
           "May.19.2017.BSM", "Jun.16.2017.BT", "Jun.16.2017.BSM")
colnames(Calls.Prices) <- coln2
colnames(Puts.Prices) <- coln2
coln3 <- c("Strike", "Apr.21.2017.A", "May.19.2017.A", "Jun.16.2017.A")
Calls.A <- IV.Calls # for American Options
Puts.A <- IV.PUTS
colnames(Calls.A) <- coln3
colnames(Puts.A) <- coln3

Strike.C <- calls[, 1]
Strike.P <- puts[, 1]

for(i in 1:20) {
    for(j in 1:3) {
        # Calculate option prices
        sigma.C <- IV.Calls[i, j+1]
        sigma.P <- IV.PUTS[i, j+1]
        Calls.Prices[i, 2*j] <- Option_BT(T, F, S_0, Strike.C[i], tau[j], sigma.C, r, 200)
        Calls.Prices[i, 2*j+1] <- Option_BSM(T, S_0, Strike.C[i], tau[j], sigma.C, r)
        Puts.Prices[i, 2*j] <- Option_BT(F, F, S_0, Strike.C[i], tau[j], sigma.P, r, 200)
        Puts.Prices[i, 2*j+1] <- Option_BSM(F, S_0, Strike.C[i], tau[j], sigma.P, r)
        Calls.A[i, j+1] <- Option_BT(T, T, S_0, Strike.C[i], tau[j], sigma.C, r, 200)
        Puts.A[i, j+1] <- Option_BT(F, T, S_0, Strike.C[i], tau[j], sigma.P, r, 200)
    }
}
```

Option prices calculated as below:

```
knitr::kable(Calls.Prices, caption = "Call option prices")
```

Table 3: Call option prices

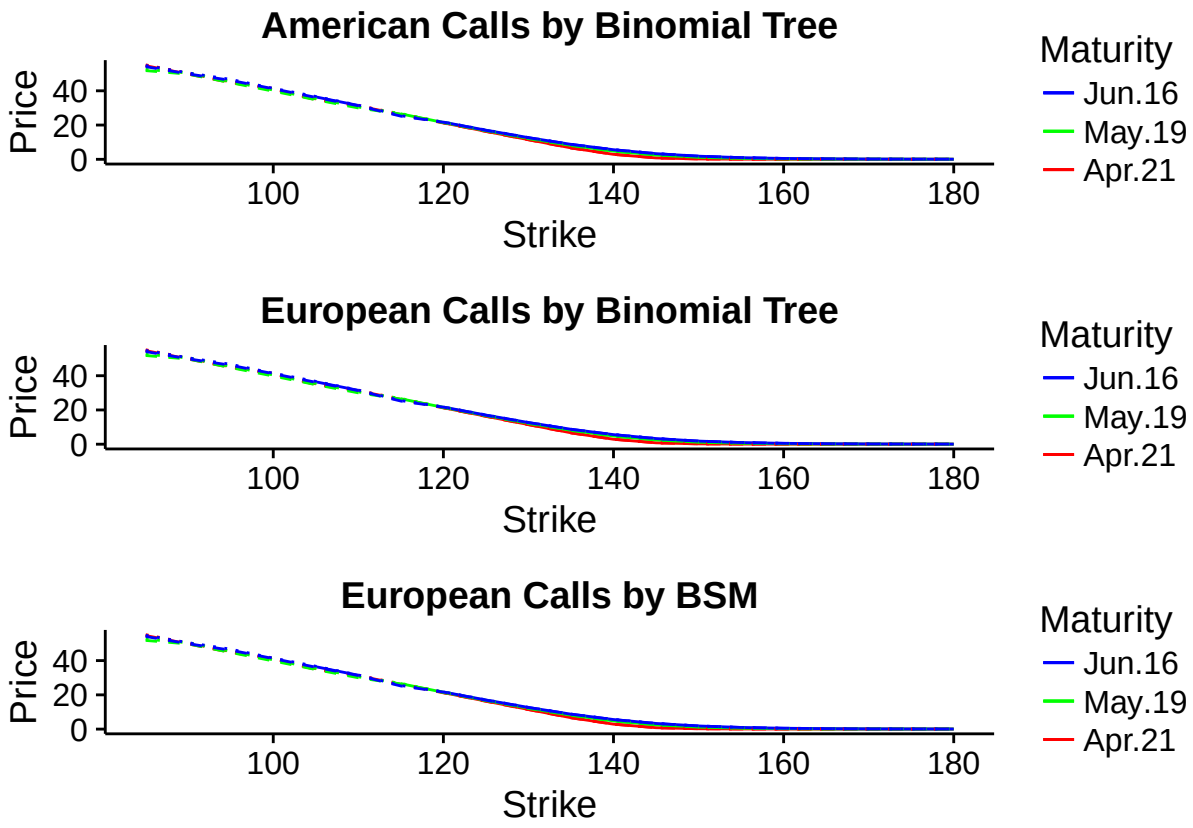
Strike	Apr.21.2017.BT	Apr.21.2017.BSM	May.19.2017.BT	May.19.2017.BSM	Jun.16.2017.BT	Jun.16.2017.BSM
85	NaN	NaN	NaN	NaN	NaN	NaN
90	NaN	NaN	NaN	NaN	NaN	NaN
95	NaN	NaN	NaN	NaN	NaN	NaN
100	NaN	NaN	NaN	NaN	NaN	NaN
105	NaN	NaN	NaN	NaN	36.4245050	36.4249476
110	NaN	NaN	NaN	NaN	31.4495726	31.4499050
115	NaN	NaN	26.5231789	26.5248273	NaN	NaN
120	21.2997548	21.2999747	21.4982929	21.4997943	21.7002222	21.6999983
125	16.3496497	16.3498322	16.7474725	16.7497934	16.9974505	16.9997264
130	11.4731474	11.4747221	12.2245760	12.2245537	12.6521306	12.6498457
135	6.6745536	6.6745310	8.1243695	8.1240977	8.7284509	8.7242033
140	2.9071621	2.9045854	4.8001588	4.7996343	5.6031569	5.5991185
145	0.8242187	0.8248432	2.5030676	2.4992795	3.3011906	3.2998510
150	0.2136965	0.2144368	1.1847667	1.1888044	1.8194331	1.8248673
155	0.0746342	0.0748951	0.5412111	0.5447718	0.9666646	0.9646598
160	0.0446715	0.0449278	0.2646097	0.2644634	0.5063430	0.5090990
165	0.0244507	0.0249191	0.1346654	0.1349466	0.2714231	0.2743525
170	0.0147570	0.0149970	0.0741648	0.0747919	0.1431181	0.1446342
175	0.0097388	0.0099726	0.0387873	0.0398822	0.0844326	0.0849895
180	0.0243213	0.0249486	0.0436652	0.0449982	0.0544061	0.0549075

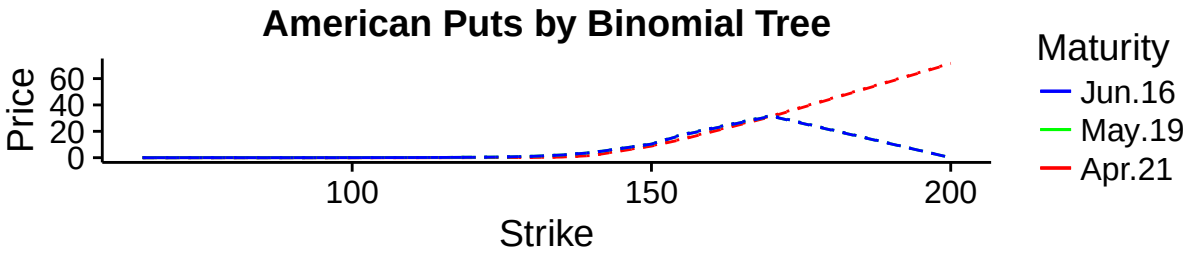
```
knitr::kable(Puts.Prices, caption = "Put option prices")
```

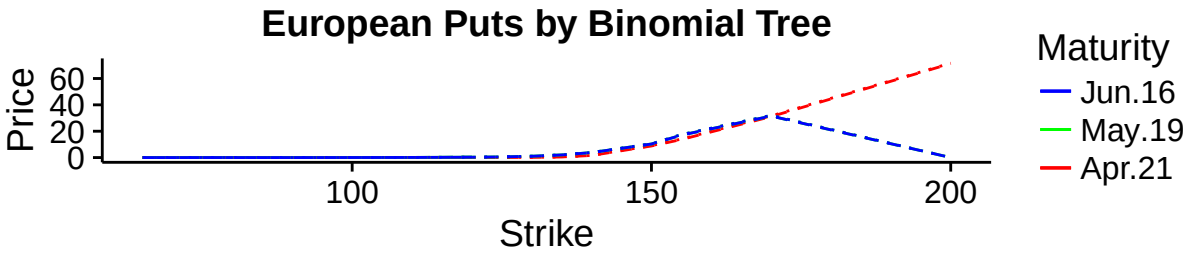
Table 4: Put option prices

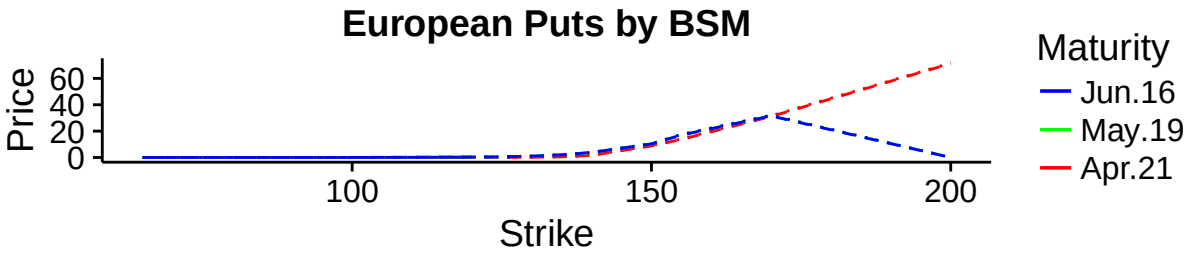
Strike	Apr.21.2017.BT	Apr.21.2017.BSM	May.19.2017.BT	May.19.2017.BSM	Jun.16.2017.BT	Jun.16.2017.BSM
65	0.0194649	0.0199947	0.0145866	0.0149816	0.0146201	0.0149977
70	0.0048334	0.0049986	0.0146668	0.0149667	0.0146814	0.0149764
75	0.0048319	0.0049867	0.0289814	0.0299751	0.0289057	0.0299411
80	0.0048001	0.0049879	0.0293621	0.0299356	0.0293952	0.0299278
85	0.0047203	0.0049932	0.0144893	0.0149635	0.0145497	0.0149815
90	0.0048290	0.0049826	0.0192997	0.0199498	0.0192566	0.0199579
95	0.0096414	0.0099934	0.0244925	0.0249269	0.0244548	0.0249342
100	0.0097823	0.0099556	0.0446852	0.0449281	0.0447387	0.0449754
105	0.0147737	0.0149970	0.0742451	0.0748472	0.0742691	0.0747252
110	0.0195418	0.0199104	0.1133570	0.1146827	0.1134604	0.1144392
115	0.0347317	0.0348401	0.1846984	0.1844090	0.1841433	0.1840145
120	0.0449931	0.0448645	0.2946222	0.2944115	0.2940656	0.2943508
125	NaN	NaN	NaN	NaN	NaN	NaN
130	NaN	NaN	NaN	NaN	NaN	NaN
135	NaN	NaN	NaN	NaN	NaN	NaN
140	NaN	NaN	NaN	NaN	NaN	NaN
150	NaN	NaN	NaN	NaN	NaN	NaN
155	NaN	NaN	NaN	NaN	NaN	NaN
170	NaN	NaN	NaN	NaN	NaN	NaN
200	71.4740147	71.4746237	NaN	NaN	NaN	NaN

c)


American Calls by Binomial Tree
Price
40
20
0
100
120
140
160
180
Strike
Maturity
Jun.16
May.19
Apr.21
European Calls by Binomial Tree
Price
40
20
0
100
120
140
160
180
Strike
Maturity
Jun.16
May.19
Apr.21
European Calls by BSM
Price
40
20
0
100
120
140
160
180
Strike
Maturity
Jun.16
May.19
Apr.21





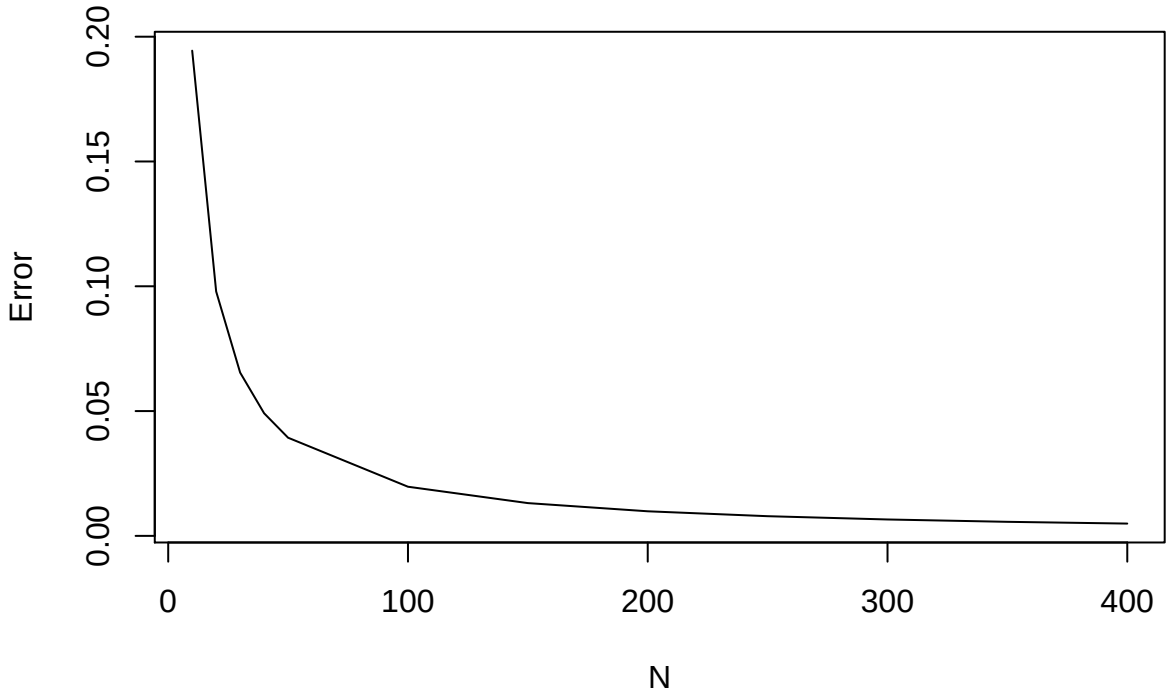


From charts and plots, results calculated by binomial tree and BSM model are similar, even with Amrican ones.

d)

```
Error.by.N <- function(N) {
    # error for European Put
    epsilon <- abs(Option_BSM(isCall = F) - Option_BT(isCall = F, N = N))
    return(epsilon)
}

N <- c(10, 20, 30, 40, 50, 100, 150, 200, 250, 300, 350, 400)
Error <- sapply(N, Error.by.N)
plot(N, Error, type = "l")
```



From this plot we can see that as N goes up, the error is decreasing and approaching a very small number (near zero).

Question 2

a)

Implement a trinomial tree to price European, American Call and Put options:

```
Option_TT <- function(isCall=T, isAmerican=F, S0=100, K=100,
                      Tm=1, sigma=0.25, r=0.06, div=0.03, N=200) {
    # The Trinomial Tree
    dt <- Tm/N
    nu <- r - div - 0.5*sigma^2
    dx <- sigma*sqrt(3*dt)
    edx <- exp(dx)

    pu = 0.5*((sigma^2*dt + nu^2*dt^2)/dx^2 + nu*dt/dx)
    pm = 1.0 - (sigma^2*dt + nu^2*dt^2)/dx^2
    pd = 0.5*((sigma^2*dt + nu^2*dt^2)/dx^2 - nu*dt/dx)

    disc <- exp(-r*dt) # discount factor
    St <- rep(0, 2*N + 1)
    Ct <- rep(0, 2*N + 1)
    Pt <- rep(0, 2*N + 1)
    St[1] <- S0*exp(-N*dx)
    for (i in 1:(2*N)) {St[i+1] <- St[i]*edx}
    Ct <- pmax(Ct, St - K)
    Pt <- pmax(Pt, K - St)

    temp <- matrix(rep(0, (N+1)*(2*N+1)), ncol = (N + 1))
```

```
    C <- temp
    P <- temp
    C[, N+1] <- Ct
    P[, N+1] <- Pt

    for (i in N:1) {
        for (j in (N - i + 2):(N + i)) {
            C[j, i] = disc*(pu*C[j+1, i+1] + pm*C[j, i+1] + pd*C[j-1, i+1])
            P[j, i] = disc*(pu*P[j+1, i+1] + pm*P[j, i+1] + pd*P[j-1, i+1])
            St[j] <- St[j]/exp(-dx)
            if (isAmerican==T) {
                C[j, i] = max(C[j, i], St[j] - K)
                P[j, i] = max(P[j, i], K - St[j])}
        }
    }
    if (isCall) {ans <- C[N + 1, 1]
    } else {ans <- P[N + 1, 1]}
    return(ans)
}
```

b)

```
# European & American Calls
Call <- c(Option_TT(), Option_BSM(sigma = 0.25, r = 0.06, div = 0.03),
          Option_TT(isAmerican = T))

# European & American Calls Puts
Put <- c(Option_TT(isCall = F), Option_BSM(isCall = F, sigma = 0.25, r = 0.06, div = 0.03),
         Option_TT(isCall = F, isAmerican = T))
df <- data.frame(Call, Put)
rownames(df) <- c('Trinomial Tree', "Black-Scholes", "Trinomial Tree American")
```

```
knitr::kable(df, caption = "Pricing by trinomial tree")
```

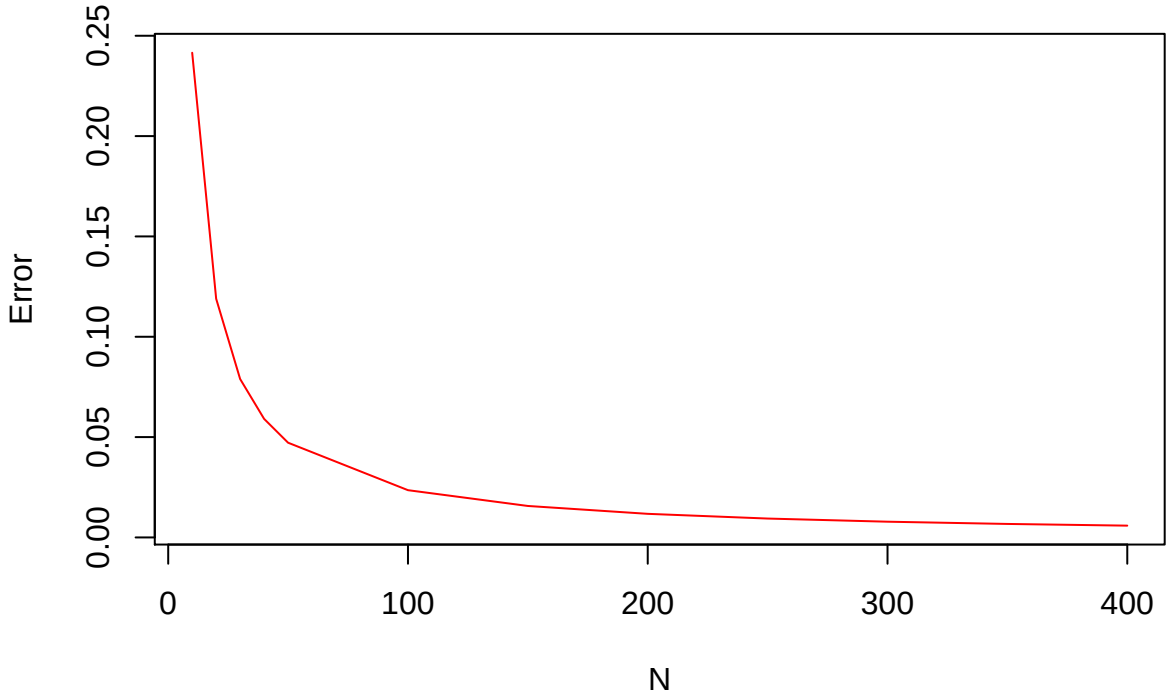
Table 5: Pricing by trinomial tree

	Call	
Trinomial Tree	11.00132	
Black-Scholes	11.01308	
Trinomial Tree American	45556.19316	
Using given parameters, th	e results for	European Call and Put are very close between trinomial tree method and BSM

```
Error.by.N <- function(N) {
    # error for European Put
    # for trinomial tree
    epsilon <- abs(Option_BSM(isCall = F, sigma = 0.25, r=0.06, div = 0.03) -
                       Option_TT(isCall = F, N = N))
    return(epsilon)
}

N <- c(10, 20, 30, 40, 50, 100, 150, 200, 250, 300, 350, 400)
```

```
Error <- sapply(N, Error.by.N)
plot(N, Error, type = "l", col = "red")
```



Similar to binomial tree senario, from this plot we can see that as N goes up, the error is decreasing and approaching zero.

Question 3

a)

```
Option_UPandOUT <- function(S0=10, K=10, Tm=0.3, sigma=0.2, r=0.01, div=0, H=11, N=500) {
    # European Up-and Out
    # Binomial tree
    dt <- Tm/N
    nu <- r - div - 0.5*sigma^2
    dx <- sqrt(sigma^2*dt + (nu*dt)^2)
    p <- 0.5 + 0.5*(nu*dt/dx)

    disc <- exp(-r*dt)
    St <- rep(0, N + 1)
    C <- rep(0, N + 1)
    St[1] <- S0*exp(-N*dx)
    for (i in 1:N) {St[i+1] <- St[i]*exp(2*dx)}
    C <- pmax(C, St - K)
    C[C >= H - K] = 0

    for (i in N:1) {
        for (j in 1:i) {
            St[j] <- St[j]/exp(-dx)
            if (St[j] >= H) {C[j] = 0
```

```
            } else {C[j] = disc*(p*C[j + 1] + (1 - p)*C[j])}
        }
    }
    ans <- C[1]
    return(ans)
}
Option_UPandOUT(N=1000)
```

```
## [1] 0.05578389
```

The price of the European Up-and-Out call option is 0.0549897.

b)

```
Option_UPandOUT_BS <- function(S0=10, K=10, Tm=0.3, sigma=0.2, r=0.01, div=0, H=11) {
    # European Up-and-Out
    # Explicit formulas
    nu <- r - div - 0.5*sigma^2
    d_sh <- (log(S0/H) + nu*Tm)/sigma/sqrt(Tm)
    d_hs <- (log(H/S0) + nu*Tm)/sigma/sqrt(Tm)
    C_bs <- Option_BSM(T, S0, K, Tm, sigma, r, div)
    C_bs.h <- Option_BSM(T, S0, H, Tm, sigma, r, div)
    C_bs.hs <- Option_BSM(T, H^2/S0, K, Tm, sigma, r, div)
    C_bs.h.hs <- Option_BSM(T, H^2/S0, H, Tm, sigma, r, div)
    ans <- C_bs - C_bs.h - (H - K)*exp(-r*Tm)*pnorm(d_sh) -
        (H/S0)^(2*nu/sigma^2)*(C_bs.hs - C_bs.h.hs - (H - K)*exp(-r*Tm)*pnorm(d_hs))
    return(ans)
}
Option_UPandOUT_BS()
```

```
## [1] 0.0530928
```

The price of European Up-and-Out is 0.053.

c)

To price an European Up-and-In call option by analytical solution:

```
# The analytical solution
Option_UPandIn <- function(S0=10, K=10, Tm=0.3, sigma=0.2, r=0.01, div=0, H=11) {
    nu <- r - div - 0.5*sigma^2
    d_sh <- (log(S0/H) + nu*Tm)/sigma/sqrt(Tm)
    d_hs <- (log(H/S0) + nu*Tm)/sigma/sqrt(Tm)
    C_bs.h <- Option_BSM(T, S0, H, Tm, sigma, r)
    P_bs.hs <- Option_BSM(F, H^2/S0, K, Tm, sigma, r)
    P_bs.h.hs <- Option_BSM(F, H^2/S0, H, Tm, sigma, r)
    ans <- C_bs.h + (H - K)*exp(-r*Tm)*pnorm(d_sh) +
        (H/S0)^(2*nu/sigma^2)*(P_bs.hs - P_bs.h.hs + (H - K)*exp(-r*Tm)*pnorm(-d_hs))
    return(ans)
}
Option_UPandIn()
```

```
## [1] 0.3981948
```

The price of European Up-and-In is 0.398 by the analytical solution.

By The In-Out parity:

```
# The In-Out parity
# UI_BS + UO_BS = C_BS
Option_BSM(T, 10, 10, 0.3, 0.2, 0.01) - Option_UPandOUT_BS()
```

```
## [1] 0.3981948
```

The price of European Up-and-In is 0.398 by the In-Out parity which is nearly equal to the former one.

d)

```
American_UpandIn <- function(S0=10, K=10, Tm=0.3, sigma=0.2, r=0.01, div=0, H=11) {
    nu <- r - div - 0.5*sigma^2
    d_bs <- (log(H^2/S0/min(H, K)) + nu*Tm)/sigma/sqrt(Tm)
    d_sh <- (log(S0/H) + nu*Tm)/sigma/sqrt(Tm)
    C_bs <- Option_BSM(T, S0, K, Tm, sigma, r)
    C_bs.h <- Option_BSM(T, S0, H, Tm, sigma, r)
    P_bs.h.hs <- Option_BSM(F, H^2/S0, min(H, K), Tm, sigma, r, div)
    ans <- (H/S0)^(2*nu/sigma^2)*(P_bs.h.hs - (min(H, K) - K)*exp(-r*Tm)*pnorm(-d_bs))
    if (K > H) {ans <- ans + C_bs - C_bs.h - (H - K)*exp(-r*Tm)*pnorm(d_sh)}
    return(ans)
}
American_UpandIn()
```

```
## [1] 0.0177448
```