Homework2

Problem 1

Question(a)

```
#Binomial Tree: j Times, j+1 nodes
Bino.tree <- function(isCall, isAmerican, K, T, S0, r, sig, N)
{
  # Precompute constants
  dt = T/N
  u = exp((r-1/2*sig^2)*dt + sig*sqrt(dt))
  d = 1/u
  p = (exp(r*dt)-d)/(u-d) #probabilities
  disc = exp(-r*dt) #discount
  M = N+1 #number of nodes
  cp = ifelse(isCall, 1, -1)

  # Intialize asset prices
  V = S = matrix(0, nrow=M, ncol=M)
  S[1,1] = S0
  #construct a asset binomial tree
  for (j in 2:M) {
      S[1, j] = S[1, j-1]*u
      for(i in 2:j) {
        S[i, j] = S[i-1, j-1]*d
      }
  }

  # Intialize option values at maturity
  for (j in 1:M) {
    V[M-j+1, M] = max( 0, cp * (S[M-j+1, M]-K))
  }

  # Step backwards through the tree
  for (j in (M-1):1) {
    for (i in 1:j) {
        V[i, j] = disc * ( p*V[i, j+1] + (1-p)*V[i+1, j+1] )
        if(isAmerican) {
            V[i, j] = max(V[i, j], cp * (S[i, j] - K))
        }
    }
  }

  # Return the price ----
  V[1,1]
}
```

Question(b)

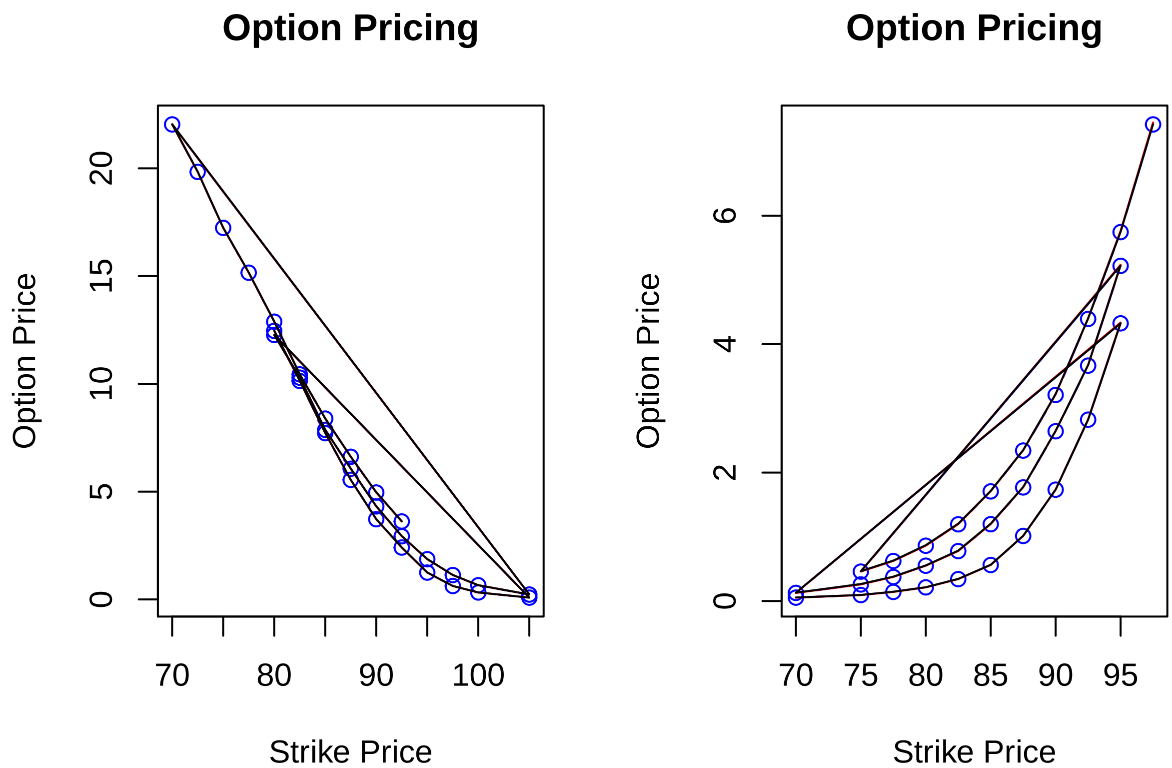
```r
getwd() # get currency work directory
```

```
## [1] "E:/621 computational method/H2"
```

```r
#read data that download from Yahoo finance
JPM1c <- read.csv("JPMCalls 170421.csv")
JPM2c <- read.csv("JPMCalls 170519.csv")
JPM3c <- read.csv("JPMCalls 170616.csv")
JPM1p <- read.csv("JPMPuts 170421.csv")
JPM2p <- read.csv("JPMPuts 170519.csv")
JPM3p <- read.csv("JPMPuts 170616.csv")
#calculate market price
MPrice1c <- matrix((JPM1c$Bid + JPM1c$Ask)/2) #one month
MPrice2c <- matrix((JPM2c$Bid + JPM2c$Ask)/2) #two month
MPrice3c <- matrix((JPM3c$Bid + JPM3c$Ask)/2) #three month
MPrice1p <- matrix((JPM1p$Bid + JPM1p$Ask)/2) #one month
MPrice2p <- matrix((JPM2p$Bid + JPM2p$Ask)/2) #two month
MPrice3p <- matrix((JPM3p$Bid + JPM3p$Ask)/2) #three month

S0 <- 91.41 # Close price of JPM at 2017-03-07
#strike price of JPM option
K1c <- matrix(JPM1c$Strike) #one month
K2c <- matrix(JPM2c$Strike) #two month
K3c <- matrix(JPM3c$Strike) #three month
K1p <- matrix(JPM1p$Strike) #one month
K2p <- matrix(JPM2p$Strike) #two month
K3p <- matrix(JPM3p$Strike) #three month
#calculate time t of each option
tau1 <- as.numeric(difftime("2017-04-21","2017-03-07",units = "days"))/365
tau2 <- as.numeric(difftime("2017-05-19","2017-03-07",units = "days"))/365
tau3 <- as.numeric(difftime("2017-06-16","2017-03-07",units = "days"))/365

# Function to calculate opntion price by BS formular
BS <- function(type, S0, K, tau, r, sigma,div){

  d1 <- (log(S0/K)+(r-div+sigma^2/2)*tau) / (sigma*sqrt(tau))
  d2 <- d1 - sigma*sqrt(tau)
  if(type=="C"){
    Price <- S0*exp(-div*tau)*pnorm(d1) - K*exp(-r*tau)*pnorm(d2)
  }
  if(type=="P"){
    Price <- K*exp(-r*tau)*pnorm(-d2) - S0*exp(-div*tau)*pnorm(-d1)
  }
  return(Price)
}
#Function to calculate error between BS and Market price
err <- function(type,S0,K,r,tau,sig,div,MPrice){
    BS(type,S0,K,tau,r,sig,div) - MPrice
}

#Function to find BS Implied Vol using Bisection Method
Ivol.BS <- function(type,S0, K, r,tau,div, MPrice){
```

```
    sig <- c()
#loop for every strike price and market price
  for(i in 1:10){
    a <- -0.001
    b <- 2
    c <- (a+b)/2
#Loop until that the value of function to sigma is less than tolerance level 1e-4
    while(abs(b-a) > 1e-4){
      fa <- err(type,S0,K[i],r,tau,a,div,MPrice[i])
      fc <- err(type,S0,K[i],r,tau,c,div,MPrice[i])
      if( fa * fc < 0)
          b <- c
      else
          a <- c
      c <- (a+b)/2
    }
   sig <- c(sig,c)
  }
  return(sig)
}

Imv1c <- Ivol.BS("C",S0,K1c,0.0075,tau1,0,MPrice1c) #one month
Imv2c <- Ivol.BS("C",S0,K2c,0.0075,tau2,0,MPrice2c) #two month
Imv3c <- Ivol.BS("C",S0,K3c,0.0075,tau3,0,MPrice3c) #three month
Imv1p <- Ivol.BS("P",S0,K1p,0.0075,tau1,0,MPrice1p) #one month
Imv2p <- Ivol.BS("P",S0,K2p,0.0075,tau2,0,MPrice2p) #two month
Imv3p <- Ivol.BS("P",S0,K3p,0.0075,tau3,0,MPrice3p) #three month

#European Call&Put
JPMEC1 <- c()
JPMEC2 <- c()
JPMEC3 <- c()
JPMEP1 <- c()
JPMEP2 <- c()
JPMEP3 <- c()
JPMAC1 <- c()
JPMAC2 <- c()
JPMAC3 <- c()
JPMAP1 <- c()
JPMAP2 <- c()
JPMAP3 <- c()
for(i in 1:10){
  a1 <- Bino.tree(isCall=T, isAmerican=F,K1c[i],tau1,S0,0.0075,Imv1c[i],200) #one month
  a2 <- Bino.tree(isCall=T, isAmerican=F,K2c[i],tau2,S0,0.0075,Imv2c[i],200) #two month
  a3 <- Bino.tree(isCall=T, isAmerican=F,K3c[i],tau3,S0,0.0075,Imv3c[i],200) #three month
  b1 <- Bino.tree(isCall=F, isAmerican=F,K1p[i],tau1,S0,0.0075,Imv1p[i],200) #one month
  b2 <- Bino.tree(isCall=F, isAmerican=F,K2p[i],tau2,S0,0.0075,Imv2p[i],200) #two month
  b3 <- Bino.tree(isCall=F, isAmerican=F,K3p[i],tau3,S0,0.0075,Imv3p[i],200) #three month
#American Call&Put
  c1 <- Bino.tree(isCall=T, isAmerican=T,K1c[i],tau1,S0,0.0075,Imv1c[i],200) #one month
  c2 <- Bino.tree(isCall=T, isAmerican=T,K2c[i],tau2,S0,0.0075,Imv2c[i],200) #two month
  c3 <- Bino.tree(isCall=T, isAmerican=T,K3c[i],tau3,S0,0.0075,Imv3c[i],200) #three month
  d1 <- Bino.tree(isCall=F, isAmerican=T,K1p[i],tau1,S0,0.0075,Imv1p[i],200) #one month
```

```r
  d2 <- Bino.tree(isCall=F, isAmerican=T,K2p[i],tau2,S0,0.0075,Imv2p[i],200) #two month
  d3 <- Bino.tree(isCall=F, isAmerican=T,K3p[i],tau3,S0,0.0075,Imv3p[i],200) #three month

  JPMEC1 <- c(JPMEC1,a1)
  JPMEC2 <- c(JPMEC2,a2)
  JPMEC3 <- c(JPMEC3,a3)
  JPMEP1 <- c(JPMEP1,b1)
  JPMEP2 <- c(JPMEP2,b2)
  JPMEP3 <- c(JPMEP3,b3)
  JPMAC1 <- c(JPMAC1,c1)
  JPMAC2 <- c(JPMAC2,c2)
  JPMAC3 <- c(JPMAC3,c3)
  JPMAP1 <- c(JPMAP1,d1)
  JPMAP2 <- c(JPMAP2,d2)
  JPMAP3 <- c(JPMAP3,d3)
}
binomial.JPMEC <- c(JPMEC1,JPMEC2,JPMEC3)
binomial.JPMEP <- c(JPMEP1,JPMEP2,JPMEP3)
binomial.JPMAC <- c(JPMAC1,JPMAC2,JPMAC3)
binomial.JPMAP <- c(JPMAP1,JPMAP2,JPMAP3)

#European Call&Put price by BS method
JPMC1 <- BS("C",S0, K1c, tau1, 0.0075, Imv1c,0)
JPMC2 <- BS("C",S0, K2c, tau2, 0.0075, Imv2c,0)
JPMC3 <- BS("C",S0, K3c, tau3, 0.0075, Imv3c,0)
JPMP1 <- BS("P",S0, K1p, tau1, 0.0075, Imv1p,0)
JPMP2 <- BS("P",S0, K2p, tau2, 0.0075, Imv2p,0)
JPMP3 <- BS("P",S0, K3p, tau3, 0.0075, Imv3p,0)

BS.JPMEC <- c(JPMC1,JPMC2,JPMC3)
BS.JPMEP <- c(JPMP1,JPMP2,JPMP3)
CMarketPrice <- c(MPrice1c,MPrice2c,MPrice3c)
PMarketPrice <- c(MPrice1p,MPrice2p,MPrice3p)
strikec <- c(K1c,K2c,K3c)
strikep <- c(K1p,K2p,K3p)

par(mfrow=c(1,2))
plot(strikec,binomial.JPMEC,typ="b", col="blue", main=c("Option Pricing"),
     xlab="Strike Price", ylab="Option Price")
lines(strikec,binomial.JPMAC,col = "red")
lines(strikec,BS.JPMEC)

plot(strikep,binomial.JPMEP,typ="b", col="blue", main=c("Option Pricing"),
     xlab="Strike Price", ylab="Option Price")
lines(strikep,binomial.JPMAP,col = "red")
lines(strikep,BS.JPMEP)
```




```
#table to compare market price,BS call option price and Euro&Amer call option
df1 <- data.frame(CMarketPrice,BS.JPMEC,binomial.JPMEC,binomial.JPMAC)
df1
```

```
##    CMarketPrice   BS.JPMEC binomial.JPMEC binomial.JPMAC
## 1        12.475 12.4750390    12.45918525    12.45918525
## 2        10.150 10.1499095    10.13512431    10.13512431
## 3         7.725  7.7252499     7.71442722     7.71442722
## 4         5.550  5.5499676     5.54648735     5.54648735
## 5         3.725  3.7249055     3.72205596     3.72205596
## 6         2.415  2.4151293     2.40766086     2.40766086
## 7         1.250  1.2499558     1.24627498     1.24627498
## 8         0.625  0.6250940     0.62380268     0.62380268
## 9         0.325  0.3251587     0.32184300     0.32184300
## 10        0.085  0.0850261     0.08393255     0.08393255
## 11       12.275 12.2750622    12.26831633    12.26831633
## 12       10.300 10.2997105    10.28955499    10.28955499
## 13        7.875  7.8751428     7.86660661     7.86660661
## 14        6.075  6.0746879     6.06201715     6.06201715
## 15        4.325  4.3248650     4.31659955     4.31659955
## 16        2.940  2.9400684     2.93150440     2.93150440
## 17        1.875  1.8749804     1.86648440     1.86648440
## 18        1.135  1.1347514     1.13065732     1.13065732
## 19        0.660  0.6602955     0.65779342     0.65779342
## 20        0.230  0.2299818     0.22609566     0.22609566
## 21       22.050 22.0499402    22.03652069    22.03652069
## 22       19.850 19.8500996    19.83379849    19.83379849
```

```
## 23       17.250 17.2500940    17.23878912    17.23878912
## 24       15.175 15.1747933    15.15754327    15.15754327
## 25       12.900 12.8998514    12.88512124    12.88512124
## 26       10.450 10.4501665    10.44183366    10.44183366
## 27        8.400  8.4000819     8.38883133     8.38883133
## 28        6.625  6.6248329     6.61735118     6.61735118
## 29        4.975  4.9746204     4.95967710     4.95967710
## 30        3.625  3.6251383     3.61699229     3.61699229
```

```r
#table to compare market price,BS put option price and Euro&Amer put option
df2 <- data.frame(PMarketPrice,BS.JPMEP,binomial.JPMEP,binomial.JPMAP)
df2
```

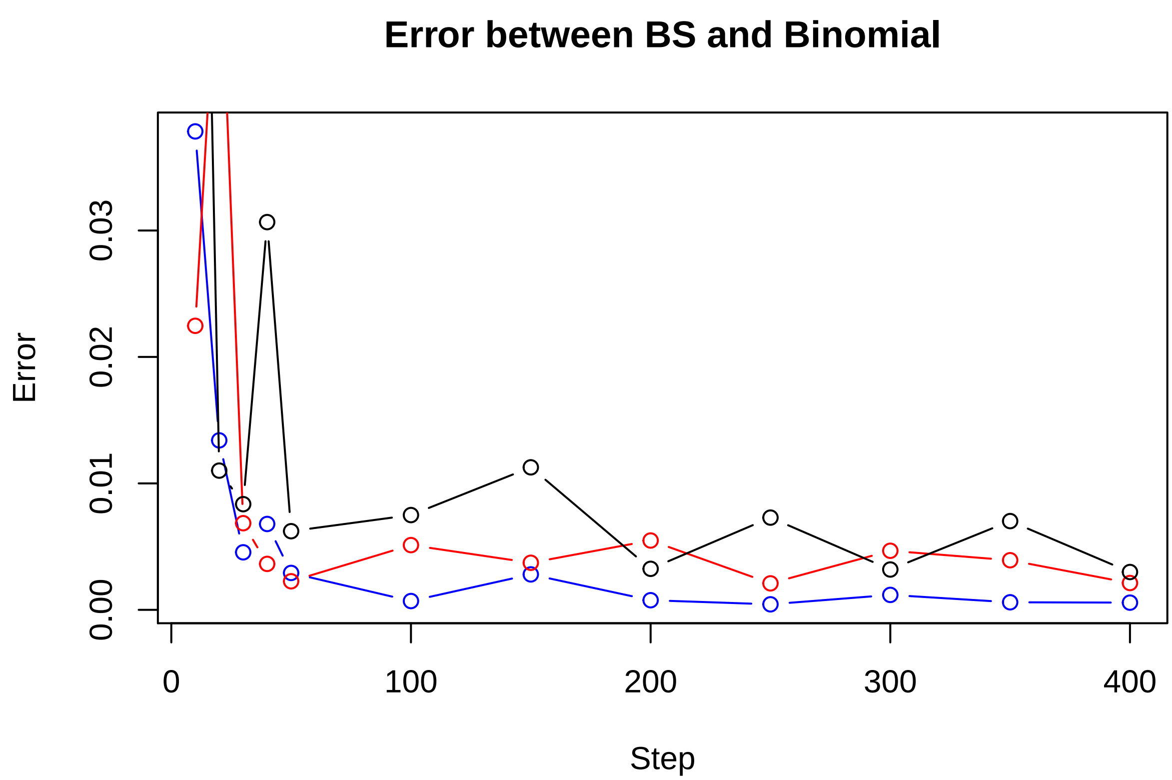
```
##    PMarketPrice   BS.JPMEP binomial.JPMEP binomial.JPMAP
## 1         0.055 0.05499290     0.05297014     0.05299594
## 2         0.095 0.09495186     0.09304797     0.09310770
## 3         0.145 0.14505023     0.14159196     0.14169307
## 4         0.215 0.21489197     0.21270531     0.21289350
## 5         0.345 0.34496555     0.34108844     0.34145975
## 6         0.565 0.56493274     0.56110632     0.56182290
## 7         1.015 1.01495103     1.01358261     1.01500985
## 8         1.735 1.73480347     1.73393756     1.73699953
## 9         2.825 2.82462747     2.82423678     2.83053220
## 10        4.325 4.32514796     4.32438677     4.33722390
## 11        0.130 0.13002211     0.12631579     0.12641428
## 12        0.265 0.26501335     0.25808386     0.25833771
## 13        0.380 0.37999379     0.37466002     0.37509492
## 14        0.555 0.55479450     0.54970538     0.55037387
## 15        0.785 0.78497930     0.77756840     0.77875121
## 16        1.205 1.20504388     1.19632651     1.19826651
## 17        1.775 1.77470919     1.76891851     1.77223229
## 18        2.650 2.65002397     2.64364209     2.64938065
## 19        3.675 3.67499985     3.66618683     3.67611518
## 20        5.225 5.22531093     5.21982686     5.23593061
## 21        0.465 0.46518681     0.45803339     0.45864827
## 22        0.630 0.63002773     0.62311175     0.62401541
## 23        0.870 0.86991407     0.86094015     0.86230923
## 24        1.205 1.20504993     1.19475701     1.19704827
## 25        1.715 1.71509519     1.70692062     1.71038327
## 26        2.350 2.35036898     2.34286968     2.34847401
## 27        3.225 3.22469982     3.20988264     3.21885571
## 28        4.400 4.40054850     4.39278058     4.40621366
## 29        5.750 5.74960676     5.74510802     5.76573990
## 30        7.425 7.42522890     7.42198618     7.45279960
```

Question(c)

Comment: From the table above, we can see both BS price and Binomial price are close to the market price, In European option, the more step that binomial tree set, the more close to the BS price. In call option, binomial tree for European and American can get exactly same data. In put option, American option price can be slightly higher than the European option price.

Question(d)

```
#Function to calculate error between BS price and Binomial price
abs.err1 <- function(S0,K,tau,r,sig,div,N){
  y <-c()
  for(k in 1:12){ #loop for the number of N
    for(i in 1:10){ #loop for the number of data
      a <- abs(BS("P",S0, K[i], tau, r, sig[i],div)-
          Bino.tree(isCall=F, isAmerican=F,K[i],tau,S0,r,sig[i],N[k]))
    }
  y <- c(y,a)
  }
  return(y)
}
N <- c(10,20,30,40,50,100,150,200,250,300,350,400)
err.one <- abs.err1(S0,K1p,tau1,0.0075,Imv1p,0,N)
err.two <- abs.err1(S0,K2p,tau2,0.0075,Imv2p,0,N)
err.three <- abs.err1(S0,K3p,tau3,0.0075,Imv3p,0,N)

plot(N,err.one,typ="b", col="blue",
     main=c("Error between BS and Binomial"),xlab="Step", ylab="Error")
lines(N,err.two,typ="b", col="red")
lines(N,err.three,typ="b", col="black")
```



Bonus(5 point)

```
#Function for error between Binomial tree and market price
err.bino <- function(isCall,S0,K,r,tau,sig,N,MPrice){
  Bino.tree(isCall, isAmerican=F,K,tau,S0,r,sig,N) - MPrice
}

#Function to find Binomial Implied Vol using Bisection Method
Ivol.Bino <- function(isCall, S0, K, tau, r, N, MPrice){
  sig <- c()
#loop for every strike price and market price
  for(i in 1:10){
    a <- -0.01
    b <- 2
    c <-(a+b)/2
#Loop until that the value of function to sigma is less than tolerance level 1e-4
    while(abs(b-a) > 1e-4){
      fb <- err.bino (isCall,S0,K[i],r,tau,b,N,MPrice[i])
      fc <- err.bino (isCall,S0,K[i],r,tau,c,N,MPrice[i])
      #print(fa*fc)
      if( fb * fc < 0 )
          a <- c
      else
          b <- c
    c <- (a+b)/2
    }
   sig <- c(sig,c)
  }
  return(sig)
}

IV1c <- Ivol.Bino(isCall = T,S0,K1c,tau1,0.0075,100,MPrice1c)
IV2c <- Ivol.Bino(isCall = T,S0,K2c,tau2,0.0075,100,MPrice2c)
IV3c <- Ivol.Bino(isCall = T,S0,K3c,tau3,0.0075,100,MPrice3c)
IV1p <- Ivol.Bino(isCall = F,S0,K1p,tau1,0.0075,100,MPrice1p)
IV2p <- Ivol.Bino(isCall = F,S0,K2p,tau2,0.0075,100,MPrice2p)
IV3p <- Ivol.Bino(isCall = F,S0,K3p,tau3,0.0075,100,MPrice3p)

#compare BS Ivol and Binomial Ivol in a table
DF.Ivol <- data.frame(c(Imv1c,Imv2c,Imv3c),c(IV1c,IV2c,IV3c),c(Imv1p,Imv2p,Imv3p),
                      c(IV1p,IV2p,IV3p))
DF.Ivol
```

```
##    c.Imv1c..Imv2c..Imv3c. c.IV1c..IV2c..IV3c. c.Imv1p..Imv2p..Imv3p.
## 1               0.3875303           0.3904604              0.3540053
## 2               0.3472270           0.3490556              0.2977638
## 3               0.2906802           0.2912730              0.2763908
## 4               0.2526973           0.2529967              0.2526363
## 5               0.2299198           0.2310368              0.2325457
## 6               0.2246071           0.2249641              0.2130657
## 7               0.2024403           0.2026363              0.2017075
## 8               0.1962116           0.1963795              0.1897386
## 9               0.1995702           0.1998146              0.1777698
## 10              0.2093407           0.2099971              0.1634804
```

```
## 11           0.2774289          0.2785143              0.3182208
## 12           0.2829249          0.2850777              0.2848179
## 13           0.2374920          0.2379070              0.2685134
## 14           0.2327899          0.2339198              0.2535523
## 15           0.2164854          0.2170512              0.2362707
## 16           0.2072034          0.2074208              0.2265612
## 17           0.1998145          0.2003667              0.2147756
## 18           0.1953567          0.1956435              0.2083026
## 19           0.1934026          0.1939873              0.1938911
## 20           0.1985932          0.1993852              0.1934026
## 21           0.3543717          0.3580113              0.2765130
## 22           0.3585852          0.3621825              0.2617961
## 23           0.3062519          0.3088164              0.2487891
## 24           0.3117478          0.3148891              0.2365760
## 25           0.2890314          0.2904143              0.2282100
## 26           0.2492166          0.2507884              0.2174014
## 27           0.2346830          0.2361281              0.2090354
## 28           0.2279657          0.2288899              0.2042723
## 29           0.2172182          0.2176646              0.1954177
## 30           0.2106842          0.2109172              0.1907157
##    c.IV1p..IV2p..IV3p.
## 1            0.3567232
## 2            0.2992473
## 3            0.2774101
## 4            0.2533647
## 5            0.2332451
## 6            0.2137389
## 7            0.2024522
## 8            0.1900615
## 9            0.1783455
## 10           0.1633784
## 11           0.3204710
## 12           0.2864885
## 13           0.2702333
## 14           0.2550209
## 15           0.2373549
## 16           0.2269270
## 17           0.2152110
## 18           0.2088930
## 19           0.1940486
## 20           0.1937419
## 21           0.2783302
## 22           0.2640379
## 23           0.2505431
## 24           0.2379070
## 25           0.2297487
## 26           0.2184621
## 27           0.2094450
## 28           0.2044765
## 29           0.1961342
## 30           0.1911656
```

Comment: From the table above, the first column is implied volatility from Call option by BS method, the second column is implied vol from call option by binomial tree. the third column is put option implied vol by

BS method. the last column is put option implied vol by binomial method. We can see that the results of BS and Binomial implied volatility that calculated by Biasection method is very similar.

Problem 2

Question(a)

```
# Trinomial Tree: j times, 2*j+1 final nodes
Tri.tree <- function(isCall, isAmerican, K, T, S0, r, sig, N, div) {

  # Precompute constants
  dt = T/N
  nu = r - div - 0.5 * sig^2
  dx=sqrt(sig^2*dt+nu*dt^2)
  pu = 0.5 * ( (sig^2*dt + nu^2 *dt^2)/dx^2 + nu*dt/dx )
  pm = 1.0 -   (sig^2*dt + nu^2 *dt^2)/dx^2
  pd = 0.5 * ( (sig^2*dt + nu^2 *dt^2)/dx^2 - nu*dt/dx )
  disc = exp(-r*dt) # discount
  firstRow = 1
  r = nRows = lastRow = 2*N+1
  firstCol = 1
  middleRow = s = nCols = lastCol = N+1
  cp = ifelse(isCall, 1, -1)

  # Intialize asset prices
  V = S = matrix(0, nrow=nRows, ncol=nCols)
  S[middleRow, firstCol] = S0
  for (j in 1:(nCols-1)) {
    for(i in (middleRow-j+1):(middleRow+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)
    }
  }
  # Intialize option values at maturity
  for (i in 1:nRows) {
    V[i, lastCol] = max( 0, cp * (S[i, lastCol]-K))
  }
  # Step backwards through the tree
  for (j in (nCols-1):1) {
    for(i in (nCols-j+1):(nCols+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))
      }
    }
  }
  # Return the price
  V[middleRow,firstCol]
}
```

Question(b)

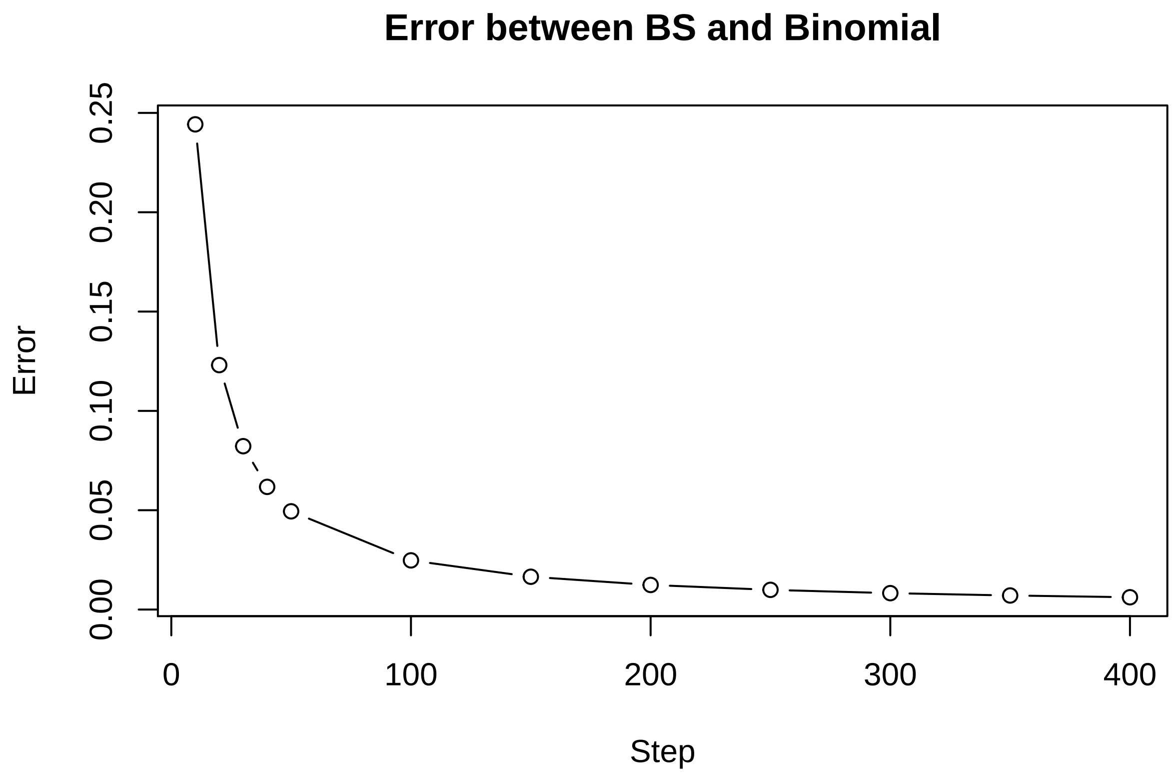
```
#European option
trinomial.JPMEC <- Tri.tree(isCall=T,isAmerican=F,100,1,100,0.06,0.25,200,0.03)
trinomial.JPMEP <- Tri.tree(isCall=F,isAmerican=F,100,1,100,0.06,0.25,200,0.03)
BS.JPMEC <- BS("C",100,100,1, 0.06,0.25,0.03)
BS.JPMEP <- BS("P",100,100,1, 0.06,0.25,0.03)
#American option
trinomial.JPMAC <- Tri.tree(isCall=T,isAmerican=T,100,1,100,0.06,0.25,200,0.03)
trinomial.JPMAP <- Tri.tree(isCall=F,isAmerican=T,100,1,100,0.06,0.25,200,0.03)
#present results in table
DF.tric <- data.frame(BS.JPMEC,trinomial.JPMEC,trinomial.JPMAC)
DF.trip <- data.frame(BS.JPMEP,trinomial.JPMEP,trinomial.JPMAP)
#call option compare
DF.tric
```

```
##   BS.JPMEC trinomial.JPMEC trinomial.JPMAC
## 1 11.01308        11.00055        11.00069
```

```
#put option compare
DF.trip
```

```
##   BS.JPMEP trinomial.JPMEP trinomial.JPMAP
## 1 8.144979        8.132595        8.505161
```

```
abs.err.tri <- function(S0,K,tau,r,sig,div,N){
  y <-c()
  for(i in 1:12){
  a <- abs(BS("P",S0, K, tau, r, sig,div)-
         Tri.tree(isCall=F, isAmerican=F,K,tau,S0,r,sig,N[i],div))
  y <- c(y,a)
  }
  return(y)
}
N <- c(10, 20, 30, 40, 50, 100, 150, 200, 250, 300, 350,400)
err2 <- abs.err.tri(100,100,1,0.06,0.25,0.03,N)
plot(N,err2,typ="b", col="black", main=c("Error between BS and Binomial"),
     xlab="Step", ylab="Error")
```



Problem 3

Question(a)

```
#Using binomial tree method to price an European Up-and-Out Call option
Bino.Barrier.UO <- function(K,T,S,sig,r,H,N){

  #precompute constants
  dt = T/N
  nu = r-0.5*sig^2
  dx = sqrt(sig^2*dt + (nu*dt)^2)
  p = 1/2 + 1/2*(nu*dt/dx)
  disc = exp(-r*dt)

  #initialise asset prices at maturity N
  St = array(0,dim = c(N+1))
  St[1] <- S * exp(-dx*N)
  St[2] <- St[1]*exp(2*(dx))
  for(j in 2:N+1){
    St[j] = St[j-1]*exp(2*(dx))
  }
  #initialise option values at maturity
  C = array(0,dim = c(N+1))
```

```
    for(j in 1:N+1){
    if(St[j]<H)
      C[j] <- max(0.0,St[j] - K)
    else
      C[j] <- 0.0
  }
  #step back through the tree applying the barrier and early exercise condition
  for(i in N:1){
    for(j in 1:i){
      St[j] = St[j]*exp(dx)
      if(St[j]<H){
        C[j] = disc*(1-p)*C[j] + disc*p*C[j+1]
      }
      else
        C[j] = 0.0
    }
  }
  C[1]
}
Bino.Barrier.UO(10,0.3,10,0.2,0.01,11,128)
```

```
## [1] 0.05340383
```

Comment: In my binomial tree for European up-and out option, I think the tree get convergence by 128 step.

Question(b)

```
#Using B-S method to price an European Up-and-Out Call option
Barrier.BS.UO <- function(S0,K,H,r,tau,sig,div){
  nu = r - div - 0.5 * sig^2
  C.BS.K <- BS("C",S0,K,tau,r,sig,div)
  C.BS.H <- BS("C",S0,H,tau,r,sig,div)
  C.BS.H2K <- BS("C",H^2/S0,K,tau,r,sig,div)
  C.BS.H2H <- BS("C",H^2/S0,H,tau,r,sig,div)
  d.BS.HS <- (log(H/S0)+nu*tau) / (sig*sqrt(tau))
  d.BS.SH <- (log(S0/H)+nu*tau) / (sig*sqrt(tau))
  UO.BS <- C.BS.K - C.BS.H - (H-K)*exp(-r*tau)*pnorm(d.BS.SH) - (H/S0)^(2*nu/sig^2)*
    ( C.BS.H2K - C.BS.H2H - (H-K)*exp(-r*tau)*pnorm(d.BS.HS) )
  return(UO.BS)
}
cat("results by binomial tree:",Bino.Barrier.UO(10,0.3,10,0.2,0.01,11,128),"\nresults by BS formula:",Ba
```

```
## results by binomial tree: 0.05340383
## results by BS formula: 0.0530928
```

Comment: the results of BS method to price this European Up-and-Out Call option are similar with the price that calculated by binomial tree.

Question(c)

```
#Using in-out parity to Price an European Up-and-In call option
Bino.Barrier.UI <- BS("C",10,10,0.3,0.01,0.2,0) - Bino.Barrier.UO(10,0.3,10,0.2,0.01,11,10)
```

```
#Using B-S method to price an European Up-and-Out Call option
Barrier.BS.UI <- function(S0,K,H,r,tau,sig,div){
  nu = r - div - 0.5 * sig^2
  C.BS.K <- BS("C",S0,K,tau,r,sig,div)
  C.BS.H <- BS("C",S0,H,tau,r,sig,div)
  P.BS.H2K <- BS("P",H^2/S0,K,tau,r,sig,div)
  P.BS.H2H <- BS("P",H^2/S0,H,tau,r,sig,div)
  d.BS.HS <- (log(H/S0)+nu*tau) / (sig*sqrt(tau))
  d.BS.SH <- (log(S0/H)+nu*tau) / (sig*sqrt(tau))

  UI.BS <- (H/S0)^(2*nu/sig^2)*(P.BS.H2K-P.BS.H2H+(H-K)*exp(-r*tau)*pnorm(-d.BS.HS))+
    C.BS.H+(H-K)*exp(-r*tau)*pnorm(d.BS.SH)
  return(UI.BS)
}

cat("results by in-out parity:",Bino.Barrier.UI,"\nresults by BS formula:",Barrier.BS.UI(10,10,11,0.01,(
```

```
## results by in-out parity: 0.3895253
## results by BS formula: 0.3981948
```

Comment: We can see from the results above by these two method are match each other.

Question (d)

```
#Pricing European Up-and-Out Put Option
Bino.EUO.Put <- function(K,T,S,sig,r,H,N){
  dt = T/N
  nu = r-0.5*sig^2
  dx = sqrt(sig^2*dt + (nu*dt)^2)
  p = 1/2 + 1/2*(nu*dt/dx)
  #precompute constants
  disc = exp(-r*dt)

  #initialise asset prices at maturity N
  St = array(0,dim = c(N+1))
  St[1] <- S * exp(-dx*N)
  St[2] <- St[1]*exp(2*(dx))
  for(j in 2:N+1){
    St[j] = St[j-1]*exp(2*(dx))
  }
  #initialise option values at maturity
  C = array(0,dim = c(N+1))
    for(j in 1:N+1){
    if(St[j]<H)
      C[j] <- max(0.0,K-St[j])
    else
      C[j] <- 0.0
    }
  #step back through the tree applying the barrier and early exercise condition
  for(i in N:1){
    for(j in 1:i){
      St[j] = St[j]*exp(dx)
```

```r
      if(St[j]<H){
        C[j] = disc*(1-p)*C[j] + disc*p*C[j+1]
      }
      else
        C[j] = 0.0
    }
  }
  C[1]
}
#Calculate European Up-and-In Put Option by in-out parity
Bino.EUI.Put <- BS("P",10,10,0.3,0.01,0.2,0) - Bino.EUO.Put(10, 0.3, 10, 0.2, 0.01,11,190)
#Calculate American Up-and-In Put Option
Bino.AUI.Put <- function(S,tau,X,H,r,sig,q,N){
  (S/H)^(1-2*(r-q)/sig)*(Bino.tree(isCall = F,isAmerican = T, X, tau, H^2/S, r, sig, N)-
                           BS("P",H^2/S, X, tau, r, sig,q)) + Bino.EUI.Put
}
#the results
Bino.AUI.Put(10,0.3,10,11,0.01,0.2,0,190)
```

```
## [1] 0.01765811
```