MGMTMFE 405 - Project 4

Yanxiang Zhao

February 7, 2019

Check Package

I used the package "quantmod" in this project to download the GOOG market data.

```
# Zhao_Yanxiang_Projcet4
##### PACKAGE ######
if (!require("quantmod")) {
  install.packages("quantmod")
  library(quantmod)
}
```

```
## Loading required package: 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

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

Functions

```
##### FUNCTIONS #####
bsCall <- function(s0,t,x,r,sigma){
  # compute d1, d2
  d1 <- (log(s0/x)+(r+sigma^2/2)*t)/(sigma*sqrt(t))
  d2 <- d1-sigma*sqrt(t)
  # output: Black-Sholes price
  c_bs <- s0*pnorm(d1)-x*exp(-r*t)*pnorm(d2)
  return(c_bs)
}
bsPut <- function(s0,t,x,r,sigma){
  # compute d1, d2
```

```r
  d1 <- (log(s0/x)+(r+sigma^2/2)*t)/(sigma*sqrt(t))
  d2 <- d1-sigma*sqrt(t)
  # output: Black-Sholes price
  p_bs <- x*exp(-r*t)*pnorm(-d2)-s0*pnorm(-d1)
  return(p_bs)
}
# combination
nCr <- function(n, r){
  if(r==0){
    return(1)
  } else {
    return(prod(sapply(1:r, function(i)(n-r+i)/(i))))
  }
}
# Halton's Low-Discrepancy Sequences
haltonSeq <- function(num, base){
  hseq <- rep(0, num)
  numBits <- 1+ceiling(log(num)/log(base))
  vetBase <- base^(-(1:numBits))
  workVet <- rep(0, numBits)
  for(i in 1:num){
    j <- 1
    ok <- 0
    while (ok==0) {
      workVet[j] <- workVet[j]+1
      if (workVet[j]<base){
        ok <-  1
      } else {
        workVet[j] <- 0
        j <- j+1
      }
    }
    hseq[i] <- sum(workVet*vetBase)
  }
  return(hseq)
}
##### Question 1 #####
# call option payoffs through binomial tree
binoCall <- function(u, d, p, s0, x, rf, t, n){
  dt <- t/n
  payoffs <- vector()
  for (k in 0:n){
    st <- s0*u^k*d^(n-k)
    payoffs[k+1] <- nCr(n,k)*p^k*(1-p)^(n-k)*ifelse(st>x, st-x, 0)
  }
  call <- sum(payoffs)*exp(-rf*t)
  return(call)
}
# compute u,d,p parameters
q1a_param <- function(r, sigma, t, n){
  dt <- t/n
  c <- .5*(exp(-r*dt)+exp((r+sigma^2)*dt))
  d <- c - sqrt(c^2 - 1)
```

```r
  u <- 1/d
  p <- (exp(r*dt) - d)/(u - d)
  return(c(u, d, p))
}
q1b_param <- function(r, sigma, t, n){
  dt <- t/n
  p <- .5
  u <- exp(r*dt)*(1+sqrt(exp(sigma^2*dt)-1))
  d <- exp(r*dt)*(1-sqrt(exp(sigma^2*dt)-1))
  return(c(u, d, p))
}
q1c_param <- function(r, sigma, t, n){
  dt <- t/n
  p <- .5
  u <- exp((r-sigma^2/2)*dt+sigma*sqrt(dt))
  d <- exp((r-sigma^2/2)*dt-sigma*sqrt(dt))
  return(c(u, d, p))
}
q1d_param <- function(r, sigma, t, n){
  dt <- t/n
  u <- exp(sigma*sqrt(dt))
  d <- exp(-sigma*sqrt(dt))
  p <- .5+.5*(((r-sigma^2/2)*sqrt(dt))/sigma)
  return(c(u, d, p))
}
##### Question 4 #####
# put option binomial model
putBino <- function(u, d, p, s0, x, rf, t, n, American){
  dt <- t/n
  if (American){
    put <- rep(0,(n+2))
    for (step in 0:n){
      k <- (n-step):0
      st <- s0*u^k*d^((n-step)-k)
      cv <- exp(-rf*dt)*na.omit(c(NA,put)*p + c(put,NA)*(1-p))
      put <- ifelse(cv>x-st, cv, ifelse(st<x, x-st, 0))
    }
  } else {
    payoffs <- vector()
    for (k in 0:n){
      st <- s0*u^k*d^(n-k)
      payoffs[k+1] <- nCr(n,k)*p^k*(1-p)^(n-k)*ifelse(st<x, x-st, 0)
    }
    put <- sum(payoffs)*exp(-rf*t)

  }
  return(put)
}
##### Question 5 #####
trinoCall <- function(u, d, pU, pD, s0, x, rf, t, n, logPrice){
  powU <- c((n):0, rep(0,(n)))
  powD <- c(rep(0,(n)), 0:(n))
  if(logPrice){
```

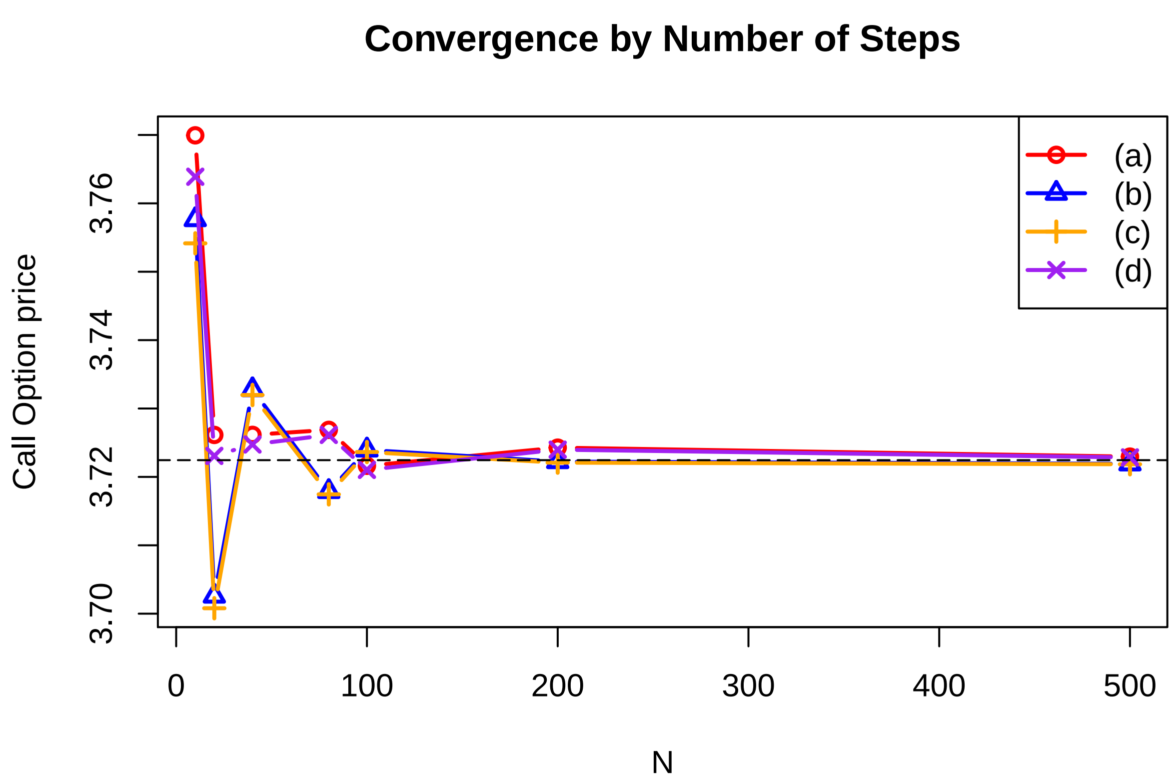
```
    st <- exp(log(s0)+u*powU+d*powD)
  } else {
    st <- s0*u^powU*d^powD
  }
  call <- ifelse(st>x, st-x, 0)
  for (step in 1:n){
    call <- exp(-rf*dt)*na.omit((c(NA, NA, call)*pU+c(NA, call, NA)*(1-pU-pD)+c(call, NA, NA)*pD))
  }
  return(call)
}
q5a_param <- function(r, sigma, t, n){
  dt <- t/n
  d <- exp(-sigma*sqrt(3*dt))
  u <- 1/d
  pD <- (r*dt*(1-u)+(r*dt)^2+sigma^2*dt)/((u-d)*(1-d))
  pU <- (r*dt*(1-d)+(r*dt)^2+sigma^2*dt)/((u-d)*(u-1))
  return(list(u=u,d=d,pU=pU,pD=pD))
}
q5b_param <- function(r, sigma, t, n){
  dt <- t/n
  d <- -sigma*sqrt(3*dt)
  u <- sigma*sqrt(3*dt)
  pD <- .5*((sigma^2*dt+(r-sigma^2/2)^2*dt^2)/(u^2)-((r-sigma^2/2)*dt)/(u))
  pU <- .5*((sigma^2*dt+(r-sigma^2/2)^2*dt^2)/(u^2)+((r-sigma^2/2)*dt)/(u))
  return(list(u=u,d=d,pU=pU,pD=pD))
}

##### Question 6 #####
call_halton <- function(s0, x, rf, sigma, t, n, b1, b2){
  # get Halton sequences
  h1 <- haltonSeq(n/2, b1)
  h2 <- haltonSeq(n/2, b2)
  # get random normal variables
  z1 <- sqrt(-2*log(h1))*cos(2*pi*h2)
  z2 <- sqrt(-2*log(h1))*sin(2*pi*h2)
  wt <- sqrt(t)*c(rbind(z1, z2)) # c(rbind(z1, z2)) mix z1 and z2
  st <- s0*exp((rf-sigma^2/2)*t + sigma*wt)
  call <- exp(-rf*t)*mean(ifelse((st-x)>0,st-x,0))
  return(call)
}
```

Question 1

```
##### Question 1 #####
# parameters
rf <- .05
sigma <- .24
s0 <- 32
x <- 30
t <- 0.5
N <- c(10,20,40,80,100,200,500)
```

```r
# run binomial methods
q1 <- list(a=NULL,b=NULL,c=NULL, d=NULL)
for (i in 1:length(N)){
  # (a)
  udp1 <- q1a_param(rf, sigma, t, N[i])
  q1$a[i] <- binoCall(udp1[1], udp1[2], udp1[3], s0, x, rf, t, N[i])
  # (b)
  udp2 <- q1b_param(rf, sigma, t, N[i])
  q1$b[i] <- binoCall(udp2[1], udp2[2], udp2[3], s0, x, rf, t, N[i])
  # (c)
  udp3 <- q1c_param(rf, sigma, t, N[i])
  q1$c[i] <- binoCall(udp3[1], udp3[2], udp3[3], s0, x, rf, t, N[i])
  # (d)
  udp4 <- q1d_param(rf, sigma, t, N[i])
  q1$d[i] <- binoCall(udp4[1], udp4[2], udp4[3], s0, x, rf, t, N[i])
}
q1mat <- matrix(unlist(q1), i)
matplot(N,q1mat, type = "b", pch = c(1:4), lwd = 2, lty = 1,
        main = "Convergence by Number of Steps",
        ylab = "Call Option price",
        col = c("red", "blue", "orange", "purple"))
legend("topright", legend = paste0("(",letters[1:4],")"),
       pch = c(1:4), lwd = 2, col = c("red", "blue", "orange", "purple"))
abline(h=bsCall(s0,t,x,rf,sigma), lty = 5)
```



Question 2 I retrieved the GOOG market price on Feb 04, 2019 after the market closure. I imported

5-year GOOG daily price data ending on Feb 04, 2019. I also found the expiration date of the options in January 2020 in on the 17th.

```
##### Question 2 #####
# (a)
# get 5-year GOOG data
end <- as.Date("2019-02-05") # retrieve data end on Feb. 04, 2019
start <- seq(end, length=2, by="-5 years")[2]
suppressWarnings(getSymbols("GOOG", src = "yahoo", from = start, to = end))
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).
```

```
## [1] "GOOG"
```

```
# get parameters
rf <- 0.02
s0 <- GOOG$GOOG.Close[nrow(GOOG)]
x <- round(s0*1.1/10)*10
t <- as.numeric(as.Date("2020-01-17") - end)/365
     # assume option expires on Jan 17, 2020
N <- 1000
# find sigma
adjPrice <- as.numeric(GOOG$GOOG.Adjusted)
ret <- (c(adjPrice, NA) - c(NA, adjPrice))/c(NA, adjPrice)
sigma <- sd(na.omit(ret))*sqrt(252)
# use binomial method stated in 1(d)
udp_2 <- q1d_param(rf, sigma, t, N)
GOOG_c_bino <- binoCall(udp_2[1], udp_2[2], udp_2[3], s0, x, rf, t, N)
cat("The call option price from binomial method is $", round(GOOG_c_bino,2))
```

```
## The call option price from binomial method is $ 66.92
```

By using the binomial model, I found the call option price to be $66.92. On Feb. 04, 2019, the market price of a call option expires on Jan. 17, 2020 with a strike price of $1250 is $68.80. The price is higher than the price I found using the model. I know that the call option price has a positive relationship with the underlying volatility. The implied volatility of the underlying must be higher than the historical volatility I input to the model.

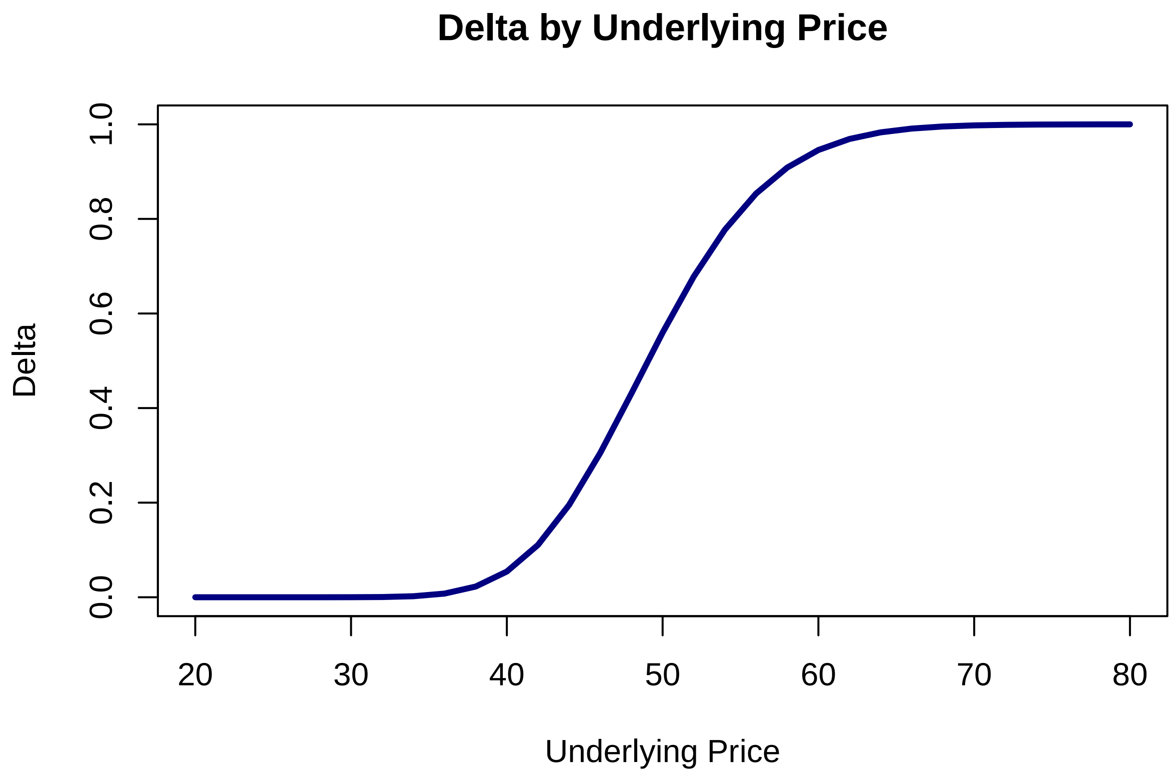
```r
# (b)
# the market price of the call option is 68.80 on Feb. 04, 2019
GOOG_c_mkt <- 68.8
dSigma <- 0.0001
iv <- sigma
GOOG_c_bino_i <- GOOG_c_bino
while(round(GOOG_c_bino_i,1) != GOOG_c_mkt){
  iv <- iv+dSigma
  udp_2 <- q1d_param(rf, iv, t, N)
  GOOG_c_bino_i <- binoCall(udp_2[1], udp_2[2], udp_2[3], s0, x, rf, t, N)
}
cat("The implied volatitily is ", iv, "\n")
```

```
## The implied volatitily is  0.2391788
```

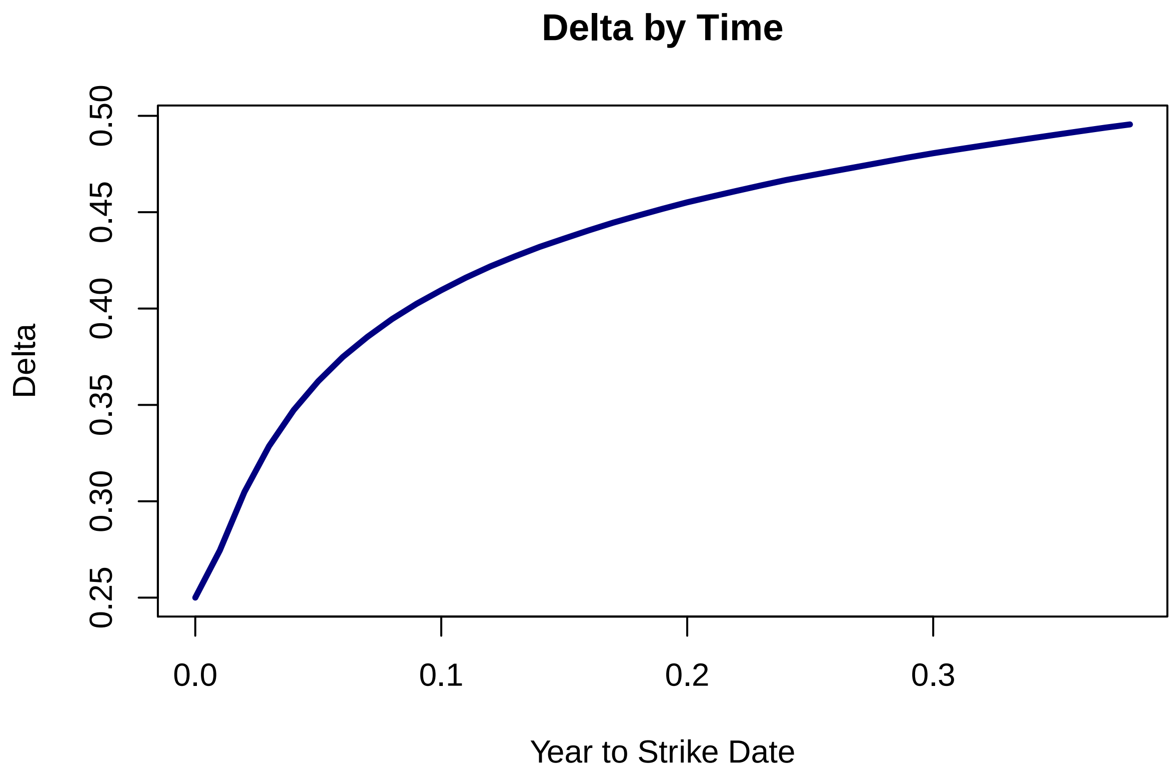
I found the implied volatility is at around 0.2392, which is slightly higher than the historical volatility 0.2349. The simple historical volatility of the underlying might not be the best estimate of the volatility today. The investors has used some other models, such as GARCH and EWMA, to find a more reasonable volatility. This is part of the reason why the market priced the option differently than I did.

Question 3

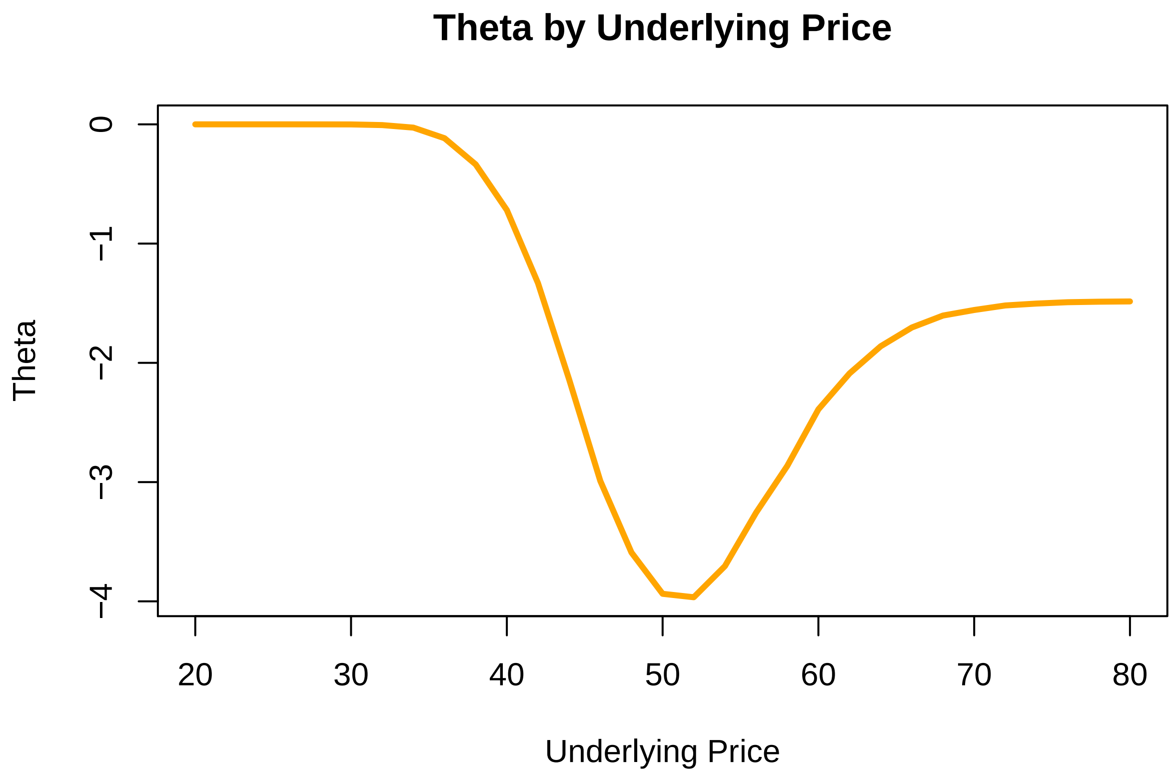
```r
##### Question 3 #####
# set parameters
N <- 1000
s0 <- seq(20, 80, 2)
x <- 50
rf <- 0.03
sigma <- 0.2
t <- 0.3846
# set changes
ds0 <- 2 # change in price
dt <- 0.000001 # change in time
dsigma <- 0.00000001 # change in sigma
drf <- 0.000001
# (i)
delta_1 <- vector()
# use binomial method stated in 1(d)
udp_3i <- q1d_param(rf, sigma, t, N)
# use binomial method to find deltas
for (i in 1:length(s0)){
  c_1 <- binoCall(udp_3i[1], udp_3i[2], udp_3i[3], s0[i]-ds0, x, rf, t, N)
  c_2 <- binoCall(udp_3i[1], udp_3i[2], udp_3i[3], s0[i]+ds0, x, rf, t, N)
  delta_1[i] <- (c_2-c_1)/(2*ds0)
}
plot(s0,delta_1, type = "l", lwd = 3, col = "navy",
     main = "Delta by Underlying Price",
     xlab = "Underlying Price", ylab = "Delta")
```



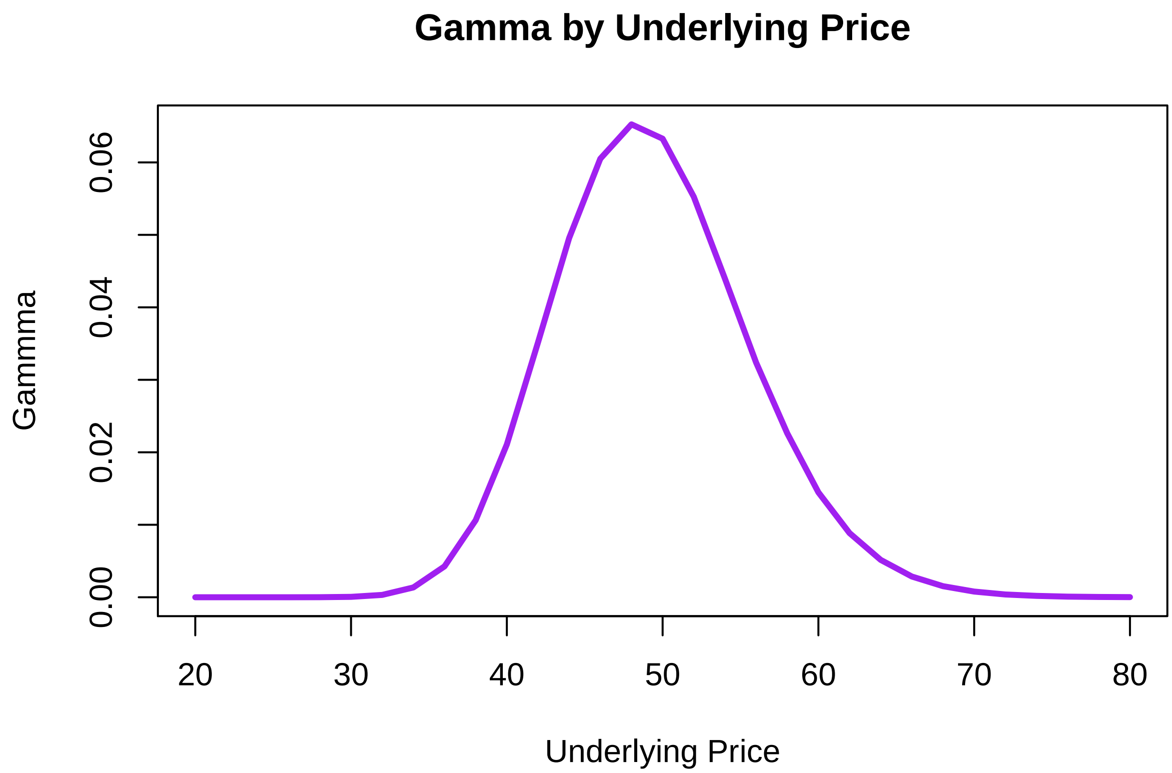

```r
# (ii)
tii <- seq(0, 0.3846, 0.01)
s0ii <- 49
delta_2 <- vector()
for (i in 1:length(tii)){
  udp_3ii <- q1d_param(rf, sigma, tii[i], N)
  c_1 <- binoCall(udp_3ii[1], udp_3ii[2], udp_3ii[3], s0ii-ds0, x, rf, tii[i], N)
  c_2 <- binoCall(udp_3ii[1], udp_3ii[2], udp_3ii[3], s0ii+ds0, x, rf, tii[i], N)
  delta_2[i] <- (c_2-c_1)/(2*ds0)
}
plot(tii,delta_2, type = "l", lwd = 3, col = "navy",
     main = "Delta by Time",
     xlab = "Year to Strike Date", ylab = "Delta")
```



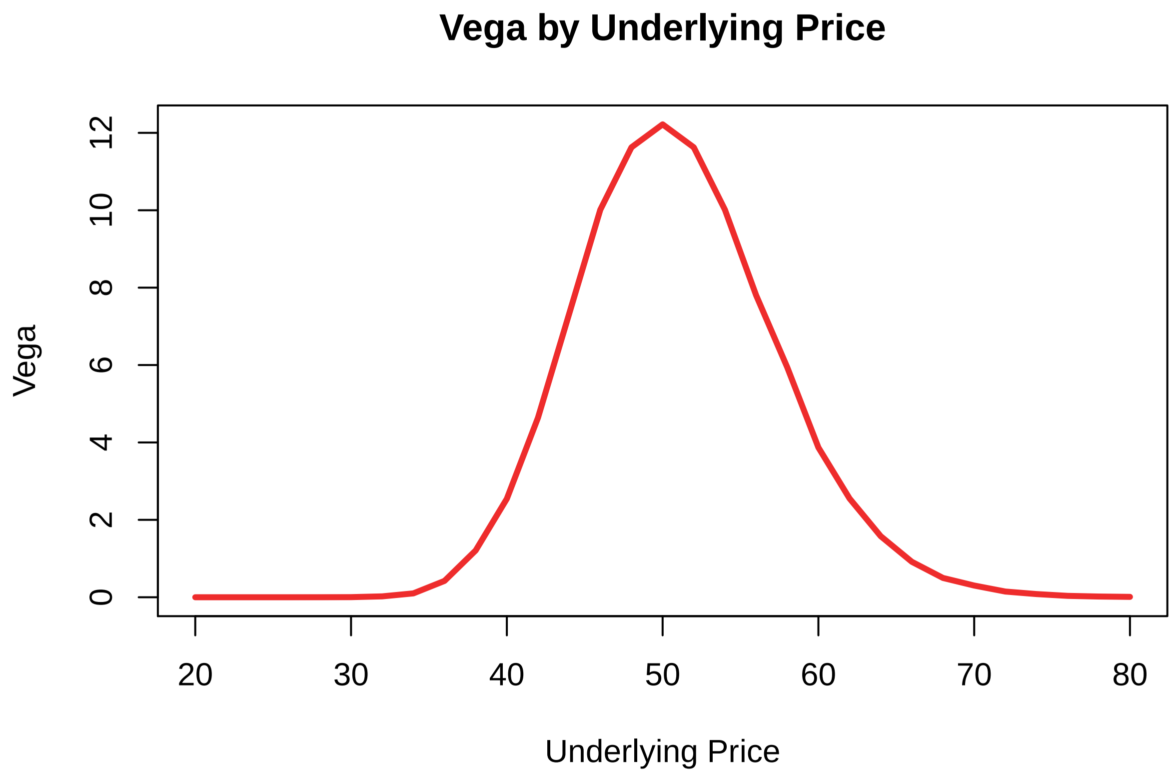

```r
# (iii)
theta <- vector()
udp_3iiia <- q1d_param(rf, sigma, t+dt, N)
udp_3iiib <- q1d_param(rf, sigma, t-dt, N)
for (i in 1:length(s0)){
  c_1 <- binoCall(udp_3iiia[1], udp_3iiia[2], udp_3iiia[3], s0[i], x, rf, t+dt, N)
  c_2 <- binoCall(udp_3iiib[1], udp_3iiib[2], udp_3iiib[3], s0[i], x, rf, t-dt, N)
  theta[i] <- (c_2-c_1)/(2*dt)
}
plot(s0,theta, type = "l", lwd = 3, col = "orange",
     main = "Theta by Underlying Price",
     xlab = "Underlying Price", ylab = "Theta")
```



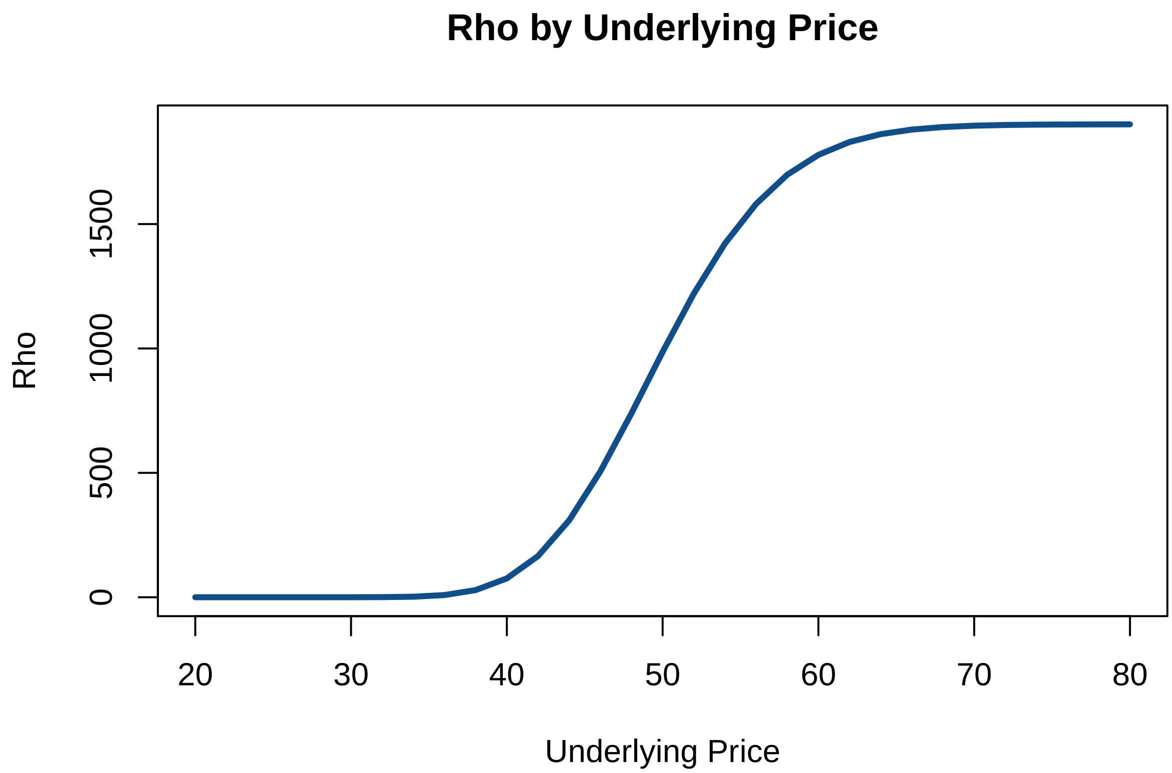

```
# (iv)
gamma <- vector()
udp_3iv <- q1d_param(rf, sigma, t, N)
for (i in 1:length(s0)){
  c_1 <- binoCall(udp_3iv[1], udp_3iv[2], udp_3iv[3], s0[i]-ds0, x, rf, t, N)
  c_2 <- binoCall(udp_3iv[1], udp_3iv[2], udp_3iv[3], s0[i], x, rf, t, N)
  c_3 <- binoCall(udp_3iv[1], udp_3iv[2], udp_3iv[3], s0[i]+ds0, x, rf, t, N)
  gamma[i] <- (c_3-2*c_2+c_1)/(ds0^2)
}
plot(s0,gamma, type = "l", lwd = 3, col = "purple",
     main = "Gamma by Underlying Price",
     xlab = "Underlying Price", ylab = "Gammma")
```




```r
# (v)
vega <- vector()
# use binomial method stated in 1(d)
for (i in 1:length(s0)){
  udp_3va <- q1d_param(rf, sigma-dsigma, t, N)
  udp_3vb <- q1d_param(rf, sigma+dsigma, t, N)
  c_1 <- binoCall(udp_3va[1], udp_3va[2], udp_3va[3], s0[i], x, rf, t, N)
  c_2 <- binoCall(udp_3vb[1], udp_3vb[2], udp_3vb[3], s0[i], x, rf, t, N)
  vega[i] <- (c_2-c_1)/(2*dsigma)
}
plot(s0,vega, type = "l", lwd = 3, col = "firebrick2",
     main = "Vega by Underlying Price",
     xlab = "Underlying Price", ylab = "Vega")
```

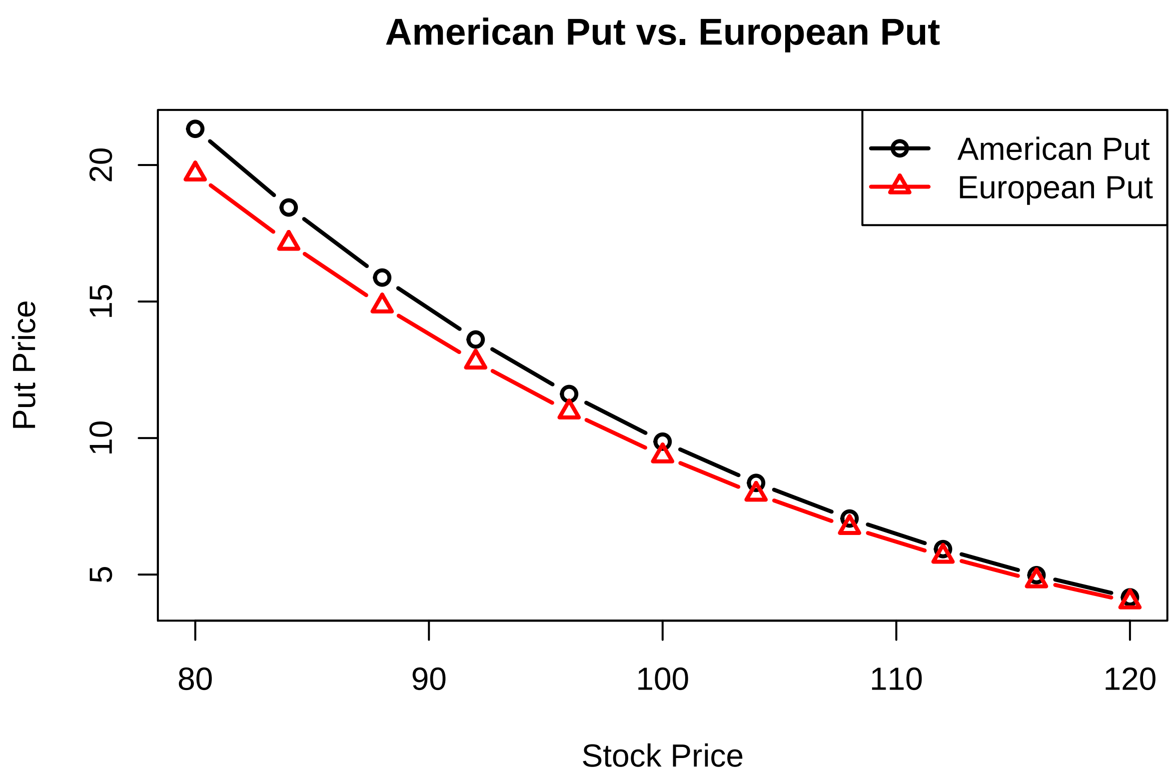



```r
# (vi)
rho <- vector()
for (i in 1:length(s0)){
  udp_3via <- q1d_param(rf-drf, sigma-dsigma, t, N)
  udp_3vib <- q1d_param(rf+drf, sigma+dsigma, t, N)
  c_1 <- binoCall(udp_3via[1], udp_3via[2], udp_3via[3], s0[i], x, rf-drf, t, N)
  c_2 <- binoCall(udp_3vib[1], udp_3vib[2], udp_3vib[3], s0[i], x, rf+drf, t, N)
  rho[i] <- (c_2-c_1)/(2*dsigma)
}
plot(s0,rho, type = "l", lwd = 3, col = "dodgerblue4",
     main = "Rho by Underlying Price",
     xlab = "Underlying Price", ylab = "Rho")
```



Question 4

```
##### Question 4 #####
t <- 1
n <- 1000
rf <- 0.05
sigma <- 0.3
x <- 100
s0 <- seq(80,120,4)
put_a <- vector()
put_u <- vector()
for (i in 1:length(s0)){
  udq_4 <- q1d_param(rf, sigma, t, n)
  put_a[i] <- putBino(udq_4[1], udq_4[2], udq_4[3], s0[i], x, rf, t, n, T)
  put_u[i] <- putBino(udq_4[1], udq_4[2], udq_4[3], s0[i], x, rf, t, n, F)
}
matplot(s0,as.matrix(cbind(put_a, put_u),i), type = "b",
        pch = c(1:2), lwd = 2, lty = 1,
        main = "American Put vs. European Put",
        xlab = "Stock Price", ylab = "Put Price")
legend("topright", legend = c("American Put", "European Put"),
       pch = c(1:2), col = c(1:2), lwd = 2)
```

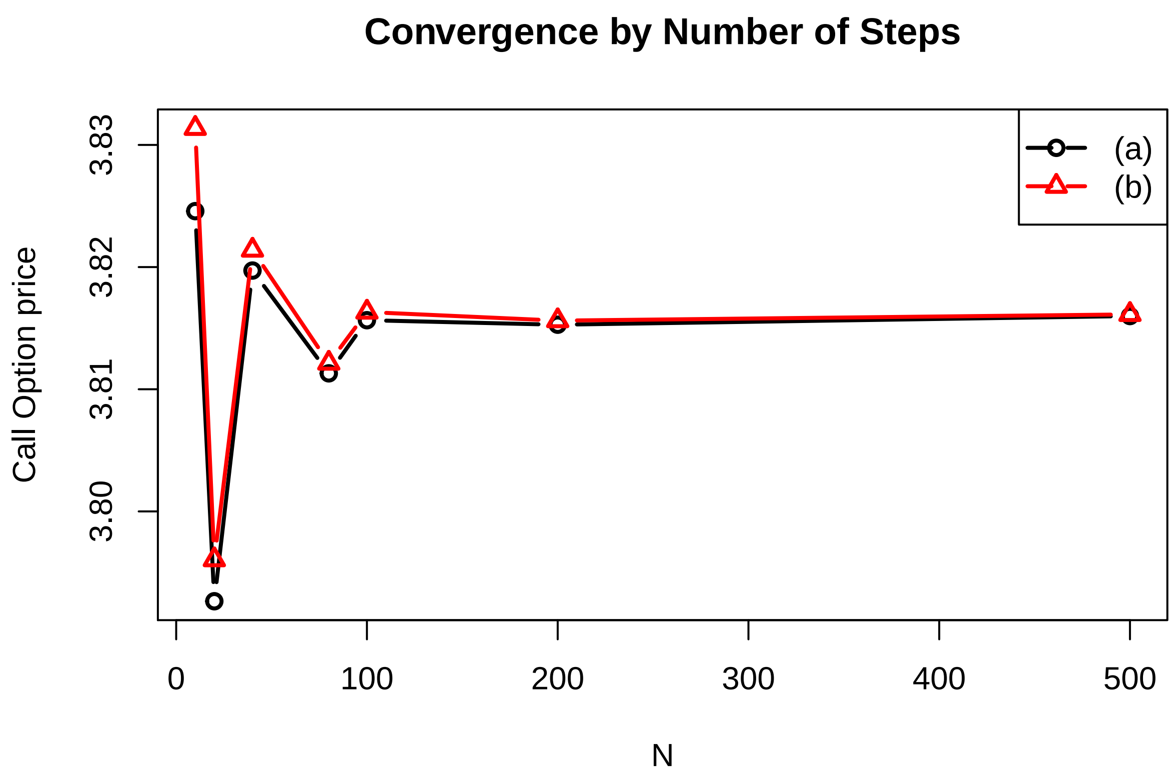


The graph shows that the price of the American put option is systematically higher than the price of the European put option. This is because the American option gives the buyer the right to early exercise. Inverstors who bought American options can exercise when the continuation value is lower than the exercise value, while the European option investors have to hold the option to maturity regardless of the two values.

Question 5

```
##### Question 5 #####
# set parameters
rf <- .05
sigma <- .24
s0 <- 32
x <- 30
t <- 0.5
N <- c(10,20,40,80,100,200,500)
# similar to q1
q5 <- list(a=NULL,b=NULL)
for (i in 1:length(N)){
  # (a)
  q5a <- q5a_param(rf, sigma, t, N[i])
  q5$a[i] <- trinoCall(q5a$u, q5a$d, q5a$pU, q5a$pD, s0, x, rf, t, N[i], F)
  # (b)
  q5b <- q5b_param(rf, sigma, t, N[i])
  q5$b[i] <- trinoCall(q5b$u, q5b$d, q5b$pU, q5b$pD, s0, x, rf, t, N[i], T)
```

```r
}
q5mat <- matrix(unlist(q5), i)
matplot(N,q5mat, type = "b", pch = c(1:2), lwd = 2, lty = 1,
        main = "Convergence by Number of Steps",
        ylab = "Call Option price")
legend("topright", legend = paste0("(",letters[1:2],")"),
       pch = c(1:2), lwd = 2, lty = 5, col = c(1:2))
```



Question 6

```r
##### Question 6 #####
# set parameters
rf <- .05
sigma <- .24
s0 <- 32
x <- 30
t <- 0.5
n <- 100000
base <- c(7,5)
C <- call_halton(s0,x,rf,sigma,t,n,base[1],base[2])
cat("The call option price = ", C)
```

```
## The call option price =  3.722732
```