FE621: Assignment 1 - Napat L

Napat L

February 11, 2018

FE621. Assignment #1.

Part 1: Data gathering component

1) Function to download the data

```
library(quantmod)

setwd("C:/Users/nackz/Desktop/Stevens Institute/Subjects/FE621 - Computational Methods in
Finance/Assignments/Assignment 1")


#3 securities to be used
sec <- c("AMZN","SPY", "^VIX")
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 <- (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$T.mat <- (df.temp$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]
    }
    thepage = readLines("https://www.federalreserve.gov/releases/h15/")
    mypattern = '<td class="data" headers="id88bf848 id88bf960 id88bfb90 col5" nowrap="nowrap"> '
    datalines = grep(mypattern,thepage,value=TRUE)
```

```
    interest <- getInterest(datalines)
    df$interest <- interest
    return(df)
}


getInterest = function(data, patterns){
    index1 = gregexpr(pattern = '<td class="data" headers="id88bf848 id88bf960 id88bfb90 col5"
nowrap="nowrap"> ', data)[[1]][1]
    index2 = gregexpr(pattern = ' </td>', data)[[1]][1]
    interest = substring(data, index1+84, index2-1)
    interest <- as.double(interest)/100
    return(interest)
}

#data1 <- getOptEqtPrice(sec)
#data2 <- getOptEqtPrice(sec)
data1 <- read.csv("data1.csv")
#data2 < read.csv("data2.csv")

#write.csv(data1, file = "data1.csv")
#write.csv(data2, file = "data2.csv")
```

2) Data gathered

From the samples of dataset shown, the maturity of AMZN was downloaded over 3 months and there are securities which are AMZN, SPY and VIX.

```
unique(data1$security)

## [1] AMZN SPY  ^VIX
## Levels: ^VIX AMZN SPY

head(data1[data1$security == "AMZN",], 4)

##                       X security Strike   Last       Chg    Bid    Ask Vol
## 1 AMZN180209C00960000     AMZN    960 488.70 153.99002 428.45 433.45   1
## 2 AMZN180209C00990000     AMZN    990 480.63  26.80002 438.60 443.60   5
## 3 AMZN180209C01000000     AMZN   1000 355.00   0.00000 399.50 404.50   1
## 4 AMZN180209C01010000     AMZN   1010 356.00   0.00000 389.50 394.50   1
##   OI   maturity  type time.stamp      T.mat equity.price interest
## 1  2 2018-02-09 calls 2018-02-05 0.01587302         1390   0.0146
## 2  6 2018-02-09 calls 2018-02-05 0.01587302         1390   0.0146
## 3  1 2018-02-09 calls 2018-02-05 0.01587302         1390   0.0146
## 4  1 2018-02-09 calls 2018-02-05 0.01587302         1390   0.0146

tail(data1[data1$security == "AMZN",], 4)

##                          X security Strike   Last        Chg    Bid   Ask
## 3954 AMZN200117P02000000     AMZN   2000 575.00 -37.130005 605.75 615.5
## 3955 AMZN200117P02020000     AMZN   2020 677.00  -3.000000 645.50 650.5
## 3956 AMZN200117P02025000     AMZN   2025 664.95  -5.049988 669.00 678.5
## 3957 AMZN200117P02100000     AMZN   2100 726.00  56.000000 734.00 743.5
##      Vol OI   maturity type time.stamp    T.mat equity.price interest
## 3954  10 32 2020-01-17 puts 2018-02-05 2.821429         1390   0.0146
## 3955   2  5 2020-01-17 puts 2018-02-05 2.821429         1390   0.0146
## 3956   1  4 2020-01-17 puts 2018-02-05 2.821429         1390   0.0146
## 3957   6  3 2020-01-17 puts 2018-02-05 2.821429         1390   0.0146
```

3) Write a paragraph describing the symbols you are downloading data for. Explain what is the SPY and its purpose.Explain what is VIX and its purpose. Understand the options symbols. Understand when each option expires. Write this information and turn it in.

AMZN

AMZN is Amazon.com, Inc. is an online retailer that offers a wide range of products. The Company products include books, music, videotapes, computers, electronics, home and garden, and numerous other products. Amazon offers personalized shopping services, Web-based credit card payment, and direct shipping to customers. There are expiration date on every Friday each weeK.

```
unique(subset(data1, security == "AMZN")$maturity)

##  [1] 2018-02-09 2018-02-16 2018-02-23 2018-03-02 2018-03-09 2018-03-16
##  [7] 2018-03-23 2018-04-20 2018-06-15 2018-07-20 2018-09-21 2019-01-18
## [13] 2019-06-21 2020-01-17
## 36 Levels: 2018-02-07 2018-02-09 2018-02-14 2018-02-16 ... 2020-12-18
```

SPY

SPY is SPDR S&P 500 ETF Trust is an exchange-traded fund incorporated in the USA. The ETF tracks the S&P 500 Index. The Trust consists of a portfolio representing all 500 stocks in the S&P 500 Index. It holds predominantly large-cap U.S. stocks. This ETF is structured as a Unit Investment Trust and pays dividends on a quarterly basis. The holdings are weighted by market capitalization. There are usually expiration date on every Wednesday and Friday each week.

```
unique(subset(data1, security == "SPY")$maturity)

##  [1] 2018-02-07 2018-02-09 2018-02-14 2018-02-16 2018-02-21 2018-02-23
##  [7] 2018-02-28 2018-03-02 2018-03-07 2018-03-09 2018-03-16 2018-03-23
## [13] 2018-03-29 2018-04-20 2018-05-18 2018-06-15 2018-06-29 2018-07-20
## [19] 2018-09-21 2018-09-28 2018-12-21 2018-12-31 2019-01-18 2019-03-15
## [25] 2019-06-21 2019-09-20 2019-12-20 2020-01-17 2020-07-17 2020-12-18
## 36 Levels: 2018-02-07 2018-02-09 2018-02-14 2018-02-16 ... 2020-12-18
```

VIX

VIX is The Chicago Board Options Exchange Volatility Index reflects a market estimate of future volatility, based on the weighted average of the implied volatilities for a wide range of strikes. The Expiration Date (usually a Wednesday) will be identified explicitly in the expiration date of the product. If that Wednesday or the Friday that is 30 days following that Wednesday is an Exchange holiday, the Expiration Date will be on the business day immediately preceding that Wednesday.

```
unique(subset(data1, security == "^VIX")$maturity)

##  [1] 2018-02-07 2018-02-14 2018-02-21 2018-02-27 2018-03-07 2018-03-21
##  [7] 2018-04-18 2018-05-16 2018-06-20 2018-07-18
## 36 Levels: 2018-02-07 2018-02-09 2018-02-14 2018-02-16 ... 2020-12-18
```

4) Important information

The equity price field shows both spot price of both equity and ETF prices. The interest field shows the interest rate (risk-free rate). The maturity and T.mat field show the expiration day and time to maturity, respectively.

```
head(data1[data1$security == "SPY",], 4)

##                         X security Strike  Last    Chg  Bid   Ask  Vol OI
## 3958 SPY180207C00255000      SPY    255 10.85 -10.23 8.11 12.50   50 11
## 3959 SPY180207C00260000      SPY    260  6.70 -11.35 6.31  7.00  191 13
## 3960 SPY180207C00261000      SPY    261  5.38  -9.99 5.43  5.85  162  8
## 3961 SPY180207C00264000      SPY    264  3.07 -11.63 3.67  3.99 2611  1
##        maturity  type time.stamp       T.mat equity.price interest
## 3958 2018-02-07 calls 2018-02-05 0.007936508       264.18   0.0146
## 3959 2018-02-07 calls 2018-02-05 0.007936508       264.18   0.0146
## 3960 2018-02-07 calls 2018-02-05 0.007936508       264.18   0.0146
## 3961 2018-02-07 calls 2018-02-05 0.007936508       264.18   0.0146
```

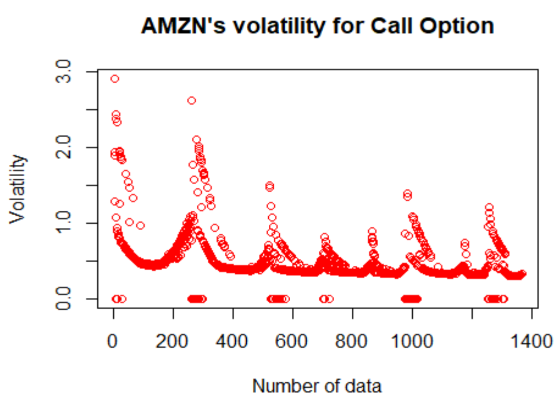
Part 2: Analysis of the data.

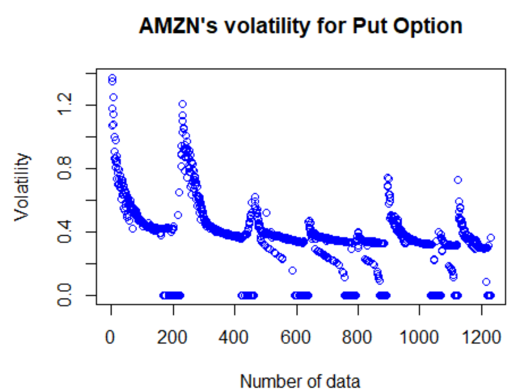
5) Black-Scholes function

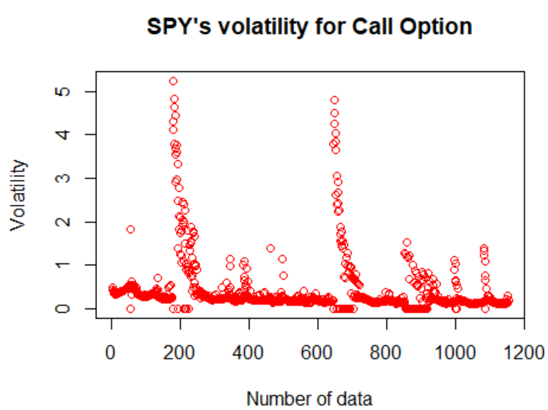
```r
BS <- function (K, S, t, r, v, type){
    d1 <- (log(S/K)+(r+(v^2/2)*t))/(v*sqrt(t))
    d2 <- d1 - v * sqrt(t)
    OptPrice <- NULL
    if(tolower(type) == "calls"){
        OptPrice <- S*pnorm(d1) - K*exp(-r*t)*pnorm(d2)
    }else{
        OptPrice <- K*exp(-r*t) * pnorm(-d2) - S*pnorm(-d1)
    }
    return(OptPrice)
}
```

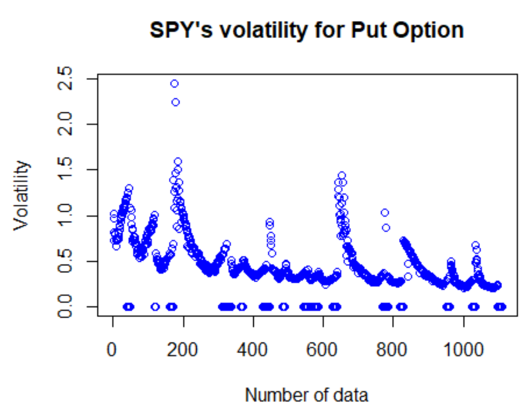
6) Implement the Bisection method

Using tolerance level of 10^{-6}, we get the volatility smiles as figures below. On the figures, one smile represents one set of volatilities within one maturity.









Sample of table that contains calculated implied volatility.

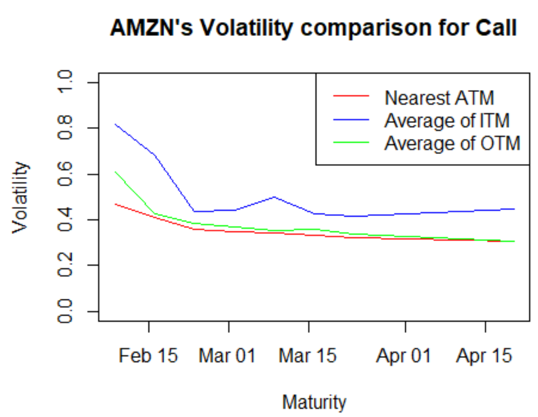
```
> tail(DATA1[DATA1$security == "AMZN",])
                       X security Strike   Last      Chg   Bid    Ask Vol OI   maturity type time.stamp
2592 AMZN180420P01740000     AMZN   1740 316.00  0.00000 311.50 316.50  10 20 2018-04-20 puts 2018-02-05
2593 AMZN180420P01780000     AMZN   1780 318.92  0.00000 349.50 354.50   5  0 2018-04-20 puts 2018-02-05
2594 AMZN180420P01800000     AMZN   1800 337.01  0.00000 370.20 375.20  15  0 2018-04-20 puts 2018-02-05
2595 AMZN180420P01820000     AMZN   1820 398.05  0.00000 388.35 393.35   2  2 2018-04-20 puts 2018-02-05
2596 AMZN180420P01880000     AMZN   1880 489.55 35.19998 488.25 493.25   1  1 2018-04-20 puts 2018-02-05
2597 AMZN180420P01920000     AMZN   1920 473.20  0.00000 486.85 491.85   1  0 2018-04-20 puts 2018-02-05
         T.mat equity.price interest    bi.vol bi.no    bi.time   money.type   new.vol new.no new.time
2592 0.2936508         1390   0.0146 0.0000000     1 0.00000000 In-the-money 0.0000000      4        0
2593 0.2936508         1390   0.0146 0.0000000     1 0.00000000 In-the-money 0.0000000      4        0
2594 0.2936508         1390   0.0146 0.0000000     1 0.00000000 In-the-money 0.0000000      4        0
2595 0.2936508         1390   0.0146 0.0000000     1 0.00000000 In-the-money 0.0000000      4        0
2596 0.2936508         1390   0.0146 0.3617421    31 0.01562881 In-the-money 0.3617421      6        0
2597 0.2936508         1390   0.0146 0.0000000     1 0.00000000 In-the-money 0.0000000      4        0
     secant.vol secant.no secant.time                key
2592  0.0000000        14 0.000000000 AMZN-1740-2018-04-20
2593  0.0000000        14 0.000000000 AMZN-1780-2018-04-20
2594  0.0000000        16 0.001003027 AMZN-1800-2018-04-20
2595  0.0000000        18 0.000000000 AMZN-1820-2018-04-20
2596  0.3617421        12 0.000000000 AMZN-1880-2018-04-20
2597  0.0000000        14 0.000000000 AMZN-1920-2018-04-20
>
```

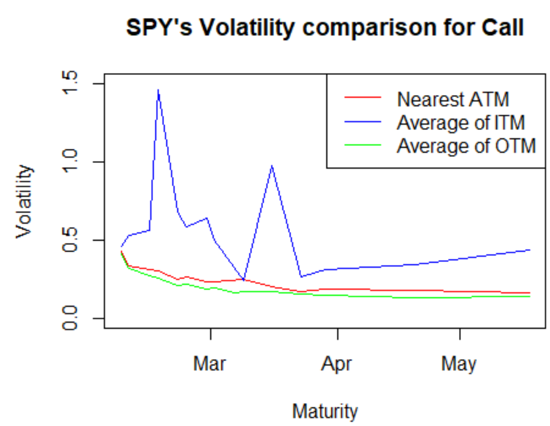
```
> tail(DATA1[DATA1$security == "SPY",])
                      X security Strike   Last Chg    Bid    Ask Vol OI   maturity type time.stamp     T.mat
6217 SPY180518P00385000      SPY    385 103.32   0 109.58 110.21  10 10 2018-05-18 puts 2018-02-05 0.4047619
6218 SPY180518P00390000      SPY    390 108.29   0 114.57 115.20  72 72 2018-05-18 puts 2018-02-05 0.4047619
6219 SPY180518P00395000      SPY    395 113.19   0 119.56 120.19  10 10 2018-05-18 puts 2018-02-05 0.4047619
6220 SPY180518P00400000      SPY    400 118.04   0 124.52 125.15  20 20 2018-05-18 puts 2018-02-05 0.4047619
6221 SPY180518P00405000      SPY    405 123.25   0 129.53 130.17  29 29 2018-05-18 puts 2018-02-05 0.4047619
6222 SPY180518P00410000      SPY    410 128.24   0 134.52 135.16  20 20 2018-05-18 puts 2018-02-05 0.4047619
     equity.price interest bi.vol bi.no bi.time   money.type new.vol new.no new.time secant.vol secant.no
6217       264.18   0.0146      0     1       0 In-the-money       0      5        0          0        19
6218       264.18   0.0146      0     1       0 In-the-money       0      4        0          0        20
6219       264.18   0.0146      0     1       0 In-the-money       0      4        0          0        26
6220       264.18   0.0146      0     1       0 In-the-money       0      4        0          0        25
6221       264.18   0.0146      0     1       0 In-the-money       0      4        0          0        16
6222       264.18   0.0146      0     1       0 In-the-money       0      4        0          0        14
     secant.time               key
6217           0 SPY-385-2018-05-18
6218           0 SPY-390-2018-05-18
6219           0 SPY-395-2018-05-18
6220           0 SPY-400-2018-05-18
6221           0 SPY-405-2018-05-18
6222           0 SPY-410-2018-05-18
>
```

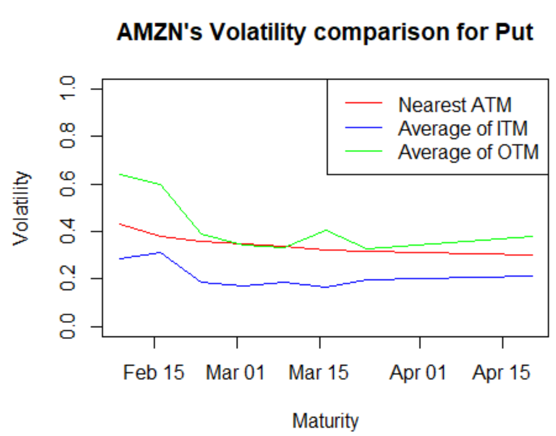
We use 1% range (0.99, 1.01) to identify "At-the-money", "Out-of-the-money" and "In-the-money", all volatilities seem very close to each other as figures below. For "At-the-money", we find the nearest value between strike price and equity price. Then, we calculate the average of volatilities between "In-the-money" and "At-the-money" within its maturity.

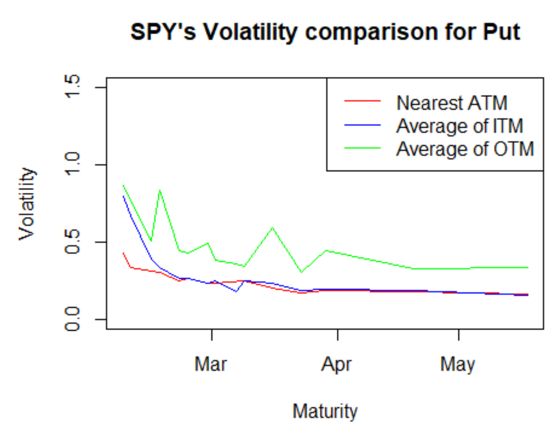
The result shows that the majority of the nearest "At-the-money" is close to the average of volatilities of "Out-of-the-money". However, for SPY Put Option, the nearest "At-the-money" volatility is closer to the average of volatility of "In-the-money".

Moreover, the result also shows that for Call option, the average of "In-the-money" is higher than the average of "Out-of-the-money". On the other hand, for Put option, the average of "Out-of-the-money" is higher than the average of "In-the-money".









Code:

```
DATA1 <- read.csv("data1.csv")
DATA1$maturity <- as.Date(DATA1$maturity)
DATA1 <- DATA1[DATA1$maturity < as.Date("2018-06-01"),]

set.seed(999)
ImpVolBisection <- function (K, S, t, r, OptVal, type, start, end, epsilon = 1e-6){
    n = 1
    a <- Sys.time()
    while(abs((BS(K, S, t, r, end, type) - OptVal) - (BS(K, S, t, r, start, type) - OptVal)) > epsilon){
```

```r
        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
    }
    b <- Sys.time()
    return(c(mid, n, as.double.difftime(b-a)))
}

bi.vol.matrix <- matrix(, nrow = nrow(DATA1), ncol = 3)
for(i in 1:nrow(DATA1)){
    bi.vol <- ImpVolBisection(DATA1$Strike[i], DATA1$equity.price[i], DATA1$T.mat[i], DATA1$interest[i],
(DATA1$Bid[i] + DATA1$Ask[i])/2, DATA1$type[i], 0.001, 8)
    bi.vol.matrix[i,1] <- bi.vol[1]
    bi.vol.matrix[i,2] <- bi.vol[2]
    bi.vol.matrix[i,3] <- bi.vol[3]
}


DATA1 <- cbind(DATA1[,1:15], data.frame(bi.vol = bi.vol.matrix[,1], bi.no = bi.vol.matrix[,2], bi.time =
bi.vol.matrix[,3]))

plot(subset(subset(DATA1, security == "AMZN"), type == "calls")$bi.vol, col = "red", xlab = "Number of
data", ylab = "Volatility", main = "AMZN's volatility for Call Option")

plot(subset(subset(DATA1, security == "AMZN"), type == "puts")$bi.vol, col = "blue", xlab = "Number of
data", ylab = "Volatility", main = "AMZN's volatility for Put Option")

plot(subset(subset(DATA1, security == "SPY"), type == "calls")$bi.vol, col = "red", xlab = "Number of
data", ylab = "Volatility", main = "SPY's volatility for Call Option")

plot(subset(subset(DATA1, security == "SPY"), type == "puts")$bi.vol, col = "blue", xlab = "Number of
data", ylab = "Volatility", main = "SPY's volatility for Put Option")

plot(subset(subset(DATA1, security == "^VIX"), type == "calls")$bi.vol, col = "red", xlab = "Number of
data", ylab = "Volatility", main = "VIX's volatility for Call Option")

plot(subset(subset(DATA1, security == "^VIX"), type == "puts")$bi.vol, col = "blue", xlab = "Number of
data", ylab = "Volatility", main = "VIX's volatility for Put Option")

money.type <- vector(mode="character", length=nrow(DATA1))

for(i in 1:nrow(DATA1)){
    if(DATA1$Strike[i] <= 1.01*DATA1$equity.price[i] & DATA1$Strike[i] >= 0.99*DATA1$equity.price[i]){
        money.type[i] <- "At-the-money"
    }else if(DATA1$type[i] == "calls" & (DATA1$Strike[i] < DATA1$equity.price[i])){
        money.type[i] <- "In-the-money"
    }else if(DATA1$type[i] == "calls" & (DATA1$Strike[i] > DATA1$equity.price[i])){
        money.type[i] <- "Out-of-the-money"
    }else if(DATA1$type[i] == "puts" & (DATA1$Strike[i] > DATA1$equity.price[i])){
        money.type[i] <- "In-the-money"
    }else if(DATA1$type[i] == "puts" & (DATA1$Strike[i] < DATA1$equity.price[i])){
        money.type[i] <- "Out-of-the-money"
    }

}

DATA1 <- cbind(DATA1[,1:18], data.frame(money.type = money.type))

library("doBy")
vol.average.sum <- summaryBy(bi.vol~security+maturity+type+money.type, data=DATA1, FUN = mean)
vol.average.sum <- vol.average.sum[!(vol.average.sum$money.type == "At-the-money"),]
```

```
vol.average.sum <- vol.average.sum[!(vol.average.sum$security == "^VIX"),]

vol.atmoney <- DATA1[DATA1$money.type == "At-the-money",]
vol.atmoney$diff <- abs(vol.atmoney$Strike - vol.atmoney$equity.price)


key <- summaryBy(diff~security+type+maturity+money.type, data=vol.atmoney , FUN = min)
key$key <- paste(key$security, key$type, key$maturity, key$diff.min, sep = "-")
vol.atmoney <- vol.atmoney[,c(2, 3, 10, 11, 14, 16, 20)]
vol.atmoney$key <- paste(vol.atmoney$security, vol.atmoney$type, vol.atmoney$maturity, vol.atmoney$diff,
sep = "-")

vol.atmoney.merge <- merge(key, vol.atmoney, by="key")
vol.atmoney.merge <- vol.atmoney.merge[,6:12]

plot(subset(vol.atmoney.merge, type.y == "calls" & security.y == "AMZN")$maturity.y,
     subset(vol.atmoney.merge, type.y == "calls"& security.y == "AMZN")$bi.vol, type = "l", col="red",
ylim = c(0, 1)
     , xlab = "Maturity", ylab="Volatility", main = "AMZN's Volatility comparison for Call")
lines(subset(vol.average.sum, type == "calls" & security == "AMZN" & money.type == "In-the-
money")$maturity,
      subset(vol.average.sum, type == "calls" & security == "AMZN"& money.type == "In-the-money")$bi.vol,
col = "blue")
lines(subset(vol.average.sum, type == "calls" & security == "AMZN" & money.type == "Out-of-the-
money")$maturity,
      subset(vol.average.sum, type == "calls" & security == "AMZN"& money.type == "Out-of-the-
money")$bi.vol, col = "green")
legend("topright", c("Nearest ATM", "Average of ITM", "Average of OTM"),
       col=c("red", "blue", "green"), lty = c(1,1,1),ncol = 1, cex = 1)

plot(subset(vol.atmoney.merge, type.y == "calls" & security.y == "SPY")$maturity.y,
     subset(vol.atmoney.merge, type.y == "calls"& security.y == "SPY")$bi.vol, type = "l", col="red", ylim
= c(0, 1.5)
     , xlab = "Maturity", ylab="Volatility", main = "SPY's Volatility comparison for Call")
lines(subset(vol.average.sum, type == "calls" & security == "SPY" & money.type == "In-the-
money")$maturity,
      subset(vol.average.sum, type == "calls" & security == "SPY"& money.type == "In-the-money")$bi.vol,
col = "blue")
lines(subset(vol.average.sum, type == "calls" & security == "SPY" & money.type == "Out-of-the-
money")$maturity,
      subset(vol.average.sum, type == "calls" & security == "SPY"& money.type == "Out-of-the-
money")$bi.vol, col = "green")
legend("topright", c("Nearest ATM", "Average of ITM", "Average of OTM"),
       col=c("red", "blue", "green"), lty = c(1,1,1),ncol = 1, cex = 1)

plot(subset(vol.atmoney.merge, type.y == "puts" & security.y == "AMZN")$maturity.y,
     subset(vol.atmoney.merge, type.y == "puts"& security.y == "AMZN")$bi.vol, type = "l", col="red", ylim
= c(0, 1)
     , xlab = "Maturity", ylab="Volatility", main = "AMZN's Volatility comparison for Put")
lines(subset(vol.average.sum, type == "puts" & security == "AMZN" & money.type == "In-the-
money")$maturity,
      subset(vol.average.sum, type == "puts" & security == "AMZN"& money.type == "In-the-money")$bi.vol,
col = "blue")
lines(subset(vol.average.sum, type == "puts" & security == "AMZN" & money.type == "Out-of-the-
money")$maturity,
      subset(vol.average.sum, type == "puts" & security == "AMZN"& money.type == "Out-of-the-
money")$bi.vol, col = "green")
legend("topright", c("Nearest ATM", "Average of ITM", "Average of OTM"),
       col=c("red", "blue", "green"), lty = c(1,1,1),ncol = 1, cex = 1)

plot(subset(vol.atmoney.merge, type.y == "calls" & security.y == "SPY")$maturity.y,
     subset(vol.atmoney.merge, type.y == "calls"& security.y == "SPY")$bi.vol, type = "l", col="red", ylim
= c(0, 1.5)
     , xlab = "Maturity", ylab="Volatility", main = "SPY's Volatility comparison for Put")
lines(subset(vol.average.sum, type == "puts" & security == "SPY" & money.type == "In-the-money")$maturity,
      subset(vol.average.sum, type == "puts" & security == "SPY"& money.type == "In-the-money")$bi.vol,
col = "blue")
lines(subset(vol.average.sum, type == "puts" & security == "SPY" & money.type == "Out-of-the-
```

```
money")$maturity,
      subset(vol.average.sum, type == "puts" & security == "SPY"& money.type == "Out-of-the-
money")$bi.vol, col = "green")
legend("topright", c("Nearest ATM", "Average of ITM", "Average of OTM"),
       col=c("red", "blue", "green"), lty = c(1,1,1),ncol = 1, cex = 1)
```

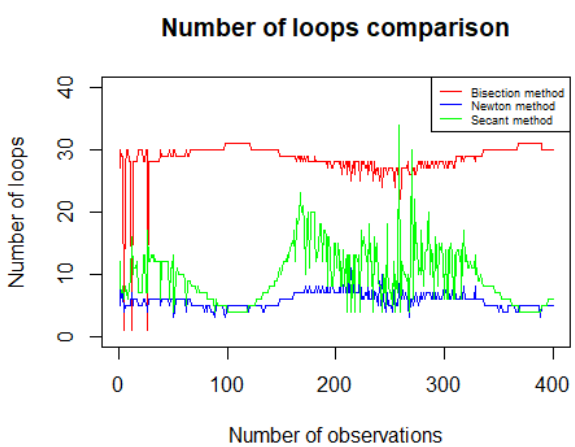
Bonus

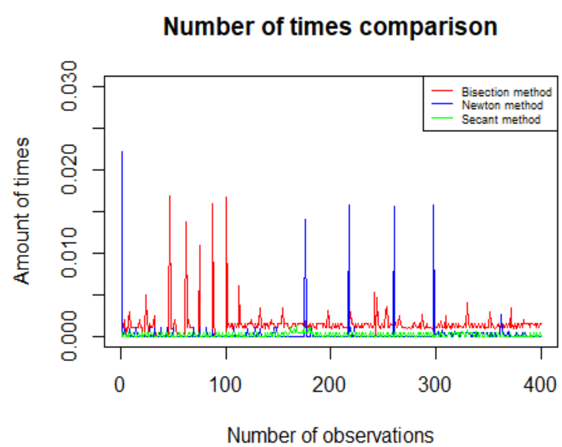
The red line represents Bisection method. The blue line represents Newton method and the green line represents the Secant method.

For the first figure, we can see that the Bisection method spends the highest number of loops, but the Newton method spend the least number of loops. It seems that Newton method is the most efficient way to find the volatility.

For the second figure, it is obvious that Bisection method consumes the highest time and Newton method spend the least.

We can conclude that Newton method is the most efficient methods among these 3 ones.





Newton method

```
Vega <- function(K, S, t, r, v){
    d1 = (log(S/K) + (r + (v^2)/2) * t)/(v*(t^(1/2)))
    dNd1=((2*pi)^(-0.5))*exp(-0.5*(d1^2))
    vega <- S*sqrt(t)*dNd1
    return(vega)
}
ImpVolNewton <- function (K, S, t, r, OptVal, type, x0 = 1, epsilon = 1e-6){
    a <- Sys.time()
    n = 1
    xn <- x0
    while(TRUE){
        if(Vega(K, S, t, r, xn) == 0 || is.nan(Vega(K, S, t, r, xn))){xn = 0;break;}
        x0 = xn
        deltax = ((OptVal - BS(K, S, t, r, x0, type))/Vega(K, S, t, r, x0))
        xn <- x0 + deltax

        if(abs(xn - x0) < epsilon){
            break;
```

```r
        }
        if(n > 10000){break;}
        n = n + 1
    }
    b <- Sys.time()
    return(c(xn, n, as.double.difftime(b-a)))
}

new.vol.matrix <- matrix(, nrow = nrow(DATA1), ncol = 3)

for(i in 1:nrow(DATA1)){
    new.vol <- ImpVolNewton(DATA1$Strike[i], DATA1$equity.price[i], DATA1$T.mat[i], DATA1$interest[i],
                            (DATA1$Bid[i] + DATA1$Ask[i])/2, DATA1$type[i])
    new.vol.matrix[i,1] <- new.vol[1]
    new.vol.matrix[i,2] <- new.vol[2]
    new.vol.matrix[i,3] <- new.vol[3]
}
DATA1 <- cbind(DATA1[,1:19], data.frame(new.vol = new.vol.matrix[,1], new.no = new.vol.matrix[,2],
new.time = new.vol.matrix[,3]))
```

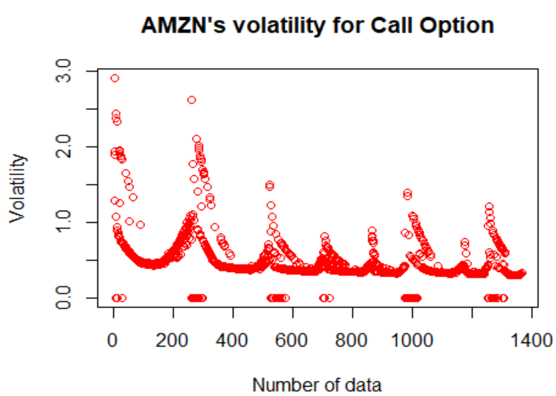
Secant method

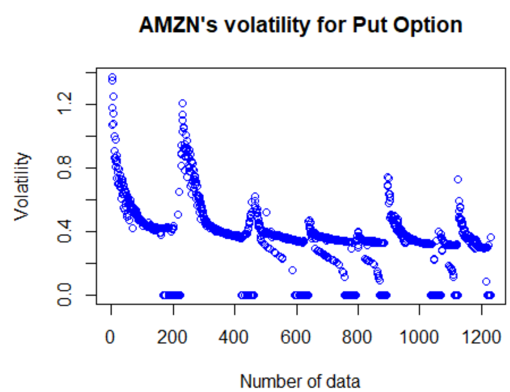
```r
ImpVolSecant <- function (K, S, t, r, OptVal, type, x0 = 0.1, x1 = 1,epsilon = 1e-6){
    a <- Sys.time()
    xn <- 0
    n = 1
    while(TRUE){
        fx1 = BS(K, S, t, r, x1, type) - OptVal
        fx0 = BS(K, S, t, r, x0, type) - OptVal
        xn <- x1 - fx1*(x1 - x0)/(fx1 - fx0)
        if(is.na(xn)){xn = 0;break;}
        if(abs(xn - x1) < epsilon){
            break;
        }
        x0 <- x1
        x1 <- xn
        if(n > 10000){break;}
        n = n + 1
    }
    b <- Sys.time()
    return(c(xn, n, as.double.difftime(b-a)))
}

secant.vol.matrix <- matrix(, nrow = nrow(DATA1), ncol = 3)
for(i in 1:nrow(DATA1)){
    secant.vol <- ImpVolSecant(DATA1$Strike[i], DATA1$equity.price[i], DATA1$T.mat[i], DATA1$interest[i],
                               (DATA1$Bid[i] + DATA1$Ask[i])/2, DATA1$type[i])
    secant.vol.matrix[i,1] <- secant.vol[1]
    secant.vol.matrix[i,2] <- secant.vol[2]
    secant.vol.matrix[i,3] <- secant.vol[3]
}

DATA1 <- cbind(DATA1[,1:22], data.frame(secant.vol = secant.vol.matrix[,1], secant.no =
secant.vol.matrix[,2], secant.time = secant.vol.matrix[,3]))

plot(DATA1[1:400,]$bi.no, type = "l", col ="red",
     xlab = "Number of observations", ylab = "Number of loops",
     main = "Number of loops comparison", ylim = c(0,40))
lines(1:400, DATA1[1:400,]$new.no, col = "blue")
lines(1:400, DATA1[1:400,]$secant.no, col = "green")
legend("topright", c("Bisection method", "Newton method", "Secant method"),
       col=c("red", "blue", "green"), lty = c(1,1,1),ncol = 1, cex = 0.55)

plot(DATA1[1:400,]$bi.time, type = "l", col ="red",
     xlab = "Number of observations", ylab = "Amount of times",
     main = "Number of times comparison", ylim = c(0,0.03))
lines(1:400, DATA1[1:400,]$new.time, col = "blue")
lines(1:400, DATA1[1:400,]$secant.time, col = "green")
```

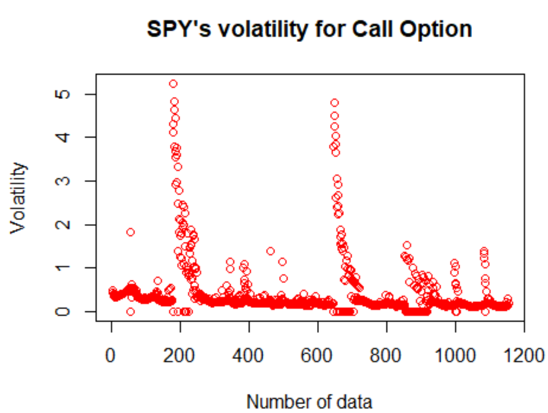
```
legend("topright", c("Bisection method", "Newton method", "Secant method"),
       col=c("red", "blue", "green"), lty = c(1,1,1),ncol = 1, cex = 0.55)
```

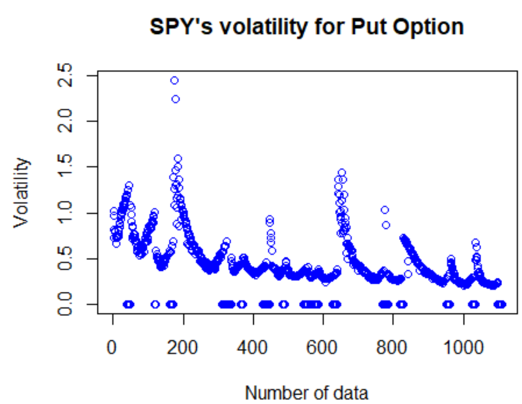
7) Present a table reporting the implied volatility values obtained for every maturity, option type and stock.

The figures below show all volatilities of all maturities. It is clear for all sets of volatilities that it is getting decreased over the time. In other words, the nearer maturity has higher volatility than the others, consequently.

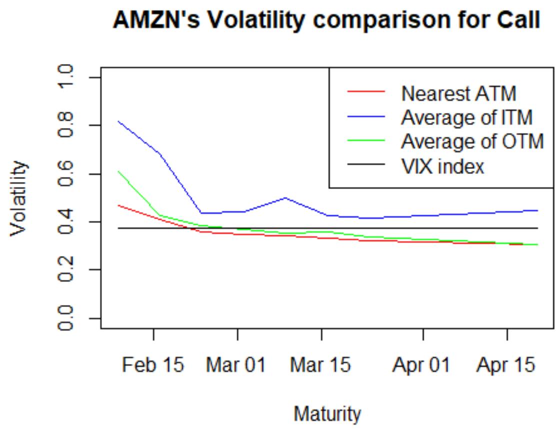


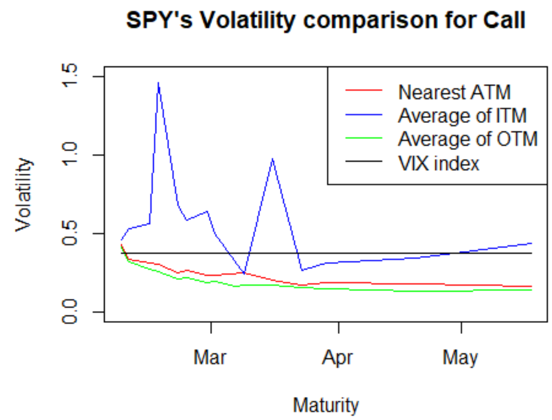


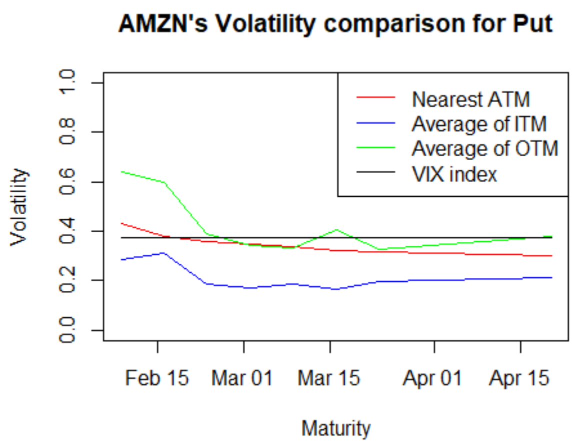


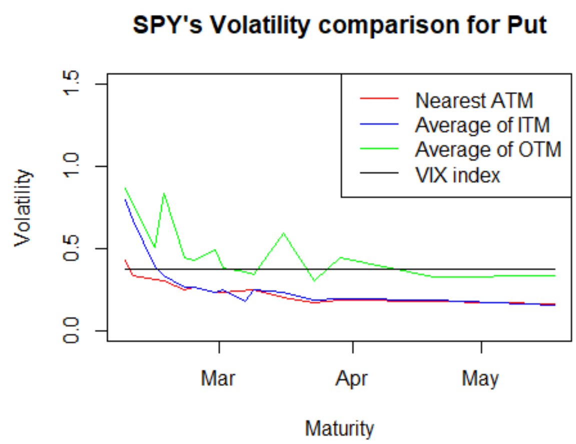


With 37.32% of VIX volatility index, it seems very accurate to measure the overall of market's volatility. As we can see from figures below, VIX index seems to be in the middle of all types of volatilities.









	security	maturity	money.type	vol.type	vix
23	AMZN	2018-02-09	At-the-money	0.4290215	0.3732
24	AMZN	2018-02-09	In-the-money	0.5509658	0.3732
25	AMZN	2018-02-09	Out-of-the-money	0.6203584	0.3732
26	AMZN	2018-02-16	At-the-money	0.3786435	0.3732
27	AMZN	2018-02-16	In-the-money	0.5338424	0.3732
28	AMZN	2018-02-16	Out-of-the-money	0.5193894	0.3732
29	AMZN	2018-02-23	At-the-money	0.3476733	0.3732
30	AMZN	2018-02-23	In-the-money	0.3377573	0.3732
31	AMZN	2018-02-23	Out-of-the-money	0.3862286	0.3732
32	AMZN	2018-03-02	At-the-money	0.1911277	0.3732
33	AMZN	2018-03-02	In-the-money	0.3203864	0.3732
34	AMZN	2018-03-02	Out-of-the-money	0.3549382	0.3732
35	AMZN	2018-03-09	At-the-money	0.2106761	0.3732
36	AMZN	2018-03-09	In-the-money	0.3053751	0.3732
37	AMZN	2018-03-09	Out-of-the-money	0.3463716	0.3732
38	AMZN	2018-03-16	At-the-money	0.3226089	0.3732
39	AMZN	2018-03-16	In-the-money	0.3454484	0.3732
40	AMZN	2018-03-16	Out-of-the-money	0.3868333	0.3732
41	AMZN	2018-03-23	At-the-money	0.3146739	0.3732
42	AMZN	2018-03-23	In-the-money	0.2661430	0.3732
43	AMZN	2018-03-23	Out-of-the-money	0.3352572	0.3732
44	AMZN	2018-04-20	At-the-money	0.2956221	0.3732
45	AMZN	2018-04-20	In-the-money	0.3814592	0.3732
46	AMZN	2018-04-20	Out-of-the-money	0.3529092	0.3732

	security	maturity	money.type	vol.type	vix
50	SPY	2018-02-09	At-the-money	0.3379756	0.3732
51	SPY	2018-02-09	In-the-money	0.6323445	0.3732
52	SPY	2018-02-09	Out-of-the-money	0.4464231	0.3732
53	SPY	2018-02-14	At-the-money	0.3119759	0.3732
54	SPY	2018-02-14	In-the-money	0.4020903	0.3732
55	SPY	2018-02-14	Out-of-the-money	0.3116754	0.3732
56	SPY	2018-02-16	At-the-money	0.2867319	0.3732
57	SPY	2018-02-16	In-the-money	0.9745588	0.3732
58	SPY	2018-02-16	Out-of-the-money	0.5796537	0.3732
59	SPY	2018-02-21	At-the-money	0.2488367	0.3732
60	SPY	2018-02-21	In-the-money	0.3193816	0.3732
61	SPY	2018-02-21	Out-of-the-money	0.2418494	0.3732
62	SPY	2018-02-23	At-the-money	0.2282080	0.3732
63	SPY	2018-02-23	In-the-money	0.3257342	0.3732
64	SPY	2018-02-23	Out-of-the-money	0.2690718	0.3732
65	SPY	2018-02-28	At-the-money	0.2311013	0.3732
66	SPY	2018-02-28	In-the-money	0.2854426	0.3732
67	SPY	2018-02-28	Out-of-the-money	0.3166096	0.3732
68	SPY	2018-03-02	At-the-money	0.2260834	0.3732
69	SPY	2018-03-02	In-the-money	0.2781073	0.3732
70	SPY	2018-03-02	Out-of-the-money	0.2268044	0.3732
71	SPY	2018-03-07	At-the-money	0.3034511	0.3732
72	SPY	2018-03-07	In-the-money	0.1760364	0.3732
73	SPY	2018-03-07	Out-of-the-money	0.1897588	0.3732
74	SPY	2018-03-09	At-the-money	0.2290727	0.3732
75	SPY	2018-03-09	In-the-money	0.2489031	0.3732
76	SPY	2018-03-09	Out-of-the-money	0.1966332	0.3732
77	SPY	2018-03-16	At-the-money	0.1967437	0.3732
78	SPY	2018-03-16	In-the-money	0.7348587	0.3732
79	SPY	2018-03-16	Out-of-the-money	0.4286633	0.3732

Showing 27 to 58 of 69 entries

Code:

```
vol.atmoney.merge$vix <- 37.32/100
plot(subset(vol.atmoney.merge, type.y == "calls" & security.y == "AMZN")$maturity.y,
     subset(vol.atmoney.merge, type.y == "calls"& security.y == "AMZN")$bi.vol, type = "l", col="red",
ylim = c(0, 1)
     , xlab = "Maturity", ylab="Volatility", main = "AMZN's Volatility comparison for Call")
lines(subset(vol.average.sum, type == "calls" & security == "AMZN" & money.type == "In-the-
money")$maturity,
      subset(vol.average.sum, type == "calls" & security == "AMZN"& money.type == "In-the-money")$bi.vol,
col = "blue")
lines(subset(vol.average.sum, type == "calls" & security == "AMZN" & money.type == "Out-of-the-
money")$maturity,
      subset(vol.average.sum, type == "calls" & security == "AMZN"& money.type == "Out-of-the-
money")$bi.vol, col = "green")
lines(subset(vol.atmoney.merge, type.y == "calls" & security.y == "AMZN")$maturity.y,
      subset(vol.atmoney.merge, type.y == "calls"& security.y == "AMZN")$vix, col = "black")
legend("topright", c("Nearest ATM", "Average of ITM", "Average of OTM", "VIX index"),
       col=c("red", "blue", "green", "black"), lty = c(1,1,1,1),ncol = 1, cex = 1)

plot(subset(vol.atmoney.merge, type.y == "calls" & security.y == "SPY")$maturity.y,
     subset(vol.atmoney.merge, type.y == "calls"& security.y == "SPY")$bi.vol, type = "l", col="red", ylim
= c(0, 1.5)
     , xlab = "Maturity", ylab="Volatility", main = "SPY's Volatility comparison for Call")
lines(subset(vol.average.sum, type == "calls" & security == "SPY" & money.type == "In-the-
money")$maturity,
      subset(vol.average.sum, type == "calls" & security == "SPY"& money.type == "In-the-money")$bi.vol,
col = "blue")
lines(subset(vol.average.sum, type == "calls" & security == "SPY" & money.type == "Out-of-the-
money")$maturity,
```

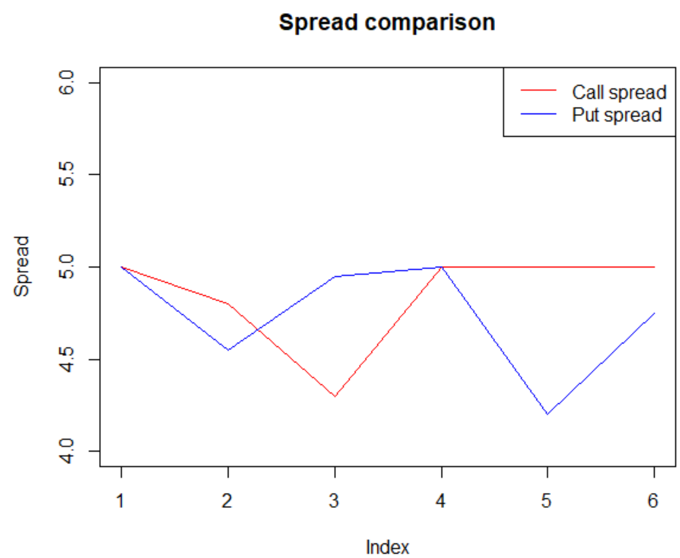
```
      subset(vol.average.sum, type == "calls" & security == "SPY"& money.type == "Out-of-the-
money")$bi.vol, col = "green")
lines(subset(vol.atmoney.merge, type.y == "calls" & security.y == "SPY")$maturity.y,
      subset(vol.atmoney.merge, type.y == "calls"& security.y == "SPY")$vix, col = "black")
legend("topright", c("Nearest ATM", "Average of ITM", "Average of OTM", "VIX index"),
       col=c("red", "blue", "green", "black"), lty = c(1,1,1,1),ncol = 1, cex = 1)

plot(subset(vol.atmoney.merge, type.y == "calls" & security.y == "SPY")$maturity.y,
     subset(vol.atmoney.merge, type.y == "calls"& security.y == "SPY")$bi.vol, type = "l", col="red", ylim
= c(0, 1.5)
     , xlab = "Maturity", ylab="Volatility", main = "SPY's Volatility comparison for Put")
lines(subset(vol.average.sum, type == "puts" & security == "SPY" & money.type == "In-the-money")$maturity,
      subset(vol.average.sum, type == "puts" & security == "SPY"& money.type == "In-the-money")$bi.vol,
col = "blue")
lines(subset(vol.average.sum, type == "puts" & security == "SPY" & money.type == "Out-of-the-
money")$maturity,
      subset(vol.average.sum, type == "puts" & security == "SPY"& money.type == "Out-of-the-
money")$bi.vol, col = "green")
lines(subset(vol.atmoney.merge, type.y == "puts" & security.y == "SPY")$maturity.y,
      subset(vol.atmoney.merge, type.y == "puts"& security.y == "SPY")$vix, col = "black")
legend("topright", c("Nearest ATM", "Average of ITM", "Average of OTM", "VIX index"),
       col=c("red", "blue", "green", "black"), lty = c(1,1,1,1),ncol = 1, cex = 1)
```

8) Put-Call parity

We can find the Put-Call parity property when the strike price is equal to the stock price.

```
> df.parity.merge.match[,-c(1,8,10,11:13,16,18:21)]
    security.x Strike.x Bid.x Ask.x maturity.x type.x equity.price.x Bid.y Ask.y type.y parity.call parity.put
704       AMZN     1390 29.35 34.35 2018-02-09  calls           1390 26.40 31.40   puts        TRUE       TRUE
705       AMZN     1390 45.00 49.80 2018-02-16  calls           1390 40.45 45.00   puts        TRUE       TRUE
706       AMZN     1390 51.45 55.75 2018-02-23  calls           1390 49.15 54.10   puts        TRUE       TRUE
707       AMZN     1390 58.60 63.60 2018-03-02  calls           1390 56.45 61.45   puts        TRUE       TRUE
708       AMZN     1390 66.05 71.05 2018-03-09  calls           1390 62.75 66.95   puts        TRUE       TRUE
709       AMZN     1390 71.30 76.30 2018-03-16  calls           1390 66.15 70.90   puts        TRUE       TRUE
```

The spreads between Bid and Ask prices for both Call option and Put option are very close to each other.



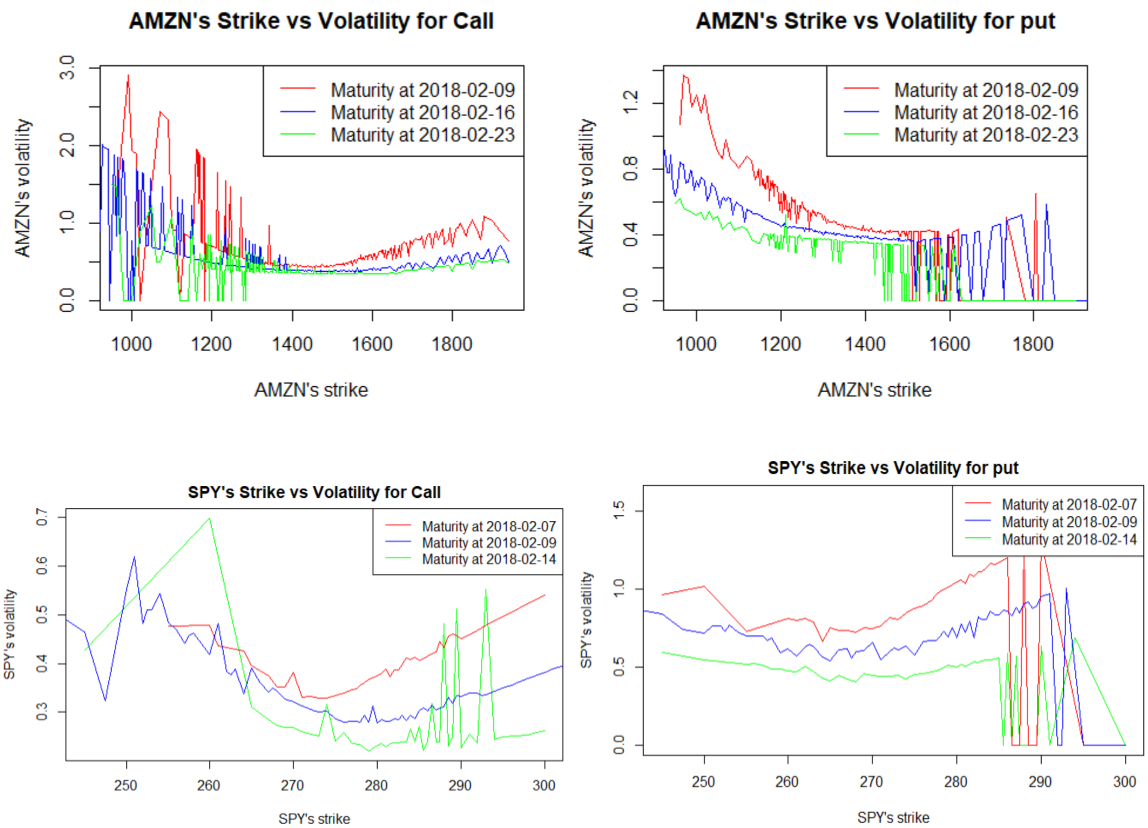
Code:

```
checkCallPutParity<-function(K, S, t, r, v){
    call <- BS(K,S,t,r,v,"calls")
    put <- BS(K,S,t,r,v,"puts")
    parity <- FALSE

    if(abs((abs(call-put)-abs(S-((K*exp(r*t))))))<.01){parity <- TRUE}

    return(parity)
}

df.parity.call <- subset(DATA1, type == "calls")
df.parity.call <- df.parity.call[,-c(1, 4, 5, 8, 12, 17:ncol(DATA1))]
df.parity.call$key <- paste(df.parity.call$security, df.parity.call$Strike, df.parity.call$maturity, sep =
"-")

df.parity.put <- subset(DATA1, type == "puts")
df.parity.put <- df.parity.put[,-c(1, 4, 5, 8, 12, 17:ncol(DATA1))]
df.parity.put$key <- paste(df.parity.put$security, df.parity.put$Strike, df.parity.put$maturity, sep = "-
")

df.parity.merge <- merge(df.parity.call, df.parity.put, by="key")
```

```r
parity.matrix <- matrix(, nrow = nrow(df.parity.merge), ncol = 2)

for(i in 1:nrow(df.parity.merge)){

    parity1 <- checkCallPutParity(df.parity.merge$Strike.x[i], df.parity.merge$equity.price.x[i],
df.parity.merge$interest.x[i], df.parity.merge$T.mat.x[i], df.parity.merge$bi.vol.x[i])
    parity2 <- checkCallPutParity(df.parity.merge$Strike.x[i], df.parity.merge$equity.price.x[i],
df.parity.merge$interest.x[i], df.parity.merge$T.mat.x[i], df.parity.merge$bi.vol.y[i])
    parity.matrix[i, 1] <- parity1
    parity.matrix[i, 2] <- parity2
}
df.parity <- data.frame(parity.call = parity.matrix[, 1], parity.put = parity.matrix[, 2])

df.parity.merge <- cbind(df.parity.merge[,1:23],  df.parity)
subset(df.parity.merge, (parity.call == TRUE | parity.put == TRUE)& security.x != "^VIX")
```

9) Plot of implied volatilities versus strike K

Mostly, it shows that the nearer maturity, the higher volatility becomes. The nearest maturity representing by red line has the highest volatility. The second nearest maturity representing by blue line has the second order and so on.

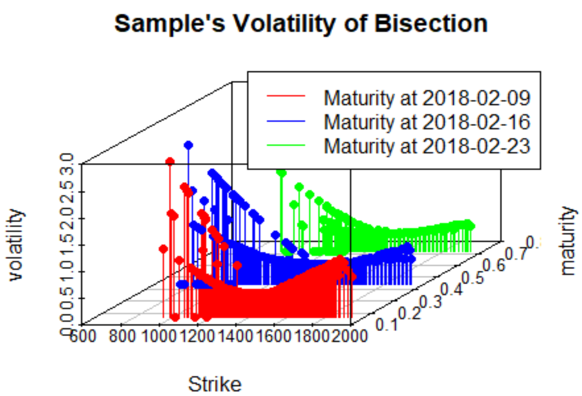


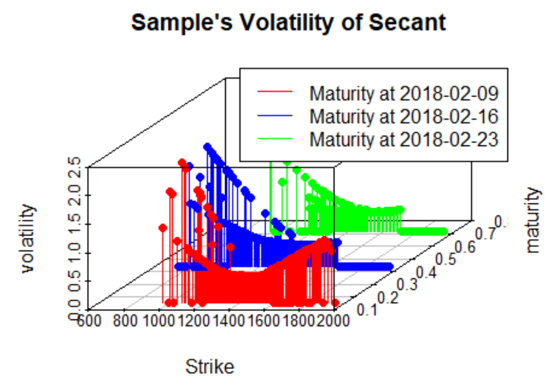
Code:

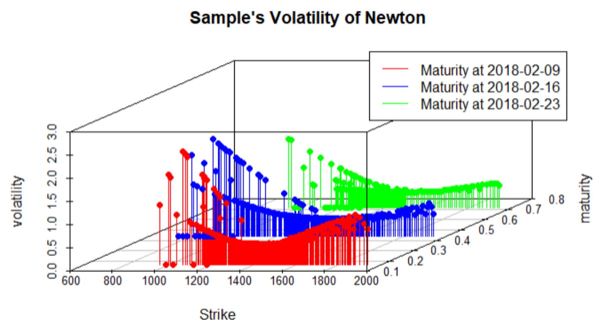
```
AMZN <- subset(DATA1, security == "AMZN")

plot(subset(subset(AMZN, type == "calls"),
            maturity == "2018-02-09")$Strike,
     subset(subset(AMZN, type == "calls"),  maturity == "2018-02-09")$bi.vol,
     col = "red", xlab = "AMZN's strike", ylab = "AMZN's volatility", main = "AMZN's Strike vs Volatility
for Call",
     type = "l")

lines(subset(subset(AMZN, type == "calls"),
             maturity == "2018-02-16")$Strike,
      subset(subset(AMZN, type == "calls"),  maturity == "2018-02-16")$bi.vol, col = "blue")

lines(subset(subset(AMZN, type == "calls"),
             maturity == "2018-02-23")$Strike,
      subset(subset(AMZN, type == "calls"),  maturity == "2018-02-23")$bi.vol, col = "green")

legend("topright", c("Maturity at 2018-02-09", "Maturity at 2018-02-16", "Maturity at 2018-02-23"),
       col=c("red", "blue", "green"), lty = c(1,1,1),ncol = 1, cex = 1)
```

```r
plot(subset(subset(AMZN, type == "puts"),
            maturity == "2018-02-09")$Strike,
     subset(subset(AMZN, type == "puts"),  maturity == "2018-02-09")$bi.vol,
     col = "red", xlab = "AMZN's strike", ylab = "AMZN's volatility", main = "AMZN's Strike vs Volatility
for put",
     type = "l")

lines(subset(subset(AMZN, type == "puts"),
             maturity == "2018-02-16")$Strike,
      subset(subset(AMZN, type == "puts"),  maturity == "2018-02-16")$bi.vol, col = "blue")

lines(subset(subset(AMZN, type == "puts"),
             maturity == "2018-02-23")$Strike,
      subset(subset(AMZN, type == "puts"),  maturity == "2018-02-23")$bi.vol, col = "green")

legend("topright", c("Maturity at 2018-02-09", "Maturity at 2018-02-16", "Maturity at 2018-02-23"),
       col=c("red", "blue", "green"), lty = c(1,1,1),ncol = 1, cex = 1)

SPY <- subset(DATA1, security == "SPY")

plot(subset(subset(SPY, type == "calls"),
            maturity == "2018-02-14")$Strike,
     subset(subset(SPY, type == "calls"),  maturity == "2018-02-14")$bi.vol,
     col = "green", xlab = "SPY's strike", ylab = "SPY's volatility", main = "SPY's Strike vs Volatility
for Call",
     type = "l")

lines(subset(subset(SPY, type == "calls"),
             maturity == "2018-02-07")$Strike,
      subset(subset(SPY, type == "calls"),  maturity == "2018-02-07")$bi.vol, col = "red")

lines(subset(subset(SPY, type == "calls"),
             maturity == "2018-02-09")$Strike,
      subset(subset(SPY, type == "calls"),  maturity == "2018-02-09")$bi.vol, col = "blue")

legend("topright", c("Maturity at 2018-02-07", "Maturity at 2018-02-09", "Maturity at 2018-02-14"),
       col=c("red", "blue", "green"), lty = c(1,1,1),ncol = 1, cex = 1)

plot(subset(subset(SPY, type == "puts"),
            maturity == "2018-02-14")$Strike,
     subset(subset(SPY, type == "puts"),  maturity == "2018-02-14")$bi.vol,
     col = "green", xlab = "SPY's strike", ylab = "SPY's volatility", main = "SPY's Strike vs Volatility
for put",
     type = "l", ylim = c(0, 1.6))

lines(subset(subset(SPY, type == "puts"),
             maturity == "2018-02-07")$Strike,
      subset(subset(SPY, type == "puts"),  maturity == "2018-02-07")$bi.vol, col = "red")

lines(subset(subset(SPY, type == "puts"),
             maturity == "2018-02-09")$Strike,
      subset(subset(SPY, type == "puts"),  maturity == "2018-02-09")$bi.vol, col = "blue")

legend("topright", c("Maturity at 2018-02-07", "Maturity at 2018-02-09", "Maturity at 2018-02-14"),
col=c("red", "blue", "green"), lty = c(1,1,1),ncol = 1, cex = 1)
```

Bonus: Create a 3D plot of the same implied








```
library("scatterplot3d")

data1.call.bi <- data.frame(Strike = subset(subset(AMZN, type == "calls"),
                                       maturity == "2018-02-09")$Strike,
                       maturity = subset(subset(AMZN, type == "calls"),
                                         maturity == "2018-02-09")$T.mat*10,
                       volatility = subset(subset(AMZN, type == "calls"),
                                           maturity == "2018-02-09")$bi.vol,
                       pcolor = "red")

data2.call.bi <- data.frame(Strike = subset(subset(AMZN, type == "calls"),
                                       maturity == "2018-02-16")$Strike,
                       maturity = subset(subset(AMZN, type == "calls"),
                                         maturity == "2018-02-16")$T.mat*10,
                       volatility= subset(subset(AMZN, type == "calls"),
                                          maturity == "2018-02-16")$bi.vol,
                       pcolor = "blue")

data3.call.bi <- data.frame(Strike = subset(subset(AMZN, type == "calls"),
                                       maturity == "2018-02-23")$Strike,
                       maturity = subset(subset(AMZN, type == "calls"),
                                         maturity == "2018-02-23")$T.mat*10,
                       volatility = subset(subset(AMZN, type == "calls"),
                                           maturity == "2018-02-23")$bi.vol,
                       pcolor = "green")

df.data.call.bi <- rbind(rbind(data1.call.bi, data2.call.bi), data3.call.bi)
scatterplot3d(df.data.call.bi, pch = 19, type="h", color = pcolor, main = "Sample's Volatility of
Bisection")
legend("topright", c("Maturity at 2018-02-09", "Maturity at 2018-02-16", "Maturity at 2018-02-23"),
col=c("red", "blue", "green"), lty = c(1,1,1),ncol = 1, cex = 1)
```

```r
data1.call.new <- data.frame(Strike = subset(subset(AMZN, type == "calls"),
                                       maturity == "2018-02-09")$Strike,
                       maturity = subset(subset(AMZN, type == "calls"),
                                         maturity == "2018-02-09")$T.mat*10,
                       volatility = subset(subset(AMZN, type == "calls"),
                                           maturity == "2018-02-09")$new.vol,
                       pcolor = "red")

data2.call.new <- data.frame(Strike = subset(subset(AMZN, type == "calls"),
                                       maturity == "2018-02-16")$Strike,
                       maturity = subset(subset(AMZN, type == "calls"),
                                         maturity == "2018-02-16")$T.mat*10,
                       volatility= subset(subset(AMZN, type == "calls"),
                                          maturity == "2018-02-16")$new.vol,
                       pcolor = "blue")

data3.call.new <- data.frame(Strike = subset(subset(AMZN, type == "calls"),
                                       maturity == "2018-02-23")$Strike,
                       maturity = subset(subset(AMZN, type == "calls"),
                                         maturity == "2018-02-23")$T.mat*10,
                       volatility = subset(subset(AMZN, type == "calls"),
                                           maturity == "2018-02-23")$new.vol,
                       pcolor = "green")
df.data.call.new <- rbind(rbind(data1.call.new, data2.call.new), data3.call.new)
scatterplot3d(df.data.call.new, pch = 19, type="h", color = pcolor, main = "Sample's Volatility of
Newton")
legend("topright", c("Maturity at 2018-02-09", "Maturity at 2018-02-16", "Maturity at 2018-02-23"),
col=c("red", "blue", "green"), lty = c(1,1,1),ncol = 1, cex = 1)

data1.call.secant <- data.frame(Strike = subset(subset(AMZN, type == "calls"),
                                       maturity == "2018-02-09")$Strike,
                       maturity = subset(subset(AMZN, type == "calls"),
                                         maturity == "2018-02-09")$T.mat*10,
                       volatility = subset(subset(AMZN, type == "calls"),
                                           maturity == "2018-02-09")$secant.vol,
                       pcolor = "red")
data2.call.secant <- data.frame(Strike = subset(subset(AMZN, type == "calls"),
                                       maturity == "2018-02-16")$Strike,
                       maturity = subset(subset(AMZN, type == "calls"),
                                         maturity == "2018-02-16")$T.mat*10,
                       volatility= subset(subset(AMZN, type == "calls"),
                                          maturity == "2018-02-16")$secant.vol,
                       pcolor = "blue")
data3.call.secant <- data.frame(Strike = subset(subset(AMZN, type == "calls"),
                                       maturity == "2018-02-23")$Strike,
                       maturity = subset(subset(AMZN, type == "calls"),
                                         maturity == "2018-02-23")$T.mat*10,
                       volatility = subset(subset(AMZN, type == "calls"),
                                           maturity == "2018-02-23")$secant.vol,
                       pcolor = "green")
df.data.call.secant <- rbind(rbind(data1.call.secant, data2.call.secant), data3.call.secant)
scatterplot3d(df.data.call.secant, pch = 19, type="h", color = pcolor, main = "Sample's Volatility of
Secant")
legend("topright", c("Maturity at 2018-02-09", "Maturity at 2018-02-16", "Maturity at 2018-02-23"),
col=c("red", "blue", "green"), lty = c(1,1,1),ncol = 1, cex = 1)
```

10) (Greeks) Calculate the derivatives of the call option price with respect to S (Delta), and sigma (Vega) and the second derivative with respect to S (Gamma)

For the sample of Geeks' approximations, the 2 methods can make very close result for Delta and Gamma but slightly close for Vega.

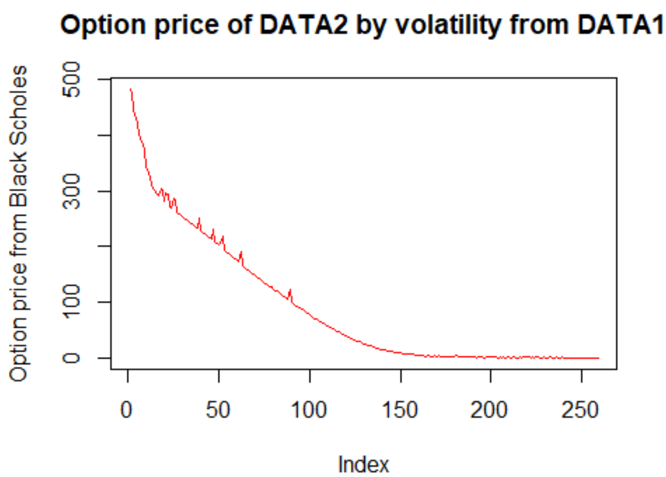
	delta.diff	delta.partial	error	gamma.diff	gamma.partial	error	vega.diff	vega.partial	error
1	0.9906941	0.9908857	0.000	0.0001104449	9.031307e-05	0	3.357593	4.404779	1.047
2	0.8660735	0.8661991	0.000	0.0004228415	4.259530e-04	0	34.987971	37.871562	2.884
3	0.9290046	0.9292864	0.000	0.0003988558	3.848413e-04	0	21.078299	23.845084	2.767
4	0.9274031	0.9277024	0.000	0.0004156420	4.014456e-04	0	21.463472	24.256998	2.794
5	1.0000000	1.0000000	0.000	NaN	0.000000e+00	NaN	0.000000	0.000000	0.000
6	0.9863893	0.9867496	0.000	0.0001847480	1.540974e-04	0	4.791448	6.115572	1.324
7	0.8423378	0.8425261	0.000	0.0005627586	5.701628e-04	0	39.255034	42.271791	3.017
8	0.8388343	0.8390325	0.000	0.0005843583	5.925321e-04	0	39.854197	42.885657	3.031
9	0.8353703	0.8355796	0.000	0.0006073135	6.162927e-04	0	40.437897	43.484221	3.046
10	0.9794214	0.9800475	0.001	0.0003021039	2.588195e-04	0	7.019668	8.692103	1.672
11	0.9793836	0.9800609	0.001	0.0003149828	2.690565e-04	0	7.028608	8.698756	1.670
12	1.0000000	1.0000000	0.000	NaN	0.000000e+00	NaN	0.000000	0.000000	0.000

```
BS.delta <- function(S,K,r,t,v) {
    d1 <- (log(S/K)+(r+0.5*(v^2))*t)/(v*sqrt(t))
    return (pnorm(d1))
}

BS.delta.partial<-function(S,K,r,t,v, type = "calls") {
    diff <- 0.01
    delta.partial <- (BS(K, S + diff, t, r, v, type) - BS(K, S, t, r, v, type))/(diff)
    return(delta.partial)
}

delta.compre <- matrix(, nrow = 100, ncol = 2)

for(i in 1:100){
    delt.diff <- BS.delta(DATA1$equity.price[i], DATA1$Strike[i], DATA1$interest[i], DATA1$T.mat[i],
DATA1$bi.vol[i])
    delt.partial <- BS.delta.partial(DATA1$equity.price[i], DATA1$Strike[i], DATA1$interest[i],
DATA1$T.mat[i], DATA1$bi.vol[i])
    delta.compre[i, 1] <- delt.diff
    delta.compre[i, 2] <- delt.partial
}

df.delta.compare <- data.frame(delta.diff = delta.compre[,1], delta.partial = delta.compre[,2])
df.delta.compare$error <- round(df.delta.compare$delta.partial - df.delta.compare$delta.diff, digits = 3)

BS.gamma <- function(S,K,r,t,v){
    d1<-(log(S/K)+(r+0.5*(v^2))*t)/(v*sqrt(t))
    gamma <-(2*pi)^(-0.5)*exp(-0.5*d1^2)/(S*v*sqrt(t))
    return(gamma)
}
```

```r
BS.gamma.partial<-function(S,K,r,t,v, type = "calls") {
    diff <- 0.01
    gamma.partial<-(BS(S+diff,K,r,t,v,type)-2*BS(S,K,r,t,v,type)+ BS(S-diff,K,r,t,v,type))/(diff*diff)
    return (gamma.partial)
}

gamma.compre <- matrix(, nrow = 100, ncol = 2)

for(i in 1:100){
    gam.diff <- BS.gamma(DATA1$equity.price[i], DATA1$Strike[i], DATA1$interest[i], DATA1$T.mat[i],
DATA1$bi.vol[i])
    gam.partial <- BS.gamma.partial(DATA1$equity.price[i], DATA1$Strike[i], DATA1$interest[i],
DATA1$T.mat[i], DATA1$bi.vol[i])
    gamma.compre[i, 1] <- gam.diff
    gamma.compre[i, 2] <- gam.partial
}

df.gamma.compare <- data.frame(gamma.diff = gamma.compre[,1], gamma.partial = gamma.compre[,2])
df.gamma.compare$error <- round(df.gamma.compare$gamma.partial - df.gamma.compare$gamma.diff, digits = 3)

BS.vega <- function(S, K, t, r, v){
    d1 = (log(S/K) + (r + (v^2)/2) * t)/(v*(t^(1/2)))
    dNd1=((2*pi)^(-0.5))*exp(-0.5*(d1^2))
    vega <- S*sqrt(t)*dNd1
    return(vega)
}

BS.vega.partial<-function(S,K,r,t,v, type = "calls") {
    diff <- 0.01
    vega.partial <- (BS(K, S, t, r, v + diff, type) - BS(K, S, t, r, v, type))/(diff)
    return(vega.partial)
}

vega.compare <- matrix(, nrow = 100, ncol = 2)

for(i in 1:100){
    veg.diff <- BS.vega(DATA1$equity.price[i], DATA1$Strike[i], DATA1$interest[i], DATA1$T.mat[i],
DATA1$bi.vol[i])
    veg.partial <- BS.vega.partial(DATA1$equity.price[i], DATA1$Strike[i], DATA1$interest[i],
DATA1$T.mat[i], DATA1$bi.vol[i])
    vega.compare[i, 1] <- veg.diff
    vega.compare[i, 2] <- veg.partial
}

df.vega.compare <- data.frame(vega.diff = vega.compare[,1], vega.partial = vega.compare[,2])
df.vega.compare$error <- round(df.vega.compare$vega.partial - df.vega.compare$vega.diff, digits = 3)

df.greek <- cbind(cbind(df.delta.compare, df.gamma.compare),df.vega.compare)
```

11) Use the second dataset DATA2 and the implied volatility you calculated from DATA1 to calculate the option price by Black-Scholes formula.

From the "bs.op.price" in DATA2, it is the option price that we calculated from DATA1's volatility ("bi.vol" field).



	X	security	Strike	Bid	Ask	maturity	type	time.stamp	T.mat	equity.price	interest	bi.vol	bs.op.price
1	AMZN180209C00960000	AMZN	960.0	417.35	422.35	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	1.2940185	483.0995940
2	AMZN180209C00990000	AMZN	990.0	438.60	443.60	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	2.9160458	474.7754479
3	AMZN180209C01000000	AMZN	1000.0	399.50	404.50	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	1.9432365	447.2765460
4	AMZN180209C01010000	AMZN	1010.0	389.50	394.50	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	1.8969364	437.2568184
5	AMZN180209C01020000	AMZN	1020.0	284.50	289.40	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	0.0000000	423.0208785
6	AMZN180209C01040000	AMZN	1040.0	334.55	339.55	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	1.0765185	403.1306372
7	AMZN180209C01070000	AMZN	1070.0	358.65	363.65	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	2.4451559	394.6060892
8	AMZN180209C01080000	AMZN	1080.0	348.70	353.70	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	2.3889035	384.6067866
9	AMZN180209C01090000	AMZN	1090.0	338.65	343.65	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	2.3306118	374.5374842
10	AMZN180209C01100000	AMZN	1100.0	289.40	294.40	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	0.9375783	343.1814556
11	AMZN180209C01110000	AMZN	1110.0	329.35	334.35	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	0.9006418	333.1697554
12	AMZN180209C01120000	AMZN	1120.0	117.10	120.75	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	0.0000000	323.0386150
13	AMZN180209C01130000	AMZN	1130.0	245.00	250.00	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	0.8654736	313.2068604
14	AMZN180209C01140000	AMZN	1140.0	235.10	240.10	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	0.8390979	303.2113285
15	AMZN180209C01145000	AMZN	1145.0	230.10	235.10	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	0.8235227	298.2089095
16	AMZN180209C01150000	AMZN	1150.0	225.20	230.20	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	0.8169823	293.2241999
17	AMZN180209C01152500	AMZN	1152.5	286.90	291.90	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	0.8002360	290.7051709
18	AMZN180209C01160000	AMZN	1160.0	269.60	274.60	2018-02-09	calls	2/6/2018	0.01190476	1442.84	0.0149	1.9570791	304.8570101

Code:

```
DATA2 <- read.csv("data2.csv")
DATA2$maturity <- as.Date(DATA2$maturity, "%m/%d/%Y")
DATA2 <- DATA2[DATA2$maturity < as.Date("2018-06-01"),]
DATA2.merge <- merge(DATA2[,-c(4,5,8,9)], DATA1[,c(1,16)], by="X")
bs.vector <- vector(mode="numeric", length=nrow(DATA2.merge))
```

```
for(i in 1:nrow(DATA2.merge)){
    bs.vec <- BS(DATA2.merge$Strike[i], DATA2.merge$equity.price[i],
                    DATA2.merge$T.mat[i], DATA2.merge$interest[i], DATA2.merge$bi.vol[i],
DATA2.merge$type[i])
    bs.vector[i] <- bs.vec
}

DATA2.merge <- cbind(DATA2.merge[, 1:12], bs.op.price = bs.vector)
plot(subset(DATA2.merge, security == "AMZN" & maturity == "2018-02-09" & type == "calls")$bs.op.price,
type ="l", col = "red", ylab = "Option price from Black Scholes", main = "Option price of DATA2 by
volatility from DATA1")
```

Part 3: Numerical Integration of real-valued functions.

Implement the trapezoidal and the Simpson's quadrature rules to numerically approximate the indefinite integral

$$f(x) = \begin{cases} \dfrac{\sin(x)}{x}, & \text{for } x \neq 0, \\ 1, & \text{for } x = 0. \end{cases}$$

From [-10^6, 10^6] interval and N equals to 10^6, we got the result of $2\text{x}10^{-5}$.

```
> estimate <- data.frame(trapezoid.appx = trapezoid(f, 10e-6, 10e6), simpson.appx = simpson(f, 10e-6, 10e6))
> estimate
  trapezoid.appx simpson.appx
1          2e-05        2e-05
```

Code:

```r
f <- function(x){

    ifelse(x == 0, return(1), return(sin(x)/x))
}


xn <- function(a, n, N){
    return(-a+n*(2*a/N))
}

trapezoid <- function(f, a, n) {
    if (is.function(f) == FALSE) {
        stop('f must be a function with one parameter (variable)')
    }
    h <- (xn(a, n, n) - xn(a, 0, n)) / n
    j <- 1:(n - 1)
    xj <- xn(a, 0, n) + j * h
    approx <- (h / 2) * (f(xn(a, 0, n)) + 2 * sum(sapply(xj, f)) + f(xn(a, n, n)))
    return(approx)
}

simpson <- function(f, a, n) {
    if (is.function(f) == FALSE) {
        stop('f must be a function with one parameter (variable)')
    }
    h <- (xn(a, n, n) - xn(a, 0, n)) / n
    xj <- seq.int(xn(a, 0, n), xn(a, n, n), length.out = n + 1)
    xj <- xj[-1]
    xj <- xj[-length(xj)]
    approx <- (h / 3) * (f(xn(a, 0, n)) + 2 * sum(sapply(xj[seq.int(2, length(xj), 2)], f)) + 4 *
sum(sapply(xj[seq.int(1, length(xj), 2)], f)) + f(xn(a, n, n)))
    return(approx)
}

estimate <- data.frame(trapezoid.appx = trapezoid(f, 10e-6, 10e6), simpson.appx = simpson(f, 10e-6, 10e6))
estimate

##   trapezoid.appx simpson.appx
## 1          2e-05        2e-05
```

2) Compute the truncation error

For both functions, it is clear that when we increase the number of “n”, the error keeps decreasing because we decrease the size of estimating subinterval and make it more accurate.

However, when we increase the number of “a”, the error gets higher because we increase the size of integrating interval meaning that we increase the volume of integration. Therefore, small “a” means small volume. The error will be smaller in the same way. In other words, there is a small chance to have error.

```
Trapezoidal_Error(a=1000, n=12,f)

##        n Integration Result        Error
## 1     4         499.477896 4.963363e+02
## 2     8         248.294607 2.451530e+02
## 3    16         122.662075 1.195205e+02
## 4    32          59.837753 5.669616e+01
## 5    64          28.423693 2.528210e+01
## 6   128          15.707497 1.256590e+01
## 7   256           9.420146 6.278553e+00
## 8   512           3.142469 8.760280e-04
## 9  1024           3.140848 7.444872e-04
## 10 2048           3.140557 1.035706e-03
## 11 4096           3.140489 1.104007e-03
## 12 8192           3.140472 1.120826e-03
```

```
Trapezoidal_Error(a=10^6, n=12,f)

##        n Integration Result       Error
## 1     4        500000.1807 499997.0391
## 2     8        249998.7412 249995.5996
## 3    16        125000.7822 124997.6406
## 4    32         62501.8989  62498.7573
## 5    64         31249.4310  31246.2894
## 6   128         15623.1600  15620.0184
## 7   256          7813.1385   7809.9969
## 8   512          3905.0023   3901.8607
## 9  1024          1950.9326   1947.7910
## 10 2048           977.0351    973.8935
## 11 4096           486.9472    483.8056
## 12 8192           241.9031    238.7615
```

```
Simpson_Error(a=1000, n=12,f)

##        n Integration Result        Error
## 1     4         332.361568 3.292200e+02
## 2     8         164.566844 1.614253e+02
## 3    16          80.784564 7.764297e+01
## 4    32          38.896312 3.575472e+01
## 5    64          17.952340 1.481075e+01
## 6   128          11.468766 8.327173e+00
## 7   256           7.324362 4.182769e+00
## 8   512           1.049910 2.091683e+00
## 9  1024           3.140308 1.284659e-03
## 10 2048           3.140460 1.132779e-03
## 11 4096           3.140466 1.126774e-03
## 12 8192           3.140466 1.126432e-03
```

```
Simpson_Error(a=10^6, n=12,f)

##        n Integration Result       Error
## 1     4        333333.6909 333330.5493
## 2     8        166664.9280 166661.7864
## 3    16         83334.7959  83331.6543
## 4    32         41668.9378  41665.7962
## 5    64         20831.9417  20828.8001
## 6   128         10414.4030  10411.2614
## 7   256          5209.7980   5206.6564
## 8   512          2602.2902   2599.1487
## 9  1024          1299.5761   1296.4345
## 10 2048           652.4026    649.2610
## 11 4096           323.5846    320.4430
## 12 8192           160.2217    157.0801
```

Code:

```
Trapezoidal_Error<-function(a,n,f)
{
    table = matrix(0, nrow=n, ncol=3, dimnames=list(
        c(1:n), c('n', 'Integration Result', 'Error')))
    for(i in 1:n) {
        table[i,1] = ifelse(1 == i, yes=4, no=2*table[i-1,'n'])
        table[i,2] = trapezoid(f, a, table[i,'n'])
        table[i,3] = abs(table[i,2] - pi )
    }
    return(table)
```

```
}

Simpson_Error<-function(a,n,f)
{
    table = matrix(0, nrow=n, ncol=3, dimnames=list(
        c(1:n), c('n', 'Integration Result', 'Error')))
    for(i in 1:n) {
        table[i,1] = ifelse(1 == i, yes=4, no=2*table[i-1,'n'])
        table[i,2] = simpson(f, a, table[i,'n'])
        table[i,3] = abs(table[i,2]- pi )
    }
    return(table)
}
```

3) ensure the convergence of the numerical algorithms

From the results, the Trapezoid's rule is more efficient than Simpson's rule. As we can see, Trapezoid's rule converges faster than Simpson's rule. In the same way, the error from Trapezoid's rule is less than Simpson's rule.

```
Trapezoidal_converge(a=(10^6), n=20,f)

##                n Integration Result            Error
##       5.242880e+05       3.141594e+00      1.250210e-06
## [1] "Number of Iterations= 18"
```

```
Simpson_converge(a=(10^6), n=20,f)

##                n Integration Result            Error
##       1.048576e+06       3.141591e+00      2.107140e-06
## [1] "Number of Iterations= 19"
```

Code:

```
Trapezoidal_converge<-function(a,n,f,tolerance=(10^-4))
{
    table = matrix(0, nrow=n, ncol=3, dimnames=list(
        c(1:n), c('n', 'Integration Result', 'Error')))
    for(i in 1:n) {
        table[i,1] = ifelse(1 == i, yes=4, no=2*table[i-1,'n'])
        table[i,2] = trapezoid(f, a, table[i,'n'])
        table[i,3] = abs(table[i,2]-pi)
        if(table[i,3] < tolerance  && table[i,1]>2){
            print(table[i,])
            print(paste("Number of Iterations=",i))
            break
        }
    }
}
Simpson_converge<-function(a,n,f,tolerance=(10^-4))
{
    table = matrix(0, nrow=n, ncol=3, dimnames=list(
        c(1:n), c('n', 'Integration Result', 'Error')))
    for(i in 1:n) {
        table[i,1] = ifelse(1 == i, yes=4, no=2*table[i-1,'n'])
        table[i,2] = simpson(f, a,table[i,'n'])
        table[i,3] = abs(table[i,2]- pi)
        if(table[i,3] < tolerance  && table[i,1]>2){
            print(table[i,])
            print(paste("Number of Iterations=",i))
            break
        }

    }
}
```

4) Use the trapezoidal rule and Simpson's rule to approximate g(x) function

$$g(x) = 1 + e^{-x^2} \cos(8x^{2/3}).$$

For the estimation, we have result as 1.95878.

```
trapezoid2(f2, 0, 2, 10e6)

## [1] 1.958798
```

```
simpson2(f2, 0, 2, 10e6)

## [1] 1.958798
```

For the error estimation, we got the different result. The Simson's rule is more efficient than Trapezoid 's rule. As we can see, the error from Simson's rule is less than Trapezoid's rule.

```
Trapezoidal_Error2(a=0,b=2, n=10,f2)

##        n Integration Result        Error
## 1      4           2.327114 3.683163e-01
## 2      8           2.018952 6.015431e-02
## 3     16           1.969490 1.069222e-02
## 4     32           1.960819 2.021301e-03
## 5     64           1.959190 3.922999e-04
## 6    128           1.958875 7.687753e-05
## 7    256           1.958813 1.507856e-05
## 8    512           1.958801 2.922648e-06
## 9   1024           1.958799 5.301590e-07
## 10  2048           1.958798 5.976056e-08
```

```
Simpson_Error2(a=0,b=2, n=10,f2)

##        n Integration Result        Error
## 1      4           2.284302 3.255041e-01
## 2      8           1.916232 4.256635e-02
## 3     16           1.953003 5.795140e-03
## 4     32           1.957929 8.690053e-04
## 5     64           1.958647 1.507007e-04
## 6    128           1.958770 2.826325e-05
## 7    256           1.958793 5.521092e-06
## 8    512           1.958797 1.129323e-06
## 9   1024           1.958798 2.673374e-07
## 10  2048           1.958798 9.703891e-08
```

```
Trapezoidal_converge2(0, 2, 20, f2)

##                    n Integration Result        Error
##         1.280000e+02       1.958875e+00 7.687753e-05
## [1] "Number of Iterations= 6"
```

```
Simpson_converge2(0, 2, 20, f2)

##                    n Integration Result        Error
##         1.280000e+02       1.958770e+00 2.826325e-05
## [1] "Number of Iterations= 6"
```

Code:

```
f2 <- function(x){

    return(1+exp(-x^2)*cos(8*x^(2/3)))
}
trapezoid2 <- function(f, a, b, n) {
    if (is.function(f) == FALSE) {
        stop('f must be a function with one parameter (variable)')
```

```
    }
    h <- (b - a) / n
    j <- 1:(n - 1)
    xj <- a + j * h
    approx <- (h / 2) * (f(a) + 2 * sum(f(xj)) + f(b))
    return(approx)
}

simpson2 <- function(f, a, b, n) {
    if (is.function(f) == FALSE) {
        stop('f must be a function with one parameter (variable)')
    }
    h <- (b - a) / n
    xj <- seq.int(a, b, length.out = n + 1)
    xj <- xj[-1]
    xj <- xj[-length(xj)]
    approx <- (h / 3) * (f(a) + 2 * sum(f(xj[seq.int(2, length(xj), 2)])) + 4 * sum(f(xj[seq.int(1,
length(xj), 2)])) + f(b))
    return(approx)
}

Trapezoidal_Error2<-function(a,b,n,f)
{
    table = matrix(0, nrow=n, ncol=3, dimnames=list(
        c(1:n), c('n', 'Integration Result', 'Error')))
    for(i in 1:n) {
        table[i,1] = ifelse(1 == i, yes=4, no=2*table[i-1,'n'])
        table[i,2] = trapezoid2(f, a, b, table[i,'n'])
        table[i,3] = abs(table[i,2] - integrate(f, a, b)$value)
    }
    return(table)
}

Simpson_Error2<-function(a,b,n,f)
{
    table = matrix(0, nrow=n, ncol=3, dimnames=list(
        c(1:n), c('n', 'Integration Result', 'Error')))
    for(i in 1:n) {
        table[i,1] = ifelse(1 == i, yes=4, no=2*table[i-1,'n'])
        table[i,2] = simpson2(f, a, b, table[i,'n'])
        table[i,3] = abs(table[i,2]- integrate(f, a, b)$value)
    }
    return(table)
}

Trapezoidal_converge2<-function(a,b,n,f,tolerance=10^-4)
{
    table = matrix(0, nrow=n, ncol=3, dimnames=list(
        c(1:n), c('n', 'Integration Result', 'Error')))
    for(i in 1:n) {
        table[i,1] = ifelse(1 == i, yes=4, no=2*table[i-1,'n'])
        table[i,2] = trapezoid2(f,a, b, table[i,'n'])
        table[i,3] = abs(table[i,2]-integrate(f, a, b)$value)
        if(table[i,3] < tolerance  && table[i,1]>2){
            print(table[i,])
            print(paste("Number of Iterations=",i))
            break
        }
    }
}

Simpson_converge2<-function(a,b,n,f,tolerance=10^-4)
{
    table = matrix(0, nrow=n, ncol=3, dimnames=list(
        c(1:n), c('n', 'Integration Result', 'Error')))
    for(i in 1:n) {
        table[i,1] = ifelse(1 == i, yes=4, no=2*table[i-1,'n'])
        table[i,2] = simpson2(f, a, b, table[i,'n'])
        table[i,3] = abs(table[i,2]- integrate(f, a, b)$value)
```

```
        if(table[i,3] < tolerance  && table[i,1]>2){
            print(table[i,])
            print(paste("Number of Iterations=",i))
            break
        }

    }
}
```