The final report of Financial Modeling and its Application on R

The analysis on Black-Scholes model option pricing

Yulei Zhang

Yukuan Li

Shanghai Jiao Tong University

Shanghai, P.R.China

Jun. 5st, 2016

Contents

Chapter 1 Basic Analysis on Data

1.1 Basic statistics

The basic data of S&P index, such as close and open price, are obtained through Yahoo website, which is implemented with the aid of R -package *quantmod*. The basic trends and volumes in recent six years of S&P 500 index is shown in the OHLC chart as following in Figure 1-1.

The origin data is with respect to the prices instead of the returns Therefore, we derive the returns in arithmetic and log scale with different period respectively shown as Figure 1-2.

Then we obtain several parameters to describe the returns resorting to the function *basicStats()* belonging to the package *fBasics*. shown in Table 1-1. Seeing Table 1-1, we could

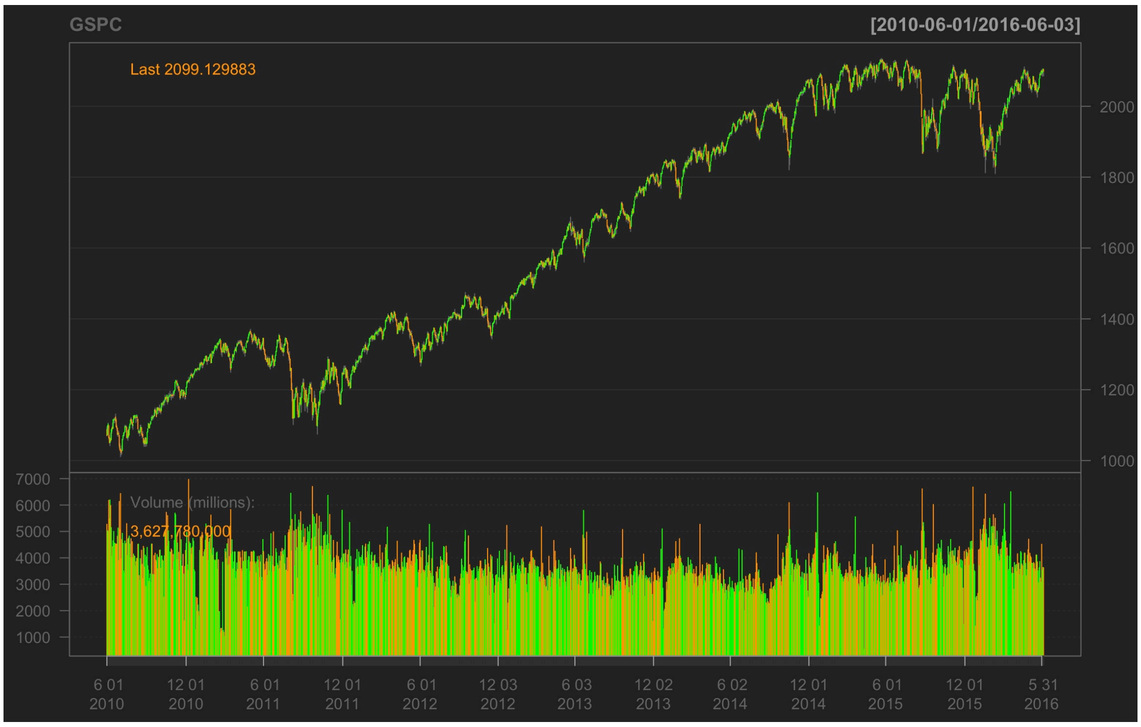


Figure 1-1: The basic plot of S&P 500 index.

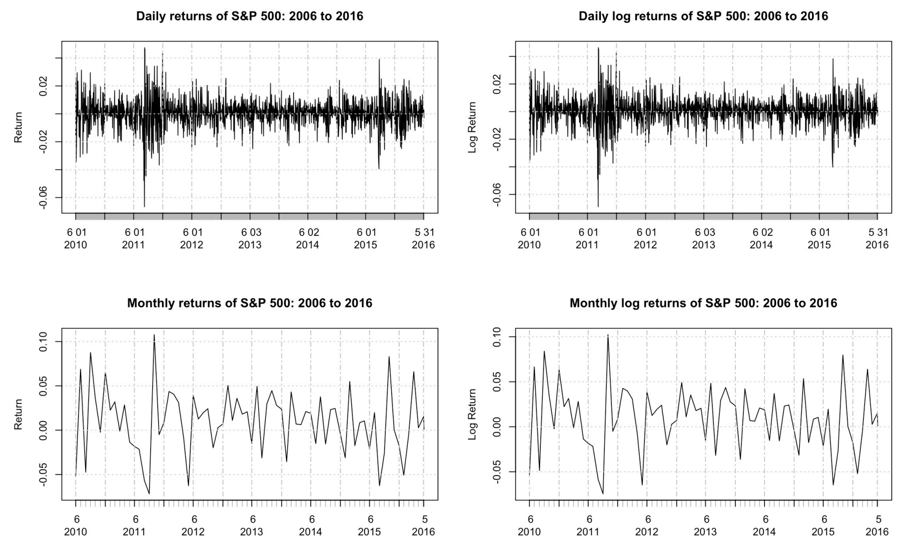


Figure 1-2: Returns of S&P500 Index in arithmetic and log scale with different period(monthly and daily).

Table 1-1: The basic statistics summary for S&P 500 Index

	Daily Returns	Monthly Returns
Mean	0.000483	0.009698
Variance	0.000097	0.001321
Skewness	-0.338293	0.042594
Kurtosis	3.959204	-0.002401

conclude that the condition of symmetry are quite different. Skewness being negative, daily returns have a tendency of leaning to the right side which is absolutely opposite to the monthly's. The detailed test are mentioned in the next section.

1.2 Return Distribution

That whether the returns are normal distributed are of great importance. We can roughly judge whether the data are normal distributed through the density of different range. The histogram are attained to described as Figure 1-3.

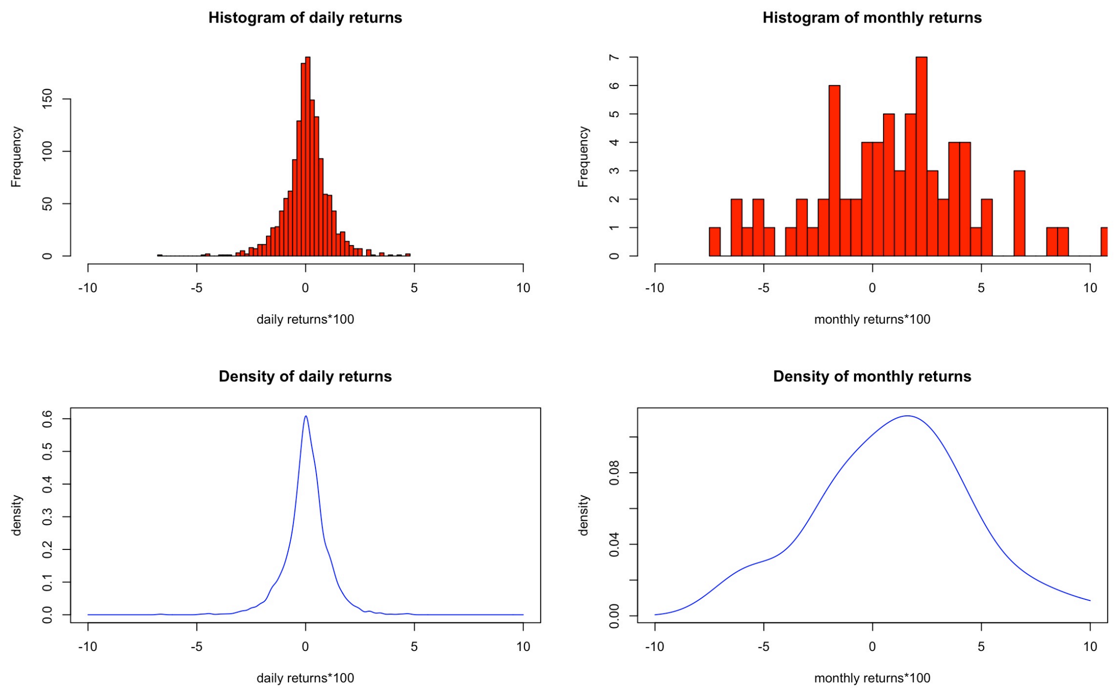


Figure 1-3: The various returns plots of S&P500 Index.

Table 1-2: The Jarque-Bera method for S&P 500 Index

	p-value	χ^2
Daily Returns	$< 2.2 \times 10^{-16}$	1022.3871
Monthly Returns	0.9786	0.0432

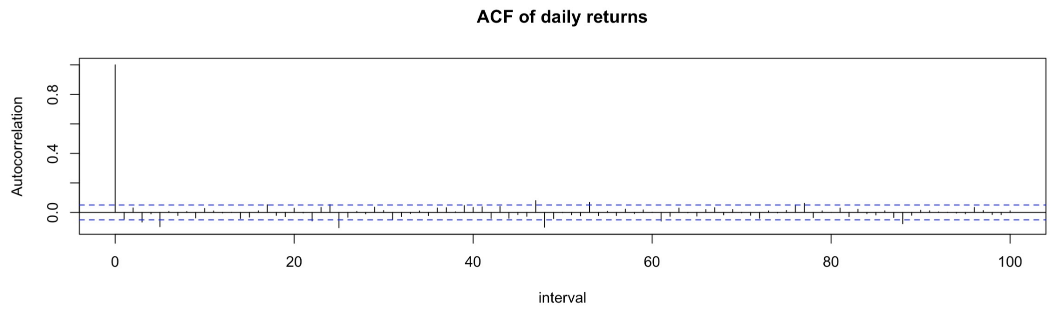
From the picture, daily return looks like normal distributed perfectly, while monthly data distribute dispersedly relatively. However, a detailed test is necessary. Here, we adopt Jarque-Bera method. The results are as following in Table 1-2. Surely, we could conclude that the null hypothesis of zero mean return cannot be rejected at the 5% level(even smaller). But, that the monthly returns are normal distributed is unconvinced.

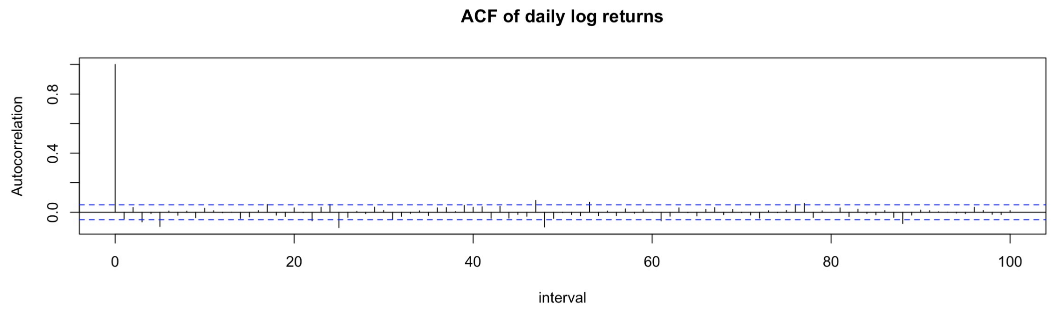
Table 1-3: The summary of various test on S&P500 Index.

	Method	Max Interval(days)	p-value	χ^2	Auto-correlated
Daily Return	Box-Pierce test	100	1.198×10^{-6}	181.36	No
	Box-Ljung test	100	3.282×10^{-7}	186.77	No
Monthly Return	Box-Pierce test	12	0.8	7.807	Yes
	Box-Ljung test	12	0.7397	8.563	Yes

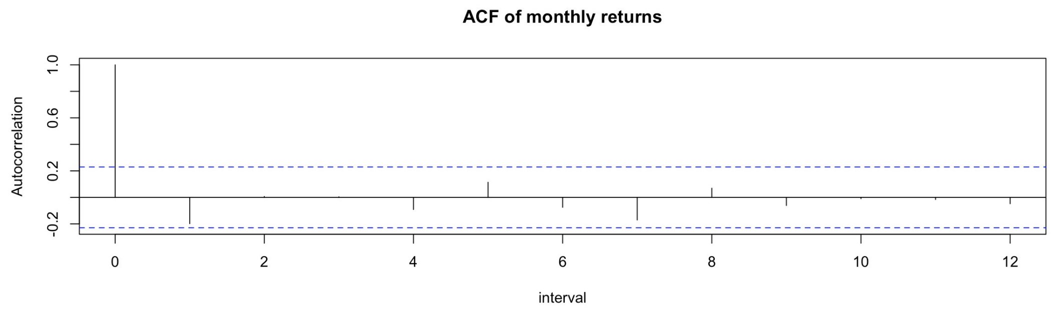
1.3 Sample Auto-correlation Function

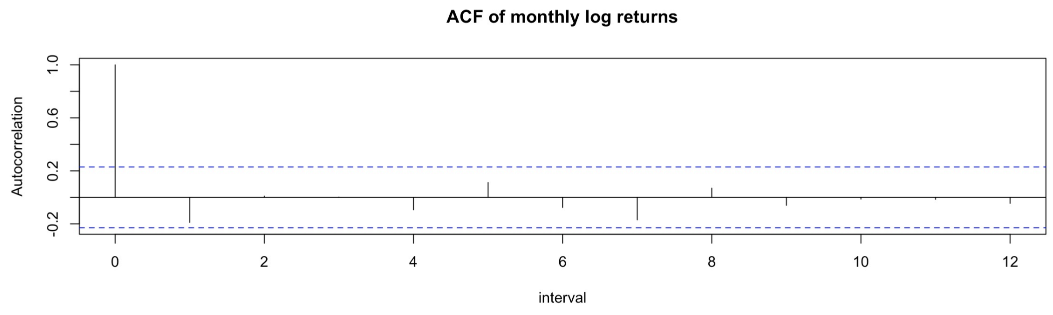
Existence of serial correlations implies that the return is predictable, indicating market inefficiency. So, correlations between the variable of interest and its past values become the focus of linear time series analysis. In this section we obtain the ACF of different intervals and make a test that whether the S&P 500 index returns are auto-correlated. Both of the returns are seemed to be not auto-correlated at the selected intervals, as shown in Figure 1-4. However, two methods of test are used to obtain accurate results as Table 1-3. The results show that monthly returns have auto-correlated property.





(a) The ACF on daily returns plots of S&P500 Index.





(b) The ACF on monthly returns plots of S&P500 Index.

Figure 1-4: The ACF plots of S&P500 Index.

Chapter 2 Garch Modelling

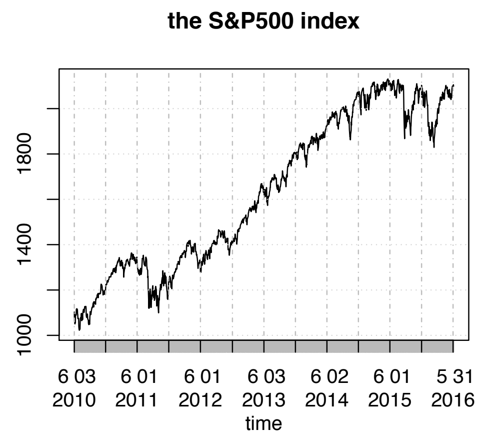
In order to find the volatility used to calculate the theoratical Black-Scholes equation, Garch model is applied in our analysis. In our case,*fGarch* and *rugarch* Package are required.

As mentioned before, S&P500 index is the underlying asset we used. Figure 2-1(a) has given a breif plot for S&P500 index. It is obvious that the S&P500 index has a significant rise from year 2011 to year 2014, data of which are abandoned because the rise may mislead the volatility calculation. Only data in the period from Jun 2014 to Jun 2016 are invovled. Then log return of S&P500 Index is calculated, but in percentage, as shown in Figure 2-1(b). So volatility will be divided by a hundred in calculation later. Some test need to be exercised on the log return of S&P500 Index.

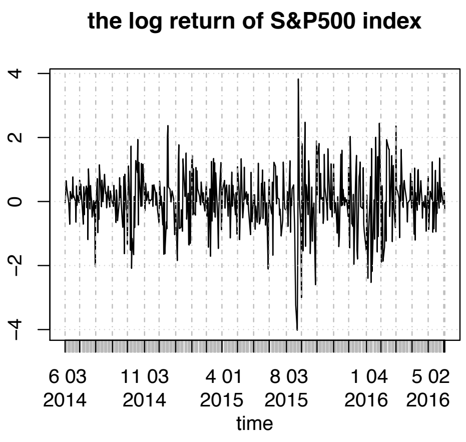
```
Box-Ljung test
data:  log return for S&P500 Index
X-squared = 6.4047, df = 12, p-value = 0.8943
```

```
Box-Ljung test
data:  squared log return for S&P500 Index
X-squared = 227.41, df = 12, p-value < 2.2e-16
```

Box-Ljung test is the first test, of which the null hypothesis is that there exists autocorrelation for samples. From the Box-Ljung test, p-value of log return is significantly larger than 0.10, which means there exists strong autocorrelation for the log return of S&P500 itself. On the contrary, the p-value of squared log return is significantly smaller than 0.01, which shows no autocorrelation. The reason why the log return of S&P500 itself is strongly correlated may result from the strong correlation bewteen the underlying strocks. But that is not the main point here, so more discussion should be in other place.. ARCH LM-test is also tested because it is necessary to examine whether the data are feasible to be modelled through *GARCH*. The null



(a) The index plots for S&P500.



(b) The log return for S&P500 Index in percentage.

Figure 2-1: S&P500

hypothesis is that there is no ARCH effect. We can see clearly the p-value of ARCH LM-test is approximately zero so that null hypothesis is rejected, which means there is strong ARCH effect for log return. So we suppose *GARCH* is valid for this log return.

```
ARCH LM-test; Null hypothesis: no ARCH effects
data:  log return for S&P500 Index
Chi-squared = 91.472, df = 12, p-value = 2.565e-14
```

2.1 standard *Garch*(1, 1) Model

Two methods are applied here in order to obtain the volatility. First one is using *fGarch* package. Basically, the simplest standard *Garch* model is used. From the fitting result, we can see the all the p-value for the fitting parameters are significant, which shows the good result of fitting. Also we have tried other standard *Garch* model, adjusting paramters, but not very much big difference. So the simplest, but also effective *Garch* Model is used.

$$\sigma_t^2 = 0.05140 + 0.18217a_t^2 + 0.75791\sigma_t^2 \quad (2\text{-}1)$$

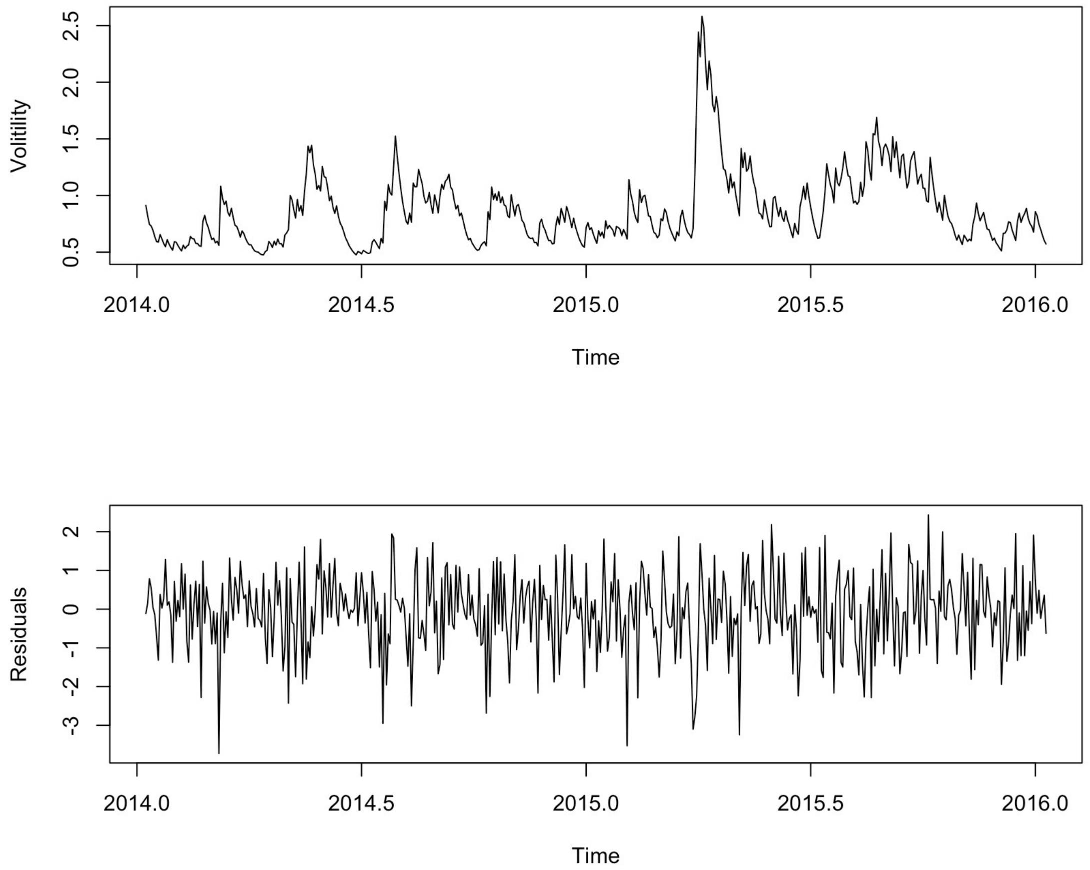


Figure 2-2: The plot of volatility and residual through standard *Garch*(1, 1) model.

It is easy to find the volatility distribution through fitted *Garch*$(1,1)$ model. The result of volatility is shown in Figure 2-2, and the residual is also given for reference. Only showing the graph is meaningless, but later on the comparsion bewteen different *Garch* model will be shown.

```
garchFit(formula = ~garch(1, 1), data = rtd.SP, trace = FALSE)
Mean and Variance Equation:
data ~ garch(1, 1)
Coefficient(s):
mu       omega     alpha1     beta1
0.057736  0.051401  0.182171  0.757908
Std. Errors:
based on Hessian
Error Analysis:
Estimate  Std.        Error    t-value Pr(>|t|)
mu       0.05774     0.03255    1.774  0.07612 .
omega    0.05140     0.01832    2.805  0.00503 **
alpha1   0.18217     0.04056    4.491 7.08e-06 ***
beta1    0.75791     0.04677   16.204  < 2e-16 ***
---
Signif.:  0  '***'  0.001  '**'  0.01  '*'  0.05  '.'  0.1
Log Likelihood:
-618.1989    normalized:  -1.221737
```

2.2 *eGarch* Model

The second method is using *rugarch* package. Because there are many other *Garch* model, so we have tried another so-called *eGarch* model. Here is one thing to mention is that in our *eGarch* model, I have set μ to be zero, because the previous analysis has given the p-value of μ_t is 0.548323. The equation for *eGarch* Model is as listed.

$$\ln \sigma_t^2 = -0.044639 - 0.385051\frac{|a_{t-1}^2| + 0.146297a_{t-1}^2}{\sigma_{t-1}} + 0.913192\ln \sigma_{t-1}^2 \qquad (2\text{-}2)$$

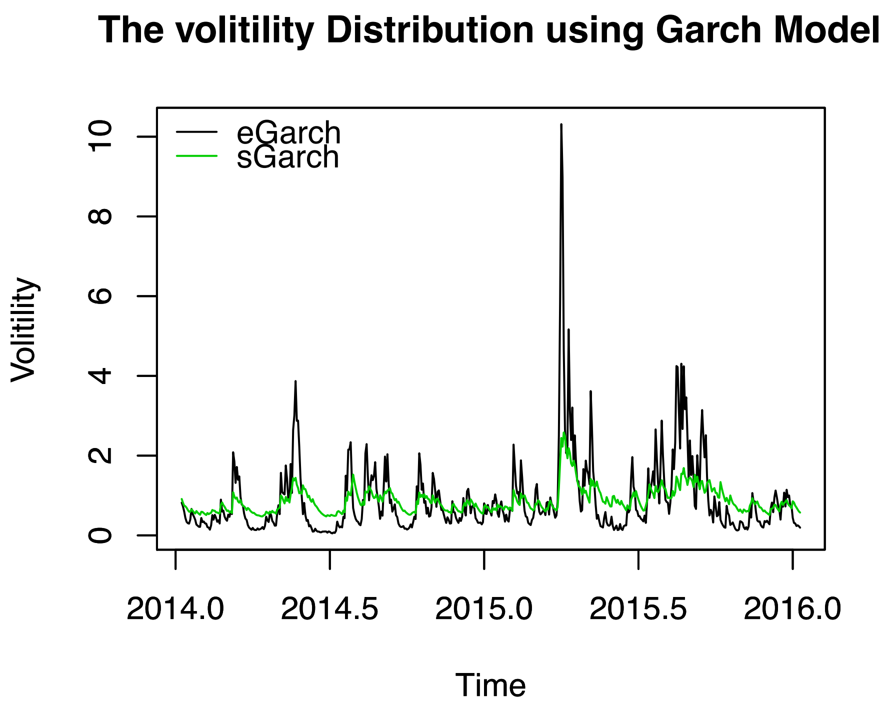


Figure 2-3: The comparsion bewteen *eGarch* model and *sGarch* model.

Table 2-1: The final result for volatility.

	$sGarch(1, 1)$	$eGarch$
volatility	0.8740411	0.8935609

```
-----------------------------------
GARCH Model: eGARCH(1,1)
Mean Model: ARFIMA(0,0,0)
Distribution: norm

Optimal Parameters
------------------------------------
        Estimate   Std. Error  t-value Pr(>|t|)
mu      0.000000          NA       NA       NA
omega  -0.044639    0.011764  -3.7947 0.000148
alpha1 -0.385051    0.045862  -8.3958 0.000000
beta1   0.913192    0.015207  60.0494 0.000000
gamma1  0.146297    0.044791   3.2662 0.001090
Robust Standard Errors:
Estimate  Std. Error  t value Pr(>|t|)
mu      0.000000          NA       NA       NA
omega  -0.044639    0.012563  -3.5532 0.000381
alpha1 -0.385051    0.057194  -6.7324 0.000000
beta1   0.913192    0.017914  50.9752 0.000000
gamma1  0.146297    0.037156   3.9374 0.000082
LogLikelihood : -581.5497
```

The comparison bewteen two methods has been also plotted in Figure 2-3. The results seem to be quite disordered. But due to the different models used, the plots are quite accaptable. The main point is to get the volatility. The final result is shown in Table 2-1. Notice that for convenience, we have taken the mean of volatility to be the one used in Black-Scholes equation.

Chapter 3 Option Pricing

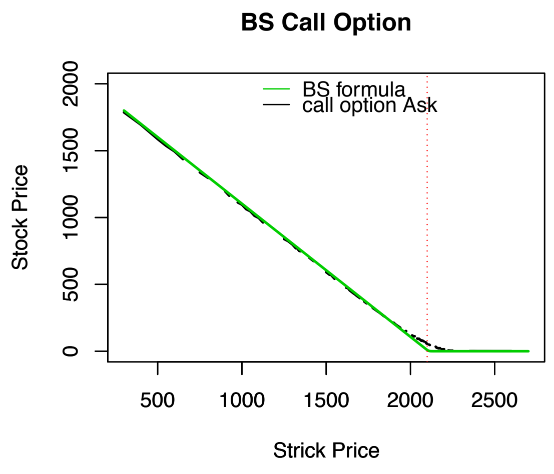
3.1 Option Pricing through Black-Scholes Model

Just with the volatility calculated in Chap.2, it is possible to calculate the option price using Black-Scholes formula. The BS formula has been given in *fOptions* package. There are totolly two set of options. One is short expired data options, the other is long. The short expired date is the last day of June, and the long expired date is the last day of September, which just differs by 3 months. And the risk-free rate is chosen as the return of U.S. 10-year Treasury Securities. There are not many difficults to calculate the option price. The results have been given in Figure 3-1.

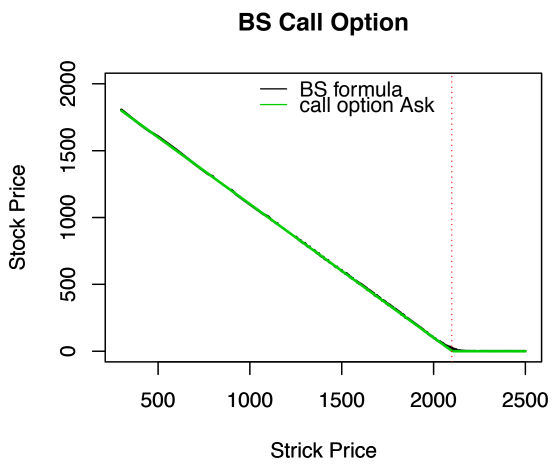
From the Figure 3-1, it is clear that the result for short expired date options is quite good. The theoratical curve and the real data point have been consistent. The red vertical line is just the at-the-money line. For the long expired date options, there are some differences near the at-the-money point. Because the price curve in Black-Scholes model would rise in excess of the at-the-money point. But the real value will react to the at-the-money pont faster and earlier. So for long expired date options, the real curvres for both put and call will a little bit different from theoratical curves near the at-the-money point.

3.2 Volatility smile

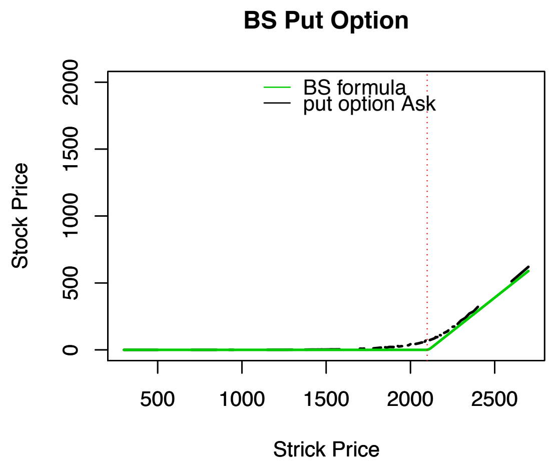
There is really a hard time to deal with the volatility. Function *GBSVolatility* is not easy to control. Implied volatility is calculated by *GBSVolatility*, which is plotted in Figure 3-2 and Figure 3-3. The result seems to be very very strange. I have checked the results from *GBSVolatility*, and it seems like that there's no reason why there are some break points and some very big fluctuations. The most weird plot is the implied volatility for call option in Figure 3-2. The minmum point has been shifted, different from the at-the-money point. This may affect



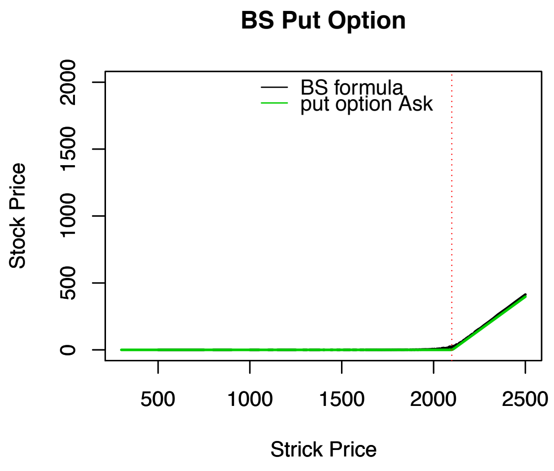
(a) The price of call option for long expired date.



(b) The price of call option for short expired date.



(c) The price of put option for long expired date.



(d) The price of put option for short expired date.

Figure 3-1: The results of Option Pricing through Black-Scholes Model.

taht the market expects larger at-the-money point, showing the S&P Index may continue to rise. For other three plots, we could see a smile face on the graph. I have tried to find the options with larger strick prices, to make the plots more symmetric. Maybe because of the unexpectedly stable S&P500 index, market used to expect a drop, so there is not any options with alrger strick prices.

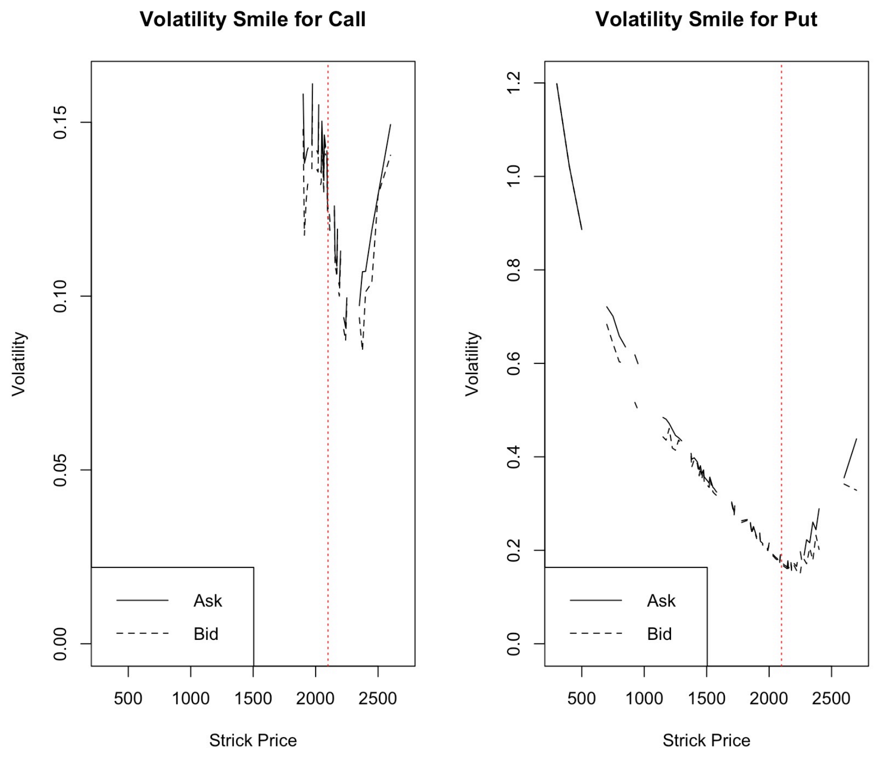


Figure 3-2: The implied volatility for long expired date.

Actually, the analysis methods in this article are really basic and simple. But we could still make a draft of what it should be. There are more advanced method, such as realized volatility to estimate the volatility. And also there are many interesting field in finance such as high frequency trading. The course have just given us the introduction of financial modelling, and led us to the gate of the world of financial engineering. Cheers!

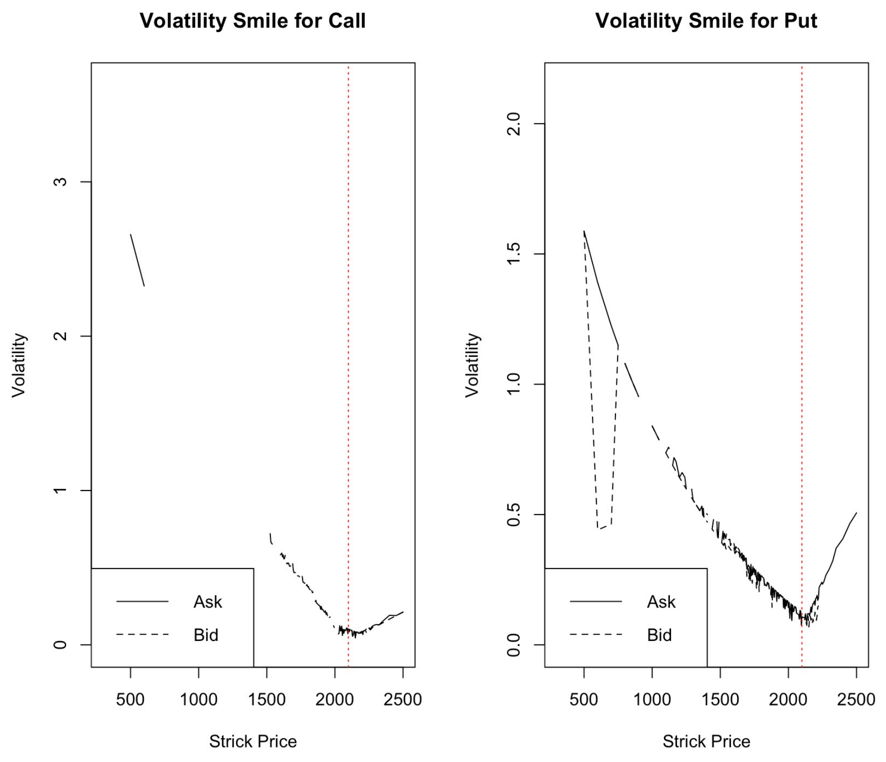


Figure 3-3: The implied volatility for short expired date.

Appendix A Code

A.1 *Garch* Model

```
library(xts)
startdate = c(2014,6,3)
da<-read.table("SP500.txt")
head(da)
tail(da)
dim(da)
da.xts<-as.xts(da)
spx.xts<-da.xts$Adj.Close
plot(spx.xts, main = "the S&P500 index", xlab = 'time')

rtd.SP<-diff(log(spx.xts))*100
rtd.SP<-rtd.SP[-1,]
rtd.SP<-rtd.SP['2014-06-03/']
plot(rtd.SP, main = "the log return of S\&P500 index", xlab = 'time
        ')
Box.test(rtd.SP, lag=12, type='Ljung-Box')
Box.test(rtd.SP^2, lag=12, type='Ljung-Box')
FinTS::ArchTest(x=rtd.SP, lags=12)

library(fGarch)
GARCH.model_1 <- garchFit(~garch(1,1), data=rtd.SP, trace=FALSE)
GARCH.model_2 <- garchFit(~garch(1,1), data=rtd.SP,  cond.dist='std
        ', trace=FALSE)
plot(GARCH.model_1)
```

```
vol_1 <- fBasics::volatility(GARCH.model_2)
res_1 <- residuals(GARCH.model_2, standardize=TRUE)
vol_1.ts <- ts(vol_1, frequency=252, start=startdate)
res_1.ts <- ts(res_1, frequency=252, start=startdate)
par(mfcol=c(2,1))
plot(vol_1.ts, xlab='Time', ylab='Volitility')
plot(res_1.ts, xlab='Time', ylab='Residuals')

require(rugarch)
spec=ugarchspec(variance.model = list(model = "eGARCH", garchOrder
        = c(1, 1)),
                mean.model = list(armaOrder = c(0, 0),include.mean
        = TRUE),
                             distribution.model = "norm",fixed.pars
        =list(mu=0))
myfit=ugarchfit(spec,data=rtd.SP,solver="solnp")

vol <- sigma(myfit)^2
vol.ts<-ts(vol, frequency = 252, start = startdate)
par(mfcol=c(1,1))
plot(vol.ts, xlab='Time', ylab='Volitility', main = "The volitility
         Distribution using Garch Model")
par(new=T)
plot(vol_1.ts, xlab='Time', ylab='Volitility', col=3, ylim = c(min(
        vol.ts),max(vol.ts)))
legend("topleft", c('eGarch','sGarch'), col = c(1,3), lty = c(1, 1)
        , bty = "n")

mean(vol.ts)
mean(vol_1.ts)
```

A.2 Option Pricing

```
fre = 252
expiredate = 90-3

SP.SP500<-spx.xts['2016-06-03']

OP.SP500<-read.csv("SP500OPL.csv")
OP.strick<-OP.SP500$Strike
OP.C.Ask<-OP.SP500$Ask
OP.C.Bid<-OP.SP500$Bid
OP.P.Ask<-OP.SP500$Ask.1
OP.P.Bid<-OP.SP500$Bid.1
n<-nrow(OP.SP500)

require(fOptions)

P<-as.numeric(SP.SP500)
T=expiredate/365
sigma=mean(vol_1.ts)/100
r=0.018

CP<-matrix(NA,2,n)
for (i in 1:n) {
    K<-OP.strick[i]
    CP[1,i]<-GBSOption(TypeFlag = "c", S = P, X = K,
                       Time = T, r = r,b = r, sigma = sigma)@price
    CP[2,i]<-GBSOption(TypeFlag = "p", S = P, X = K,
                       Time = T, r = r,b = r, sigma = sigma)@price
    }

par(mfcol=c(1,1))
plot(OP.strick,OP.C.Ask, xlab='Strick Price', ylab='Stock Price',
     main = "BS Call Option", type = "l", ylim = c(0,2000), lwd=2,
        col=1, pch=24)
par(new=T)
plot(OP.strick,CP[1,], col=3,xlab='Strick Price', ylab='Stock Price
        ',
     type = "l", ylim = c(0,2000), lwd=2, pch=24)
abline(v=P, col='red', lty=3)
legend("top", c('BS formula','call option Ask'), col = c(3,1), lty
        = c(1, 1), bty = "n")

par(mfcol=c(1,1))
plot(OP.strick,OP.P.Ask, xlab='Strick Price', ylab='Stock Price',
     main = "BS Put Option", type = "l", ylim = c(0,2000), lwd=2,
        col=1, pch=24)
par(new=T)
plot(OP.strick,CP[2,], col=3,xlab='Strick Price', ylab='Stock Price
        ',
     type = "l", ylim = c(0,2000), lwd=2, pch=24)
abline(v=P, col='red', lty=3)
legend("top", c('BS formula','put option Ask'), col = c(3,1), lty =
         c(1, 1), bty = "n")
```

A.3 Implied Volatility

```
C.vol.Ask <- numeric(n)
C.vol.Bid <- numeric(n)
P.vol.Ask <- numeric(n)
P.vol.Bid <- numeric(n)

for(i in 1:n){
    K<-OP.strick[i]
    if(!is.na(OP.C.Ask[i])){
        C.vol.Ask[i] <- GBSVolatility(price=OP.C.Ask[i], TypeFlag='
        c', S=P, X = K, Time=T, r=r, b=r)
        C.vol.Bid[i] <- GBSVolatility(price=OP.C.Bid[i], TypeFlag='
        c', S=P, X = K, Time=T, r=r, b=r)
        }
    if(!is.na(OP.P.Ask[i])){
        P.vol.Ask[i] <- GBSVolatility(price=OP.P.Ask[i], TypeFlag='
        p', S=P, X = K, Time=T, r=r, b=r)
        P.vol.Bid[i] <- GBSVolatility(price=OP.P.Bid[i], TypeFlag='
        p', S=P, X = K, Time=T, r=r, b=r)
        }
}
CC<-matrix(NA,n,8)
for (jj in 1:n) {
    if(C.vol.Ask[jj]>=0.01){
        CC[jj,1]<-C.vol.Ask[jj]
        CC[jj,2]<-OP.strick[jj]
        }
    if(C.vol.Bid[jj]>=0.01){
        CC[jj,3]<-C.vol.Bid[jj]
        CC[jj,4]<-OP.strick[jj]
        }
    if(P.vol.Ask[jj]>=0.01){
        CC[jj,5]<-P.vol.Ask[jj]
        CC[jj,6]<-OP.strick[jj]
        }
    if(P.vol.Bid[jj]>=0.01){
        CC[jj,7]<-P.vol.Bid[jj]
        CC[jj,8]<-OP.strick[jj]
        }
}

par(mfrow=c(1,2))
```

```
plot(c(OP.strick, OP.strick), c(C.vol.Bid, C.vol.Ask), typ='n',
       main='Volatility Smile for Call', xlab='Strick Price', ylab
       ='Volatility')
lines(CC[,2],CC[,1], lty=1)
lines(CC[,4],CC[,3], lty=2)
abline(v=P, col='red', lty=3)
legend('bottomleft', c('Ask', 'Bid'), lty=c(1,2))
plot(c(OP.strick, OP.strick), c(P.vol.Bid, P.vol.Ask), typ='n',
       main='Volatility Smile for Put', xlab='Strick Price', ylab
       ='Volatility')
lines(CC[,6],CC[,5], lty=1)
lines(CC[,8],CC[,7], lty=2)
abline(v=P, col='red', lty=3)
legend('bottomleft', c('Ask', 'Bid'), lty=c(1,2))

par(mfrow=c(1,2))
plot(c(OP.strick, OP.strick), c(C.vol.Bid, C.vol.Ask), typ='n',
       main='Volatility Smile for Call', xlab='Strick Price', ylab
       ='Volatility')
lines(OP.strick, C.vol.Bid, lty=1)
lines(OP.strick, C.vol.Ask, lty=2)
abline(v=P, col='red', lty=3)
legend('bottomleft', c('Ask', 'Bid'), lty=c(1,2))
plot(c(OP.strick, OP.strick), c(P.vol.Bid, P.vol.Ask), typ='n',
       main='Volatility Smile for Put', xlab='Strick Price', ylab
       ='Volatility')
lines(OP.strick, P.vol.Bid, lty=1)
lines(OP.strick, P.vol.Ask, lty=2)
abline(v=P, col='red', lty=3)
legend('bottomleft', c('Ask', 'Bid'), lty=c(1,2))
```