Hw unit 1

Question 1

Year	RAM	increase rate
1988	0.002	
1991	0.004	1
1996	0.5	124
2000	1	1
2003	2	1
2007	8	3
2014	16	1

#Question 2

LOGISTIC REGRESSION

- With logistic regression, the response variable is an indicator of some characteristic, that is, a 0/1 variable.
- Logistic regression is used to determine whether other measurements are related to the presence of some characteristic. For example, whether certain blood measures are predictive of having a disease.
- Logistic regression equation does not directly predict the probability that the indicator is equal to 1. It predicts the log odds that an observation will have an indicator equal to 1.

The model logistic regression model is that

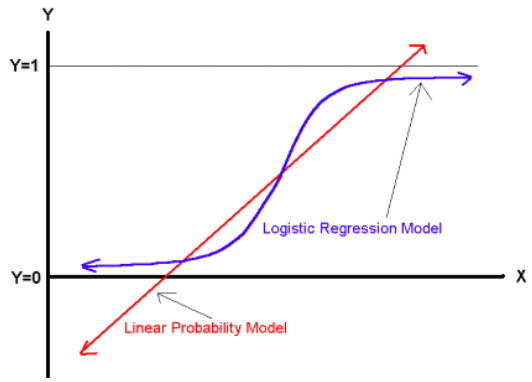
$$\mathrm{P} = \frac{\exp(\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p)}{1 + \exp(\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p)}$$

Because logistic regression predicts probabilities, rather than just classes, we can fit it using likelihood.

$$L(\beta_0, \beta) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i)^{1-y_i}$$

The null hypothesis underlying the overall model states that all βs equal zero.

Comparing the LP and Logit Models



We can use the *glm* (generalized linear model) function in R to estimate a logistic regression model using.

Hw unit 2

#Question 1

```
setwd("C:/Users/CH/Desktop/big data")
read.csv("hw unit2.csv")
plot(year,RAM,type ="o",col="red",main = "RAM of computer")
```

#Question 2

```
f<-read.csv("hw unit2.csv")
year<-f$Year;RAM<-f$RAM
plot(year,RAM,type ="o",col="black",main = "RAM of computer")
# load necessary packages
require(datasets)
require(class)
require(grDevices)
require(lattice)
# define log-returns for the DAX and FTS
x= year
y = RAM
# estimated log-returns for the DAX index for different bandwidths
splines.reg.l1 = smooth.spline(x,y, spar = 0.2)   # lambda = 0.2
splines.reg.l2 = smooth.spline(x,y, spar = 1)   # lambda = 1
splines.reg.l3 = smooth.spline(x,y, spar = 2)   # lambda = 2
# plot for the regression results
lines(splines.reg.l1, col = "red", lwd = 2)   # regression line with lambda = 0.2
lines(splines.reg.l2, col = "green", lwd = 2)   # regression line with lambda = 1
lines(splines.reg.l3, col = "blue", lwd = 2)   # regression line with lambda = 2
```

#Question 3

```
x = 6
n = 1000
lambda = 2
p = lambda / n
dbinom (x,2*n,p) # binomial probability mass function
dpois (x, 2*lambda ) # Poisson probability mass function
dpois (0, 5 )
```

Hw unit 3

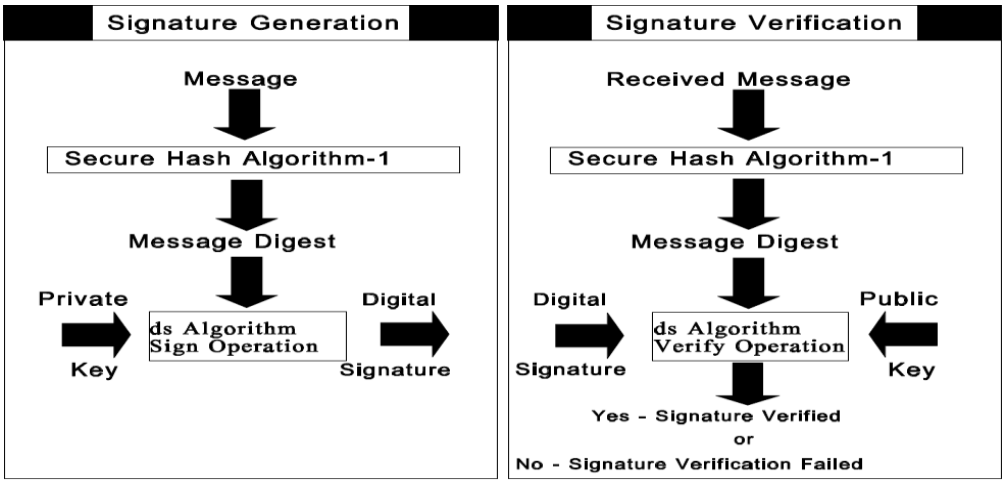
#Question 1

```
install.packages("digest")
library("digest")
digest("I learn a lot from this class when I am proper listening to the professor", "sha256")
digest("I do not learn a lot from this class when I am absent and playing on my Iphone", "sha256")
```

Question 2

Introduce DSA(Digital Signature Algorithm).

- **What Is DSA (Digital Signature Algorithm)?**
  - Digital signatures are essential to **verify the sender of a document's identity.** The signature is computer using a set of rules and algorithm such that the identity of the person can be verified.
  - The signature is generated by the use of **a private key** that known only to **the user.** The signature is verified when a public key is corresponds to the private key. With every user having a public/private key pair, this is an example of public-key cryptography.
  - **Public keys,** which are known by everyone, can be used to verify the signature of a user**. The private key**, which is never shared, is used in signature generation, which can only be done by the user.
- **What can DSA do?**
  - Digital signatures are used to detect unauthorized modifications to data. Also, the recipient of a digitally signed document in proving to a third party that the document was indeed signed by the person who it is claimed to be signed by. This is known as nonrepudiation, because the person who signed the document cannot repudiate the signature at a later time.
  - Digital signature algorithms can be used in e-mails, electronic funds transfer, electronic data interchange, software distribution, data storage, and just about any application that would need to assure the integrity and originality of data.



Question 3

- **Code**

```
install.packages(RJSONIO)
library(RJSONIO)
letter<-LETTERS[1:10]
country<-c("China","the US","the UK","Russia",
                    "Korea","Japan","Italy","Brazil","India","Germany")
data<-data.frame(letter,country)
da<-as.matrix(data)
cat(toJSON(da))
```

- **The outcome**


```
[ {
  "letter": "A",
"country": "China"
},
{
  "letter": "B",
"country": "the US"
},
{
  "letter": "C",
"country": "the UK"
},
{
  "letter": "D",
"country": "Russia"
},
{
  "letter": "E",
"country": "Korea"
},
{
  "letter": "F",
"country": "Japan"
},
{
  "letter": "G",
"country": "Italy"
},
{
  "letter": "H",
"country": "Brazil"
},
{
  "letter": "I",
"country": "India"
},
{
  "letter": "J",
"country": "Germany"
} ]
```


Question 4

```
rm(list = ls(all = TRUE))
graphics.off()
# install and load packages #
libraries = c("zoo", "tseries")
lapply(libraries, function(x) if (!(x %in% installed.packages())) {install.packages(x)})
lapply(libraries, library, quietly = TRUE, character.only = TRUE)
# load dataset #
load(file = "C:/Users/CH/Desktop/big data/crix.RData")
```

```
ret = diff(log(crix))
# d order #
Box.test(ret, type = "Ljung-Box", lag = 20)
# stationary test #
adf.test(ret, alternative = "stationary")
kpss.test(ret, null = "Trend")
par(mfrow = c(1, 2))
# acf plot #
autocorr = acf(ret, lag.max = 20, ylab = "Sample Autocorrelation", main = NA, lwd = 2, ylim
= c(-0.3, 1))
# LB test of linear dependence
print(cbind(autocorr$lag, autocorr$acf))
Box.test(ret, type = "Ljung-Box", lag = 1, fitdf = 0)
Box.test(autocorr$acf, type = "Ljung-Box")
 # plot of pacf #
autopcorr = pacf(ret, lag.max = 20, ylab = "Sample Partial Autocorrelation", main = NA, ylim
= c(-0.3, 0.3), lwd = 2)
print(cbind(autopcorr$lag, autopcorr$acf))
# arima model
par(mfrow = c(1, 1))
auto.arima(ret)
fit1 = arima(ret, order = c(1, 0, 1))
tsdiag(fit1)
Box.test(fit1$residuals, lag = 1)
# aic
aic = matrix(NA, 6, 6)
for (p in 0:4) {
   for (q in 0:3) {
      a.p.q = arima(ret, order = c(p, 0, q))
      aic.p.q = a.p.q$aic
      aic[p + 1, q + 1] = aic.p.q
   }
}
aic
# bic
bic = matrix(NA, 6, 6)
for (p in 0:4) {
   for (q in 0:3) {
      b.p.q = arima(ret, order = c(p, 0, q))
      bic.p.q = AIC(b.p.q, k = log(length(ret)))
      bic[p + 1, q + 1] = bic.p.q
   }
}
bic
```

```
# select p and q order of ARIMA model
fit4 = arima(ret, order = c(2, 0, 3))
tsdiag(fit4)
Box.test(fit4$residuals, lag = 1)
fitr4 = arima(ret, order = c(2, 1, 3))
tsdiag(fitr4)
Box.test(fitr4$residuals, lag = 1)
# to conclude, 202 is better than 213
fit202 = arima(ret, order = c(2, 0, 2))
tsdiag(fit202)
tsdiag(fit4)
tsdiag(fitr4)
AIC(fit202, k = log(length(ret)))
AIC(fit4, k = log(length(ret)))
AIC(fitr4, k = log(length(ret)))
fit202$aic
fit4$aic
fitr4$aic
# arima202 predict
fit202 = arima(ret, order = c(2, 0, 2))
crpre = predict(fit202, n.ahead = 30)
dates = seq(as.Date("02/08/2014", format = "%d/%m/%Y"), by = "days", length = length(ret))
plot(ret, type = "l", xlim = c(0, 644), ylab = "log return", xlab = "days",
        lwd = 1.5)
lines(crpre$pred, col = "red", lwd = 3)
lines(crpre$pred + 2 * crpre$se, col = "red", lty = 3, lwd = 3)
lines(crpre$pred - 2 * crpre$se, col = "red", lty = 3, lwd = 3)
# Produces GARCH estimation results using ARIMA model residuals
rm(list = ls(all = TRUE))
graphics.off()
# install and load packages
libraries = c("FinTS", "tseries", "forecast", "fGarch")
lapply(libraries, function(x) if (!(x %in% installed.packages())) {
   install.packages(x)
})
lapply(libraries, library, quietly = TRUE, character.only = TRUE)
# load dataset
load(file = "C:/Users/CH/Desktop/big data/crix.RData")
ret = diff(log(crix1))
# vol cluster
fit202 = arima(ret, order = c(2, 0, 2))
par(mfrow = c(1, 1))
res = fit202$residuals
res2 = fit202$residuals^2
```

```
# different garch model
fg11 = garchFit(data = res, data ~ garch(1, 1))
summary(fg11)
fg12 = garchFit(data = res, data ~ garch(1, 2))
summary(fg12)
fg21 = garchFit(data = res, data ~ garch(2, 1))
summary(fg21)
fg22 = garchFit(data = res, data ~ garch(2, 2))
summary(fg22)
# residual plot
reszo = zoo(fg11@residuals, order.by = index(crix1))
plot(reszo, ylab = NA, lwd = 2)
par(mfrow = c(1, 2))
fg11res2 = fg11@residuals
acfres2  = acf(fg11res2, lag.max = 20, ylab = "Sample Autocorrelation",
                 main = NA, lwd = 2)
pacfres2 = pacf(fg11res2, lag.max = 20, ylab = "Sample Partial Autocorrelation",
                   main = NA, lwd = 2, ylim = c(-0.5, 0.5))
fg12res2 = fg12@residuals
acfres2  = acf(fg12res2, lag.max = 20, ylab = "Sample Autocorrelation",
                 main = NA, lwd = 2)
pacfres2 = pacf(fg12res2, lag.max = 20, ylab = "Sample Partial Autocorrelation",
                   main = NA, lwd = 2, ylim = c(-0.5, 0.5))
# qq plot
par(mfrow = c(1, 1))
plot(fg11, which = 13)  #9,10,11,13
# kp test
set.seed(100)
x = rnorm(200)
# Do x and y come from the same distribution?
ks.test(x, fg11@residuals)
```

Hw unit 4

Question 1

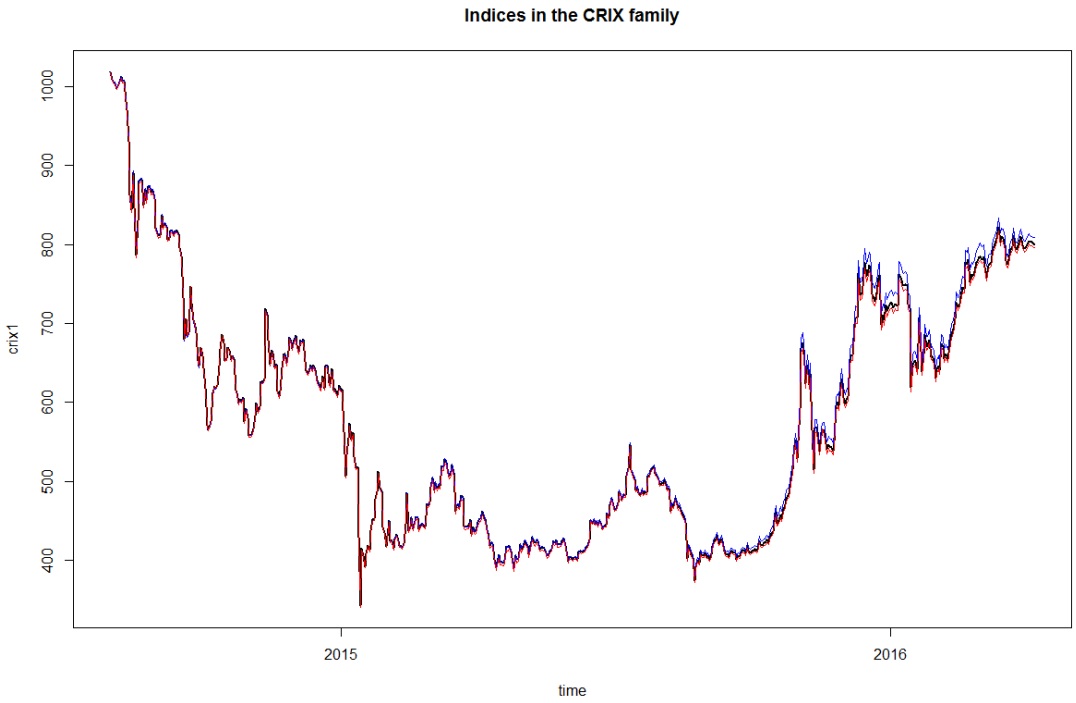


Figure 3: The daily value of indices in the CRIX family from 01/08/2014 to 06/04/2016: CRIX, ECRIX and EFCRIX.

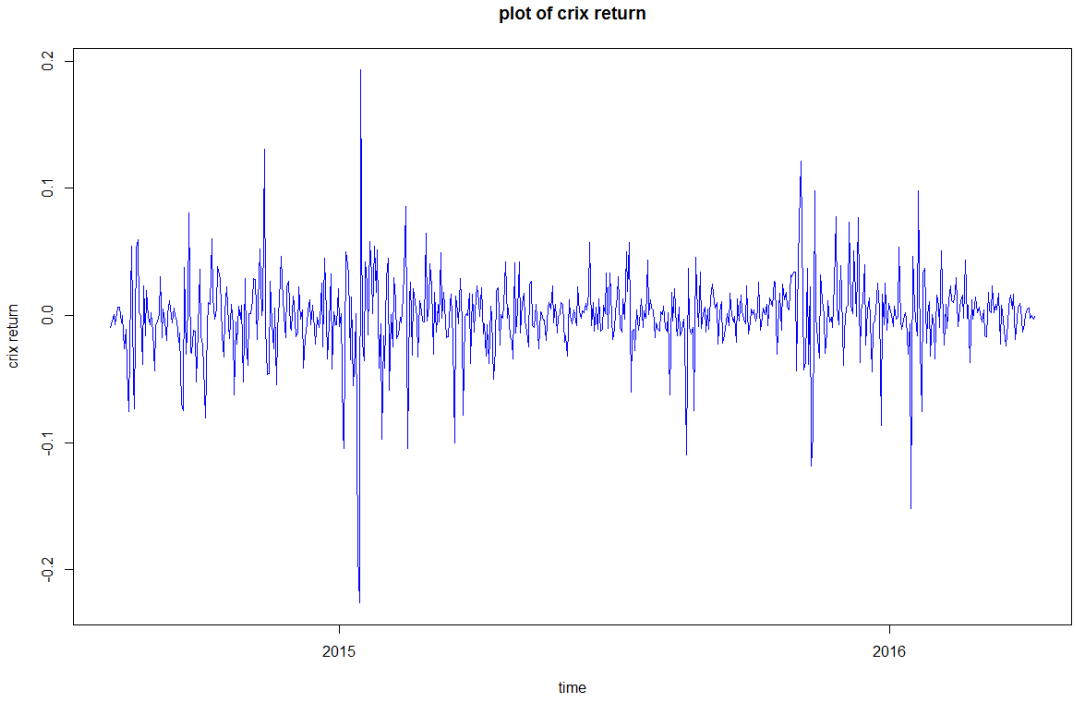


Figure 4: The log returns of CRIX index from 01/08/2014 to 06/04/2016.

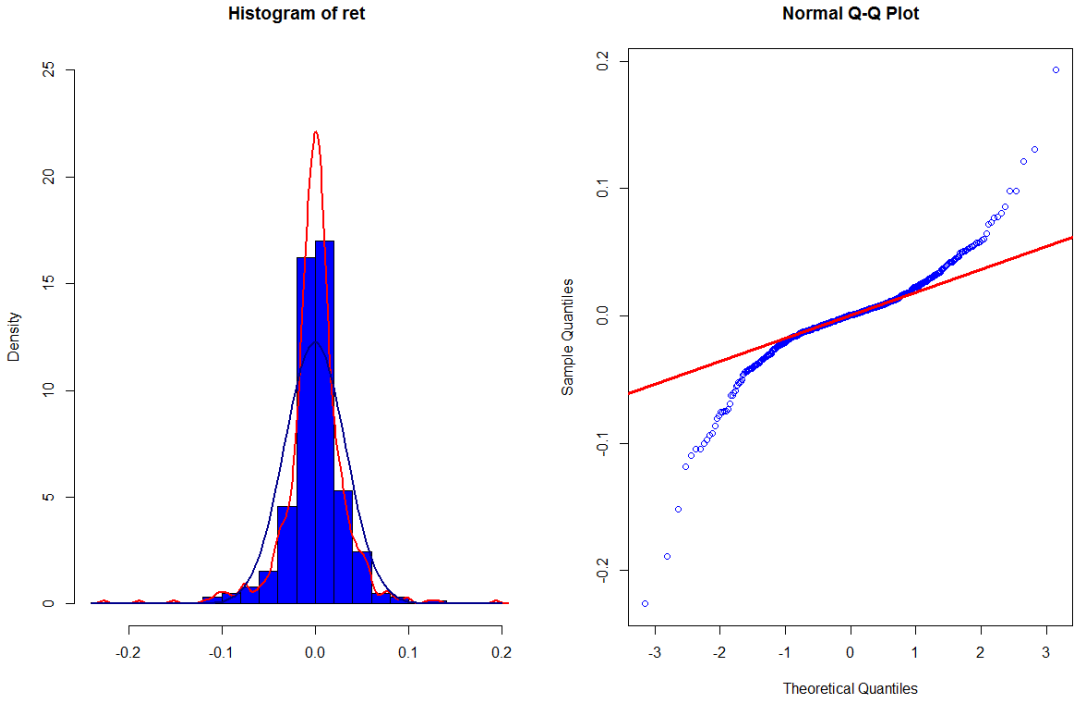


Figure 5: Histogram and QQ plot of CRIX returns from 01/08/2014 to 06/04/2016.

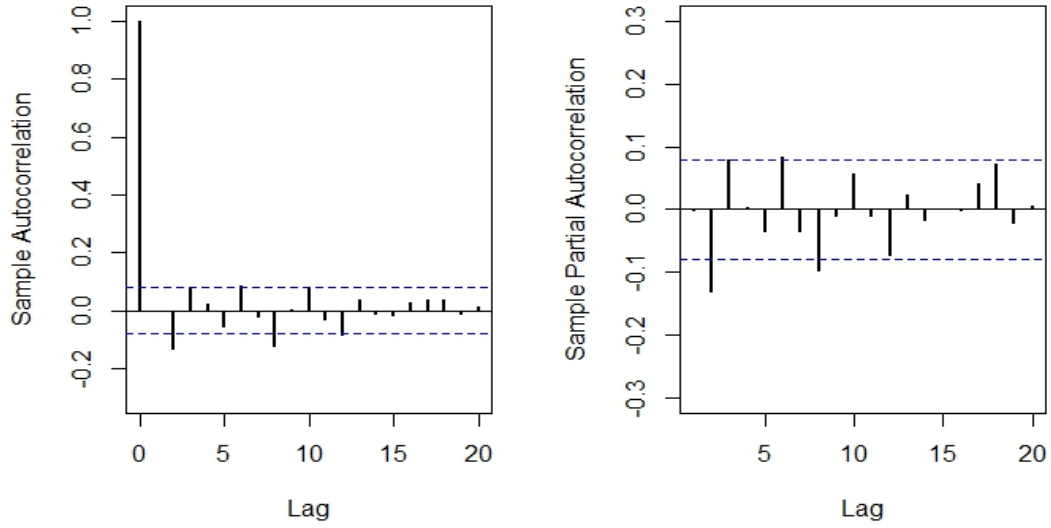
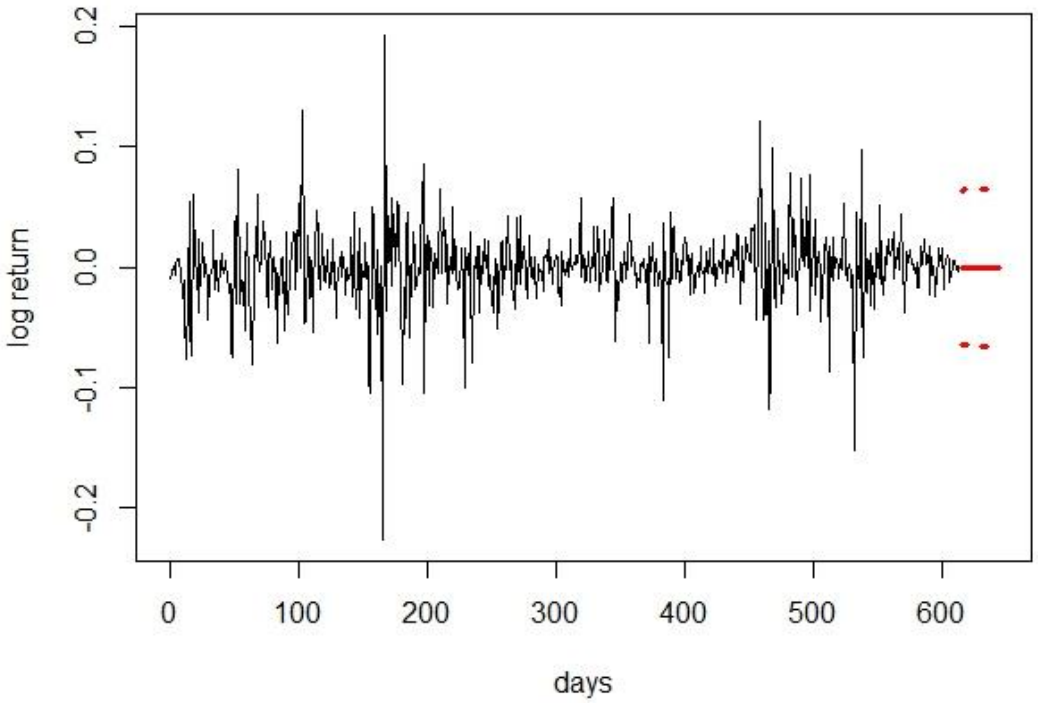


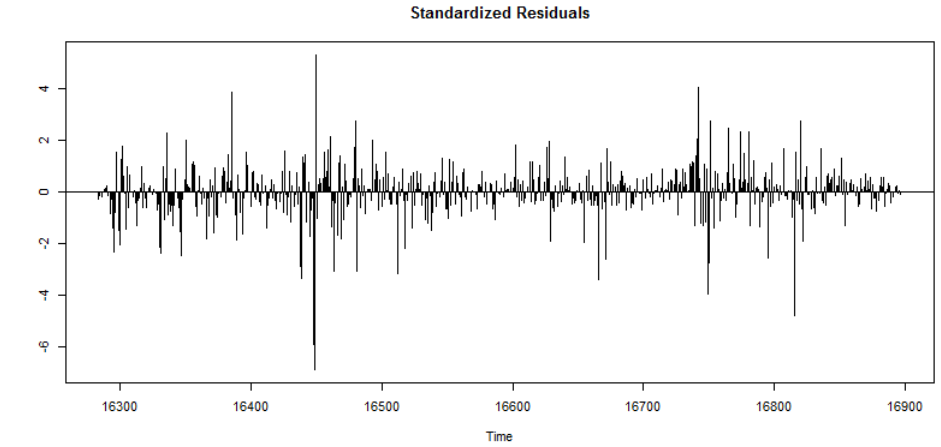
Figure 6: The sample ACF and PACF of CRIX returns from 01/08/2014 to 06/04/2016.

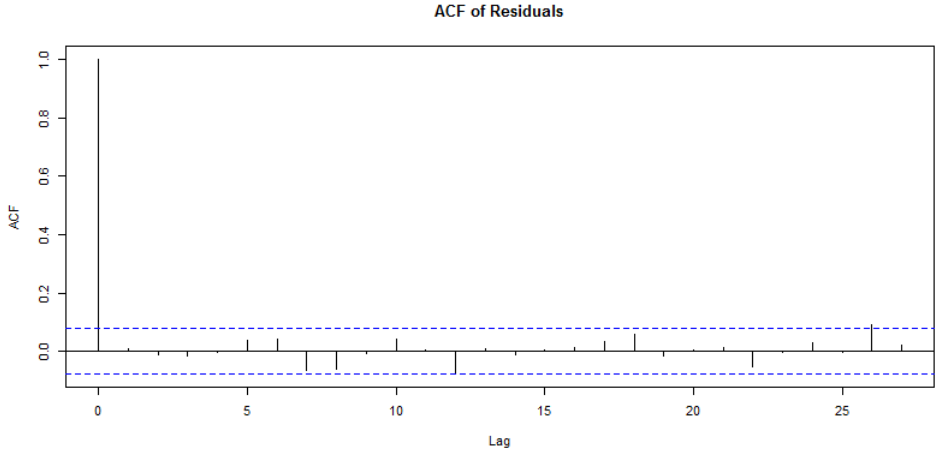
Question 2

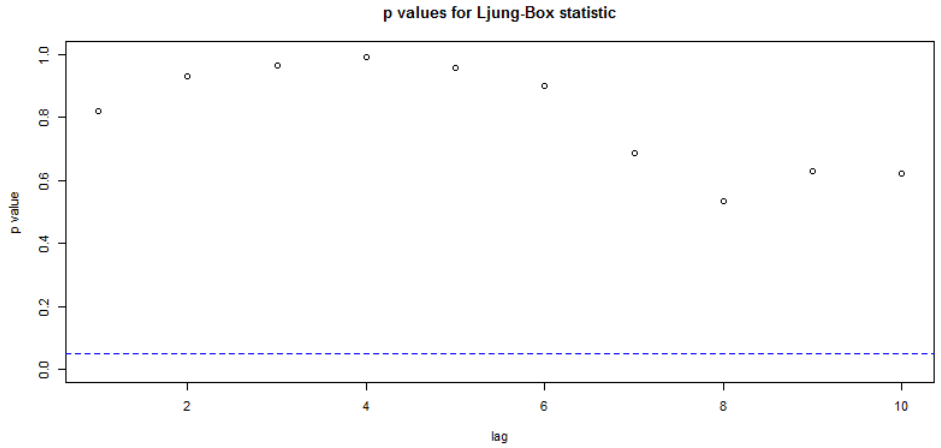


The CRIX returns and predicted values.
The confidence bands are red dashed lines.

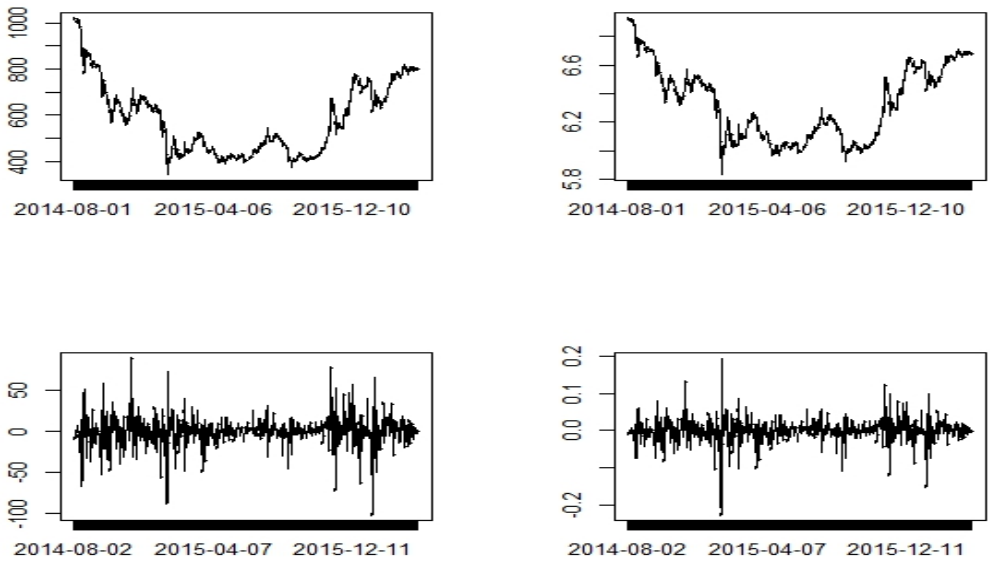
Question 3



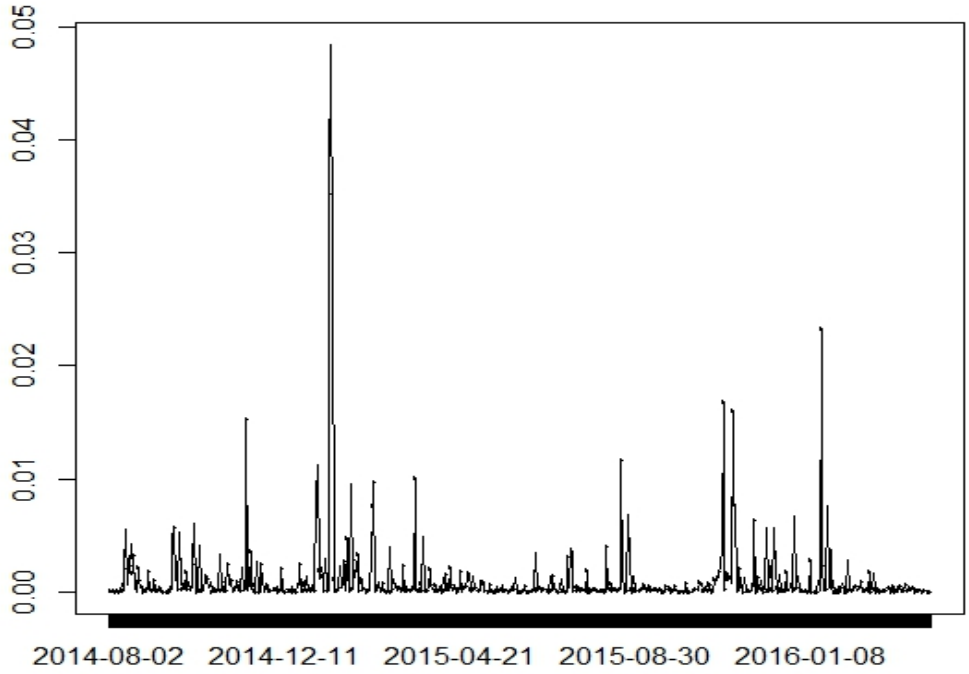




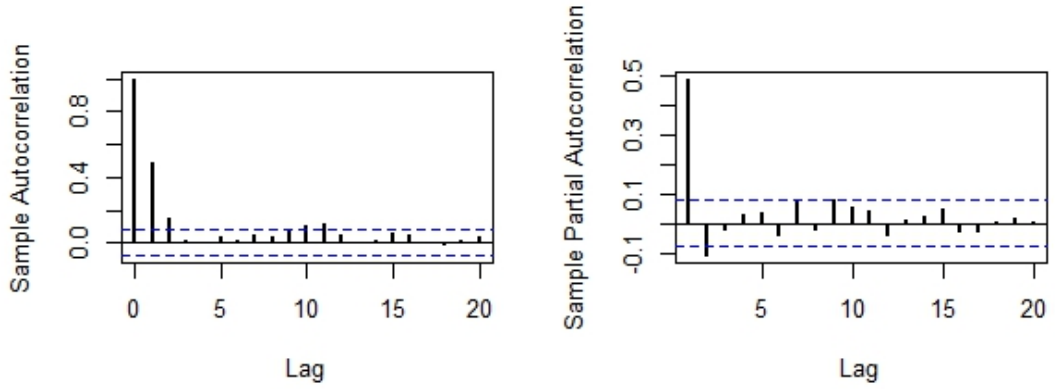
- Diagnostic plot of ARIMA(2,0,2) model
- significant p-values of Ljung-Box test statistic
- model residuals are independent



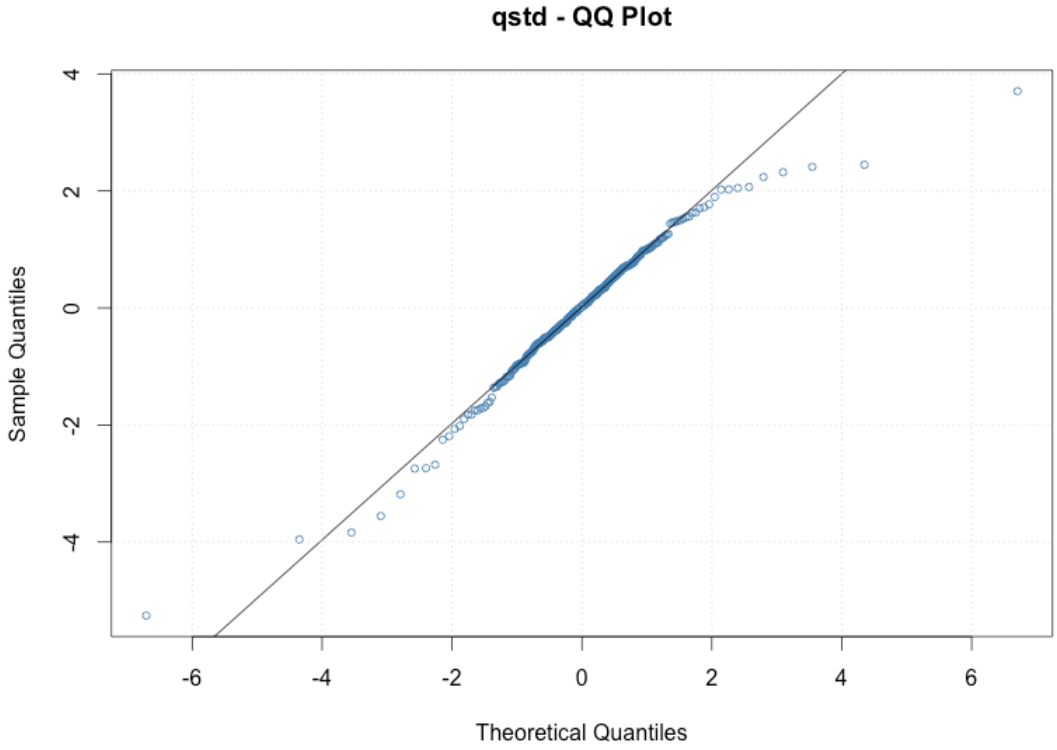
the difference of crix returns

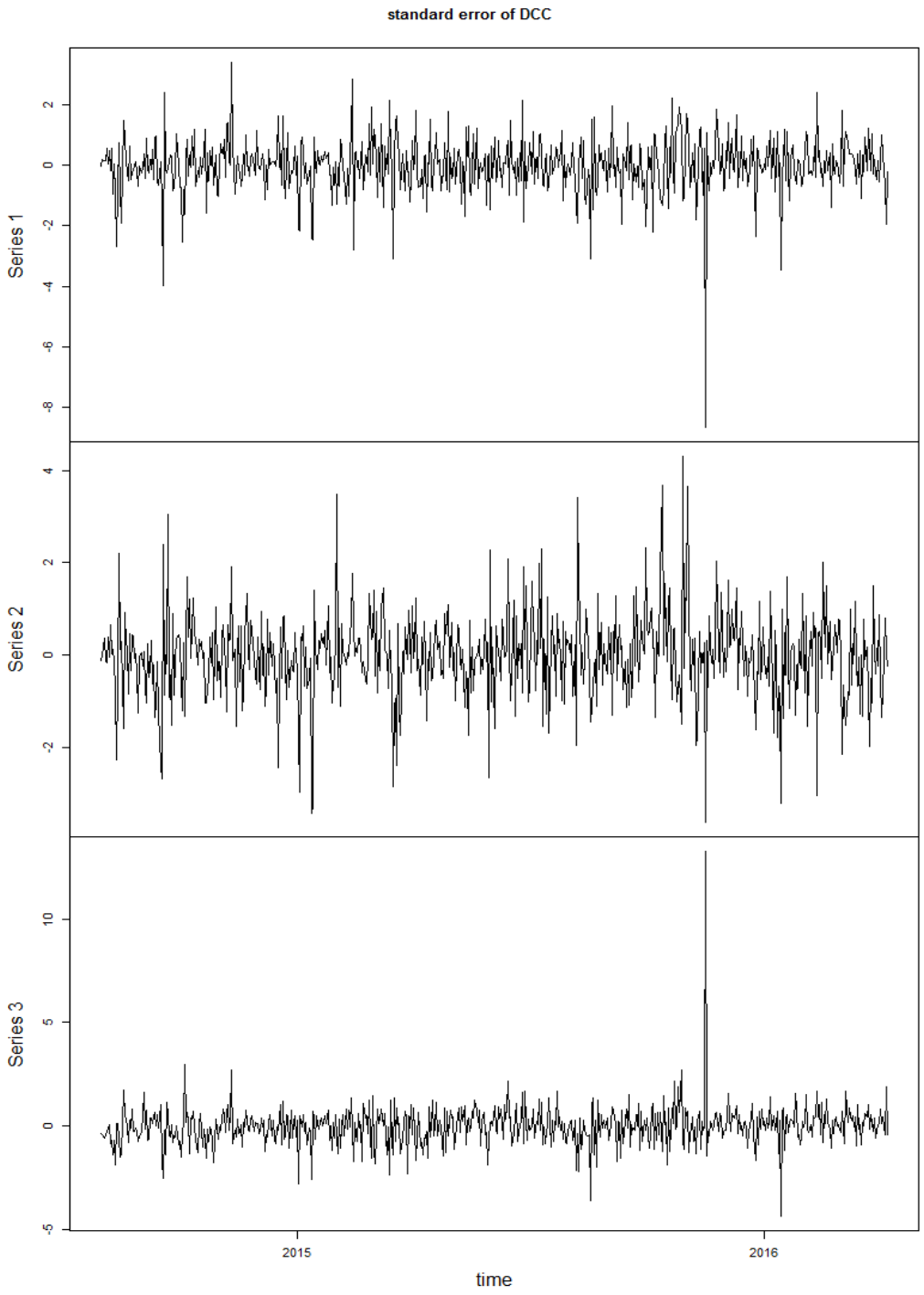


the squared ARIMA(2,0,2) residuals of CRIX returns.

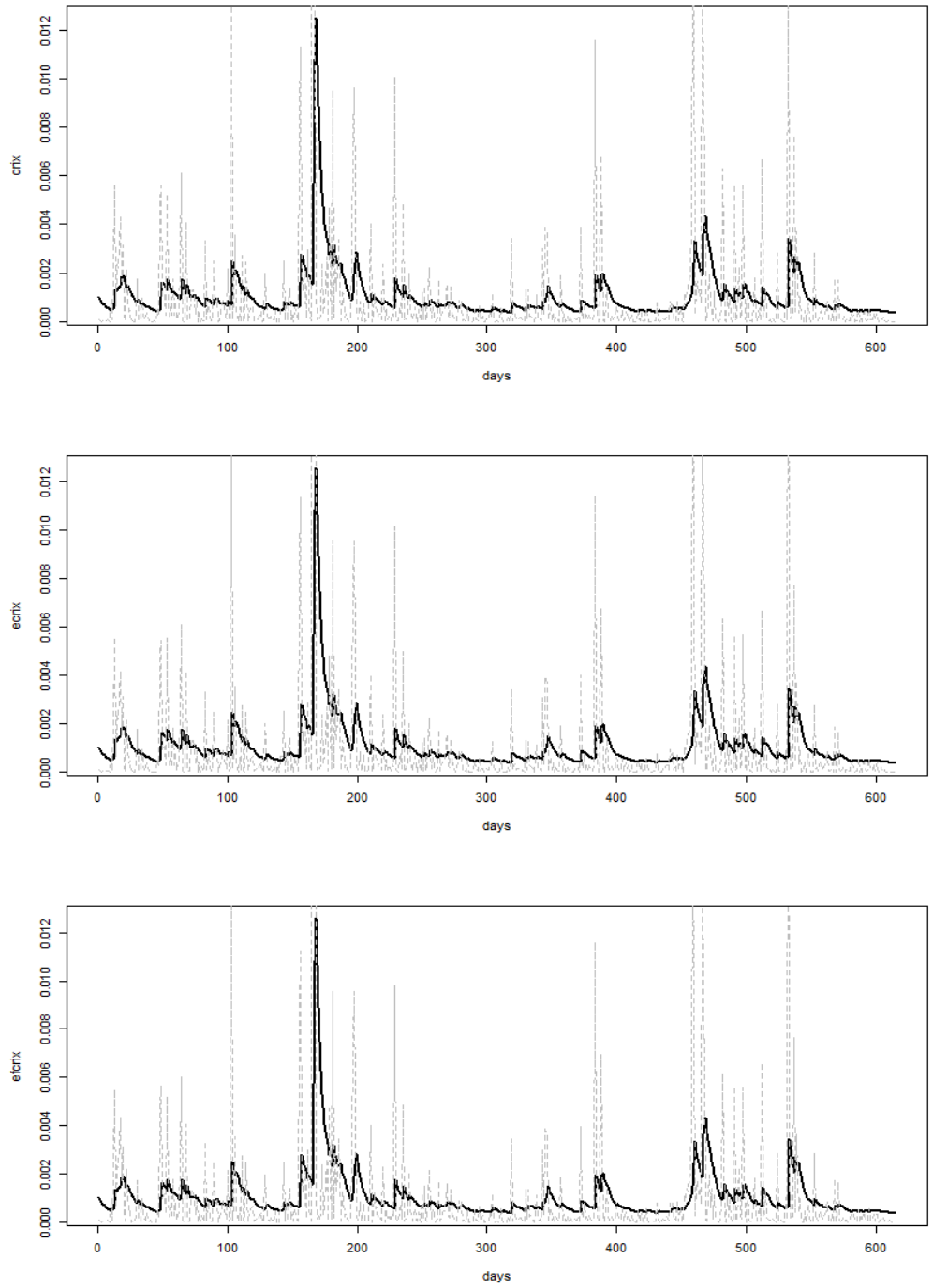


the ACF and PACF of squared ARIMA(2,0,2) residuals from 01/08/2014 to 06/04/2016

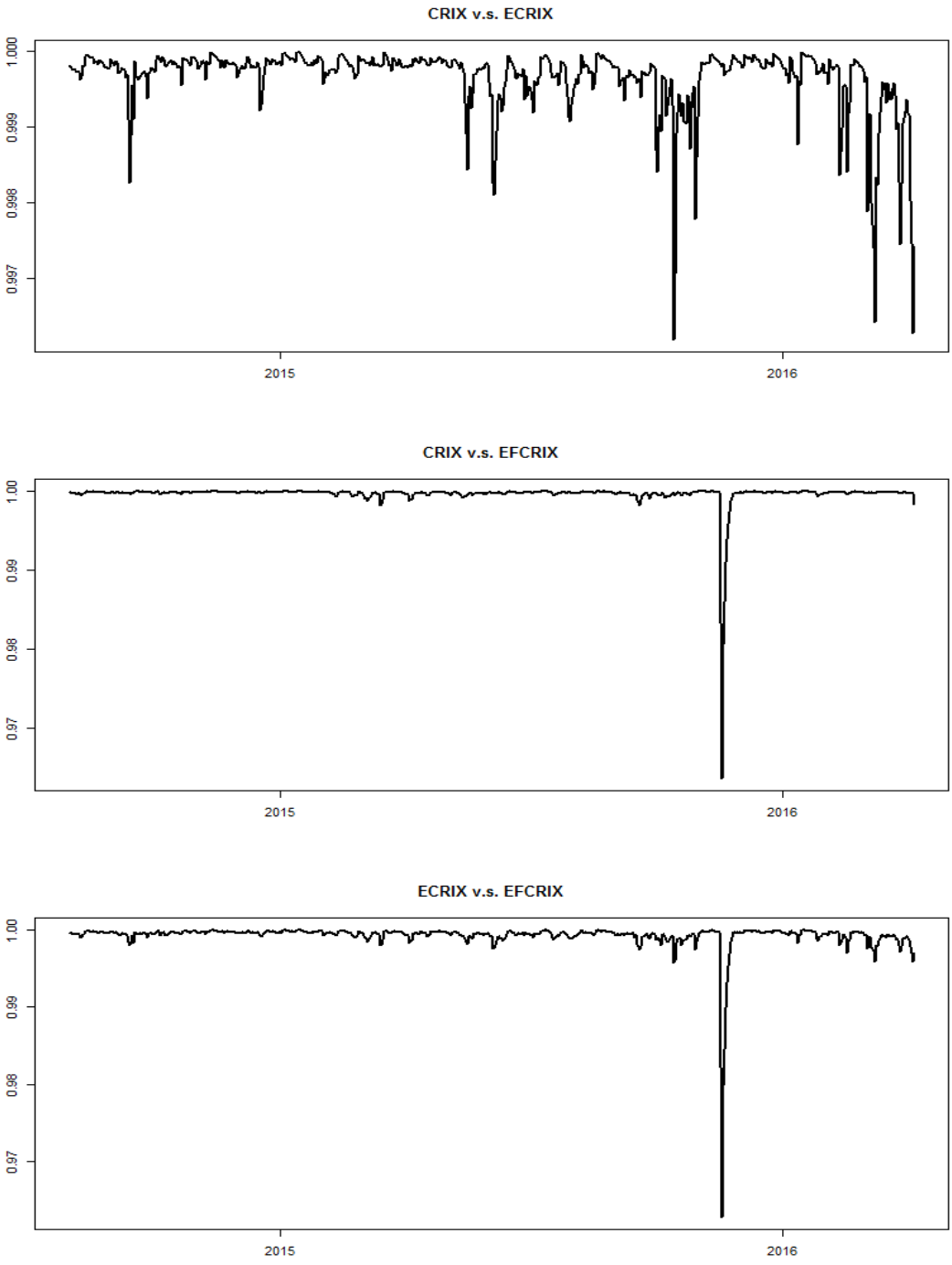




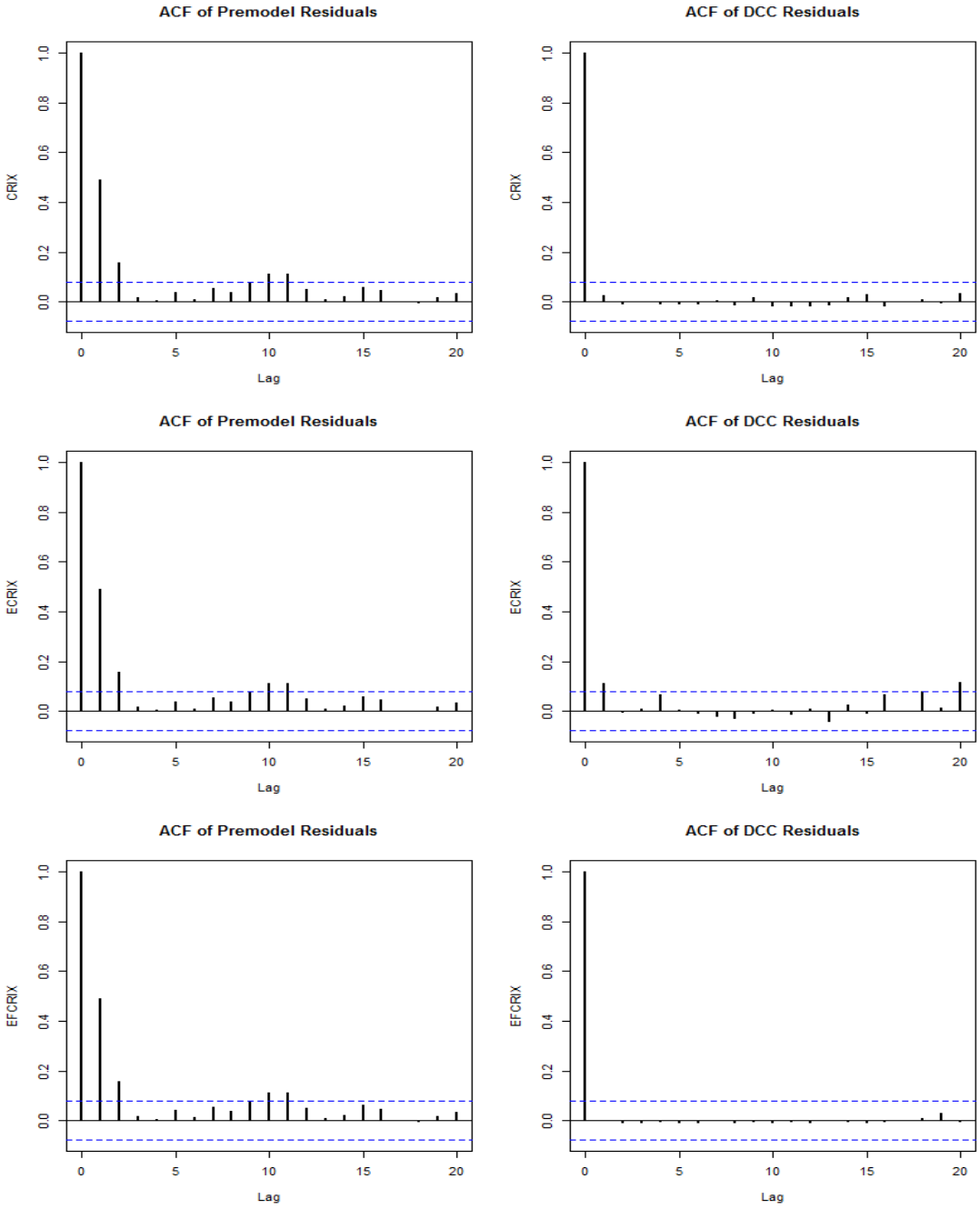
The standard error of DCC-GARCH model.
CRIX(upper), ECRIX (middle), EFCRIX(lower)



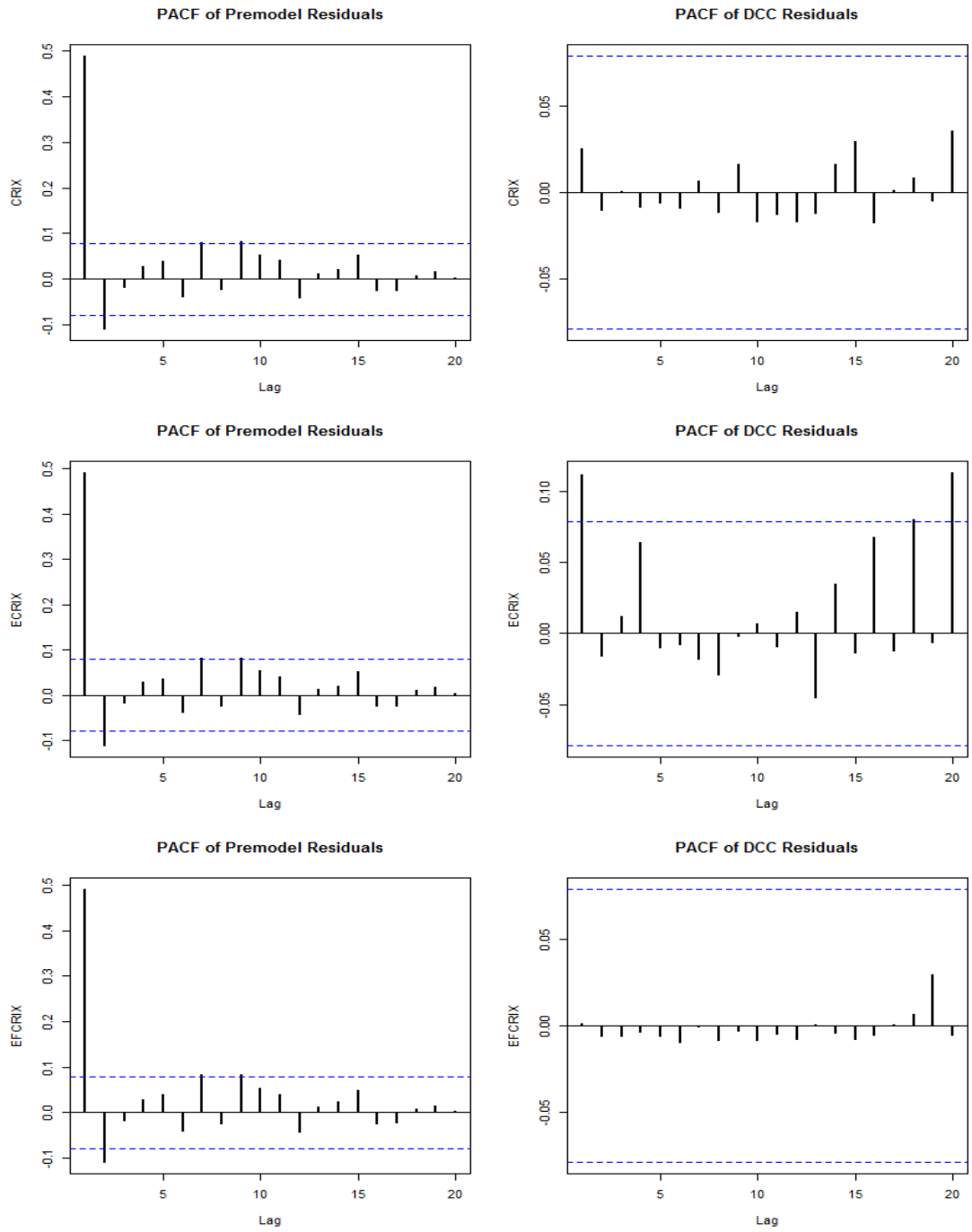
The estimated volatility (black) and realized volatility (grey) using DCC-GARCH model.
CRIX(upper), ECRIX (middle), EFCRIX(lower)



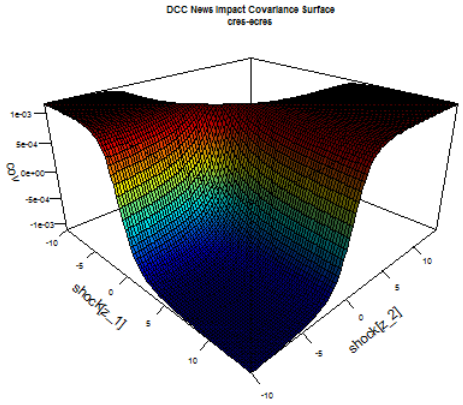
The dynamic autocorrelation between three CRIX indices:
CRIX,ECRIX and EFCRIX estimated by DCC-GARCH model.

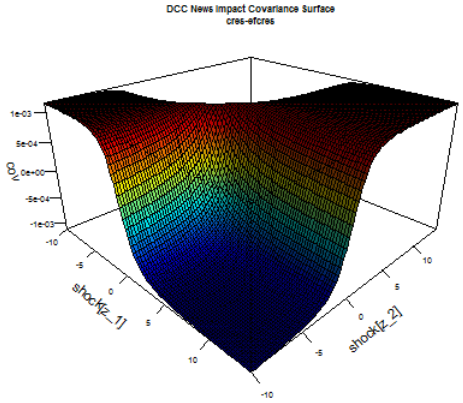


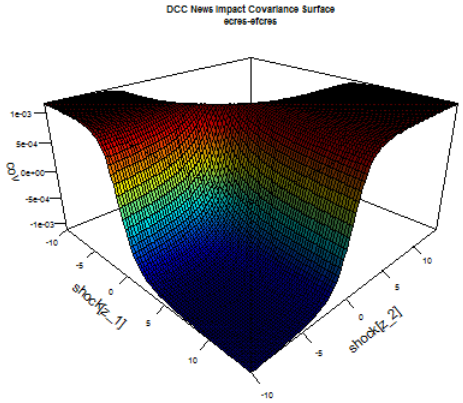
The comparison of ACF between premodel squared residuals and DCC squared residuals. CRIX(upper), ECRIX (middle), EFCRIX(lower)



The comparison of PACF between premodel squared residuals and DCC squared residuals.
CRIX(upper), ECRIX (middle), EFCRIX(lower)


DCC News Impact Covariance Surface
cres-ecres
1e-03
5e-04
0e+00
-5e-04
-1e-03
cov
-10
-5
0
5
10
shock[z_1]
-10
-5
0
5
10
shock[z_2]


DCC News Impact Covariance Surface
cres-efcres
1e-03
5e-04
0e+00
-5e-04
-1e-03
cov
-10
-5
0
5
10
shock[z_1]
-10
-5
0
5
10
shock[z_2]


DCC News Impact Covariance Surface
ecres-efcres
1e-03
5e-04
0e+00
-5e-04
-1e-03
cov
-10
-5
0
5
10
shock[z_1]
-10
-5
0
5
10
shock[z_2]