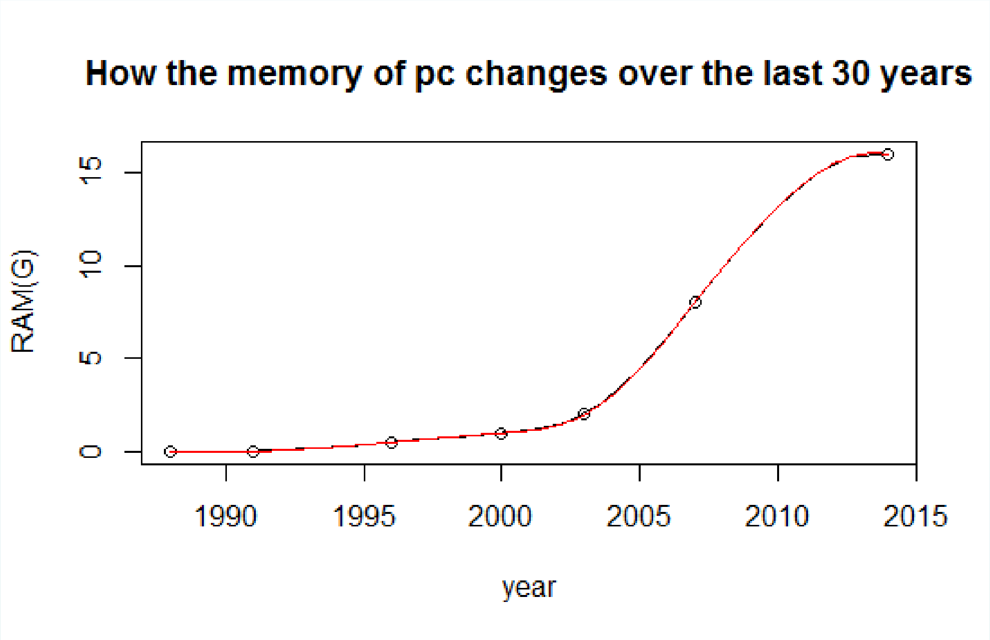
Homework1
(Question1)

This is the development path of memory of PCs



Corresponding code

```
year<c(1988,1991,1996,2000,2003,2007
,2014)
RAM<-c(0.002,0.004,0.5,1,2,8,16)
plot(year,RAM,ylab = "")
title(main ="How the memory of pc
changes over the last 30 years",ylab =
"RAM(G)")
lines(spline(year,RAM))
lines(spline(year,RAM, n = 201), col = 2)
```

Twenty years ago,the memory of pc is too small to support the analysis of FMRY

Homework1

(Question2)

Logistic regression:

In statistics, logistic regression, or logit regression, or logit model[1] is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression.[2] In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

Homework1
(Question2)

A example:Probability of passing an exam versus hours of study

Suppose we wish to answer the following question:

A group of 20 students spend between 0 and 6 hours studying for an exam. How does the number of hours spent studying affect the probability that the student will pass the exam?

The reason for using logistic regression for this problem is that the dependent variable pass/fail represented by "1" and "0" are not cardinal numbers. If the problem was changed so that pass/fail was replaced with the grade 0–100 (cardinal numbers), then simple regression analysis could be used.

The table shows the number of hours each student spent studying, and whether they passed (1) or failed (0)

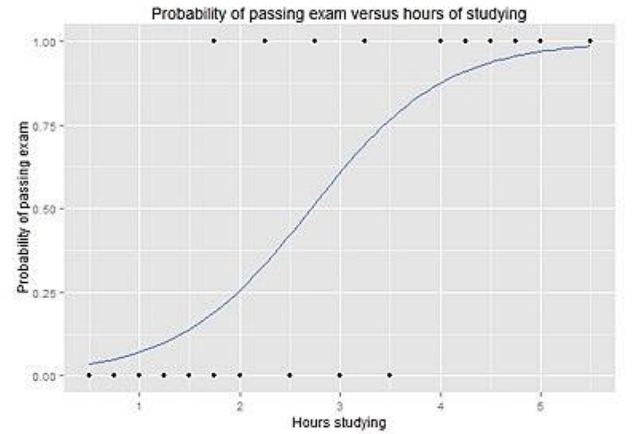
Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

Homework1
(Question2)

The graph shows the probability of passing the exam versus the number of hours studying, with the logistic regression curve fitted to the data. The logistic regression analysis gives the following output.

	Coefficient	Std.Error	z-value	P-value (Wald)
Intercept	−4.0777	1.7610	−2.316	0.0206
Hours	1.5046	0.6287	2.393	0.0167

$$\text{Probability of passing exam} = \frac{1}{1 + \exp\left(-\left(1.5046 \cdot \text{Hours} - 4.0777\right)\right)}$$



Graph of a logistic regression curve showing probability of passing an exam versus hours studying

Homework2

Code:

```
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 )
```

Homework3
(Question1)

```
library("digest")
# now do the hash code calculation
digest("I learn a lot from this class when I am proper listening to the professor")
digest("I do not learn a lot from this class when I am absent and playing on my Iphone")
```

Homewor3
(Question2)

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.

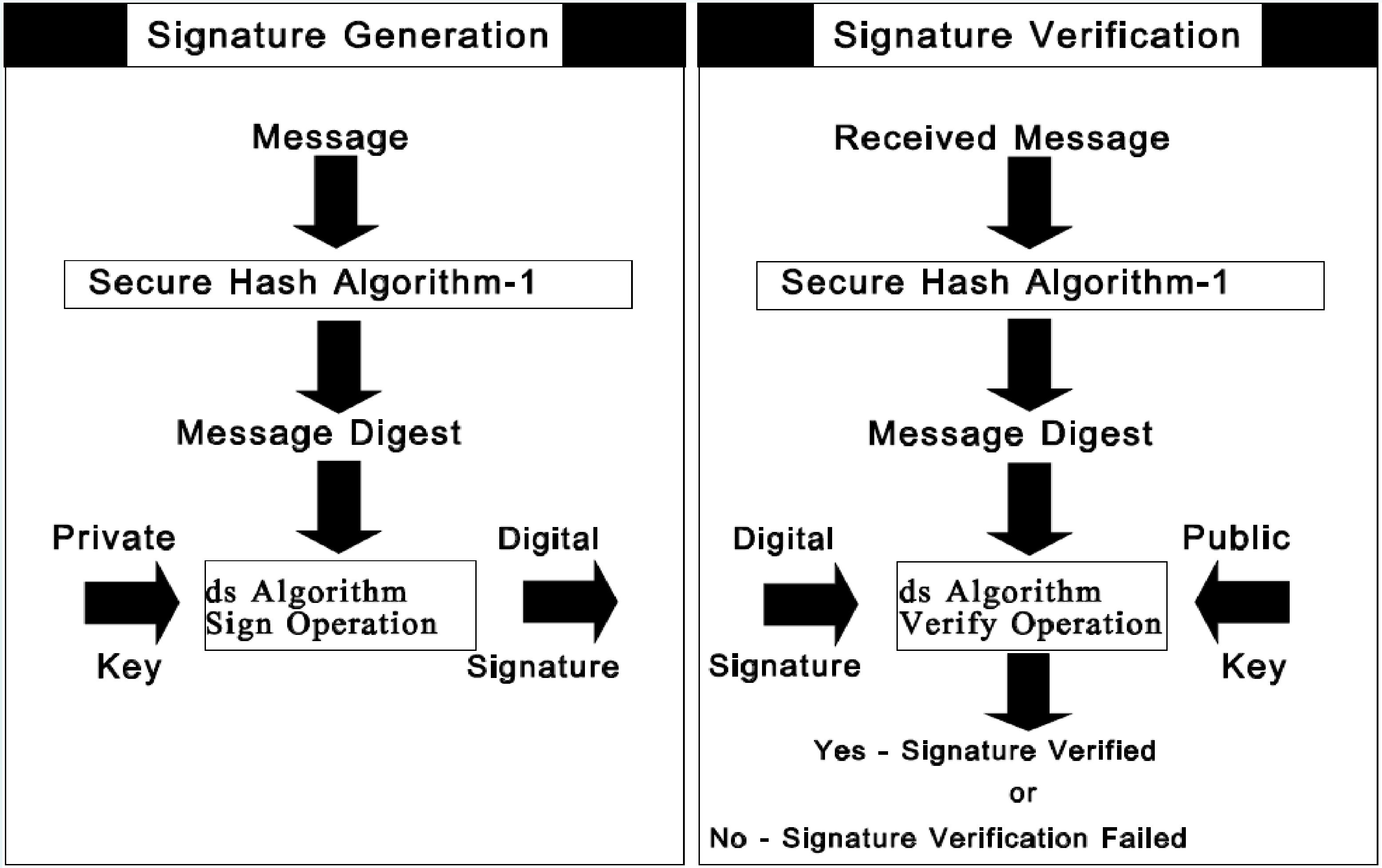
Homewor3
(Question2)

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 dat

Homewor3

(Question2)



Homewor3

(Question3)

```
install.packages("rjson",repos="http://cran.us.rproject.or")
library("rjson")
json_file = "http://crix.hu-berlin.de/data/crix.json"
json_data = fromJSON(file=json_file)
crix_data_frame = as.data.frame(json_data)
```

Homewor4

(Question1)

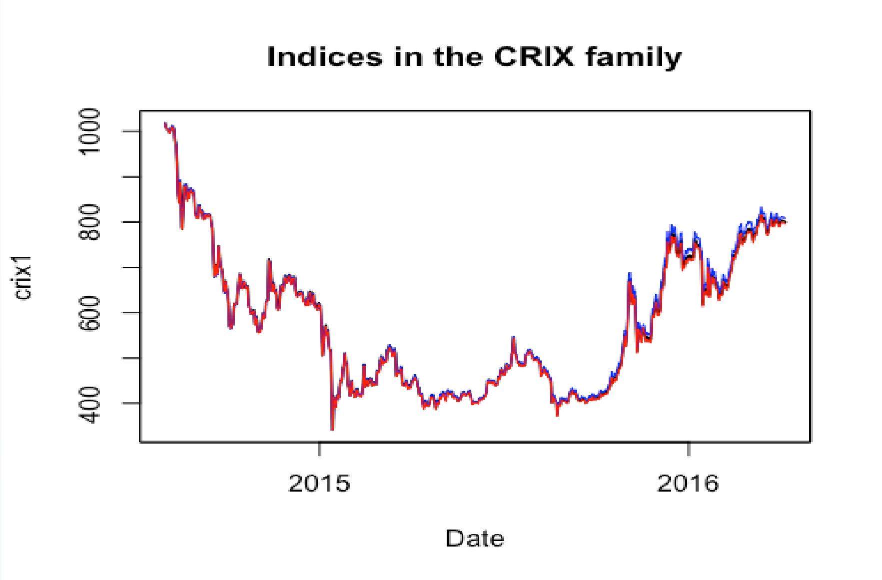


Figure 3: The daily value of indices in the CRIX family

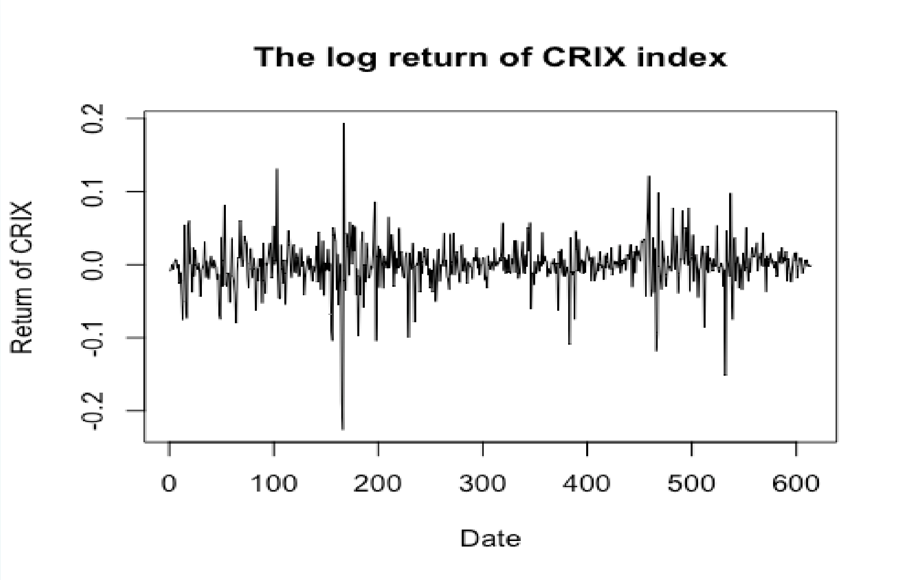
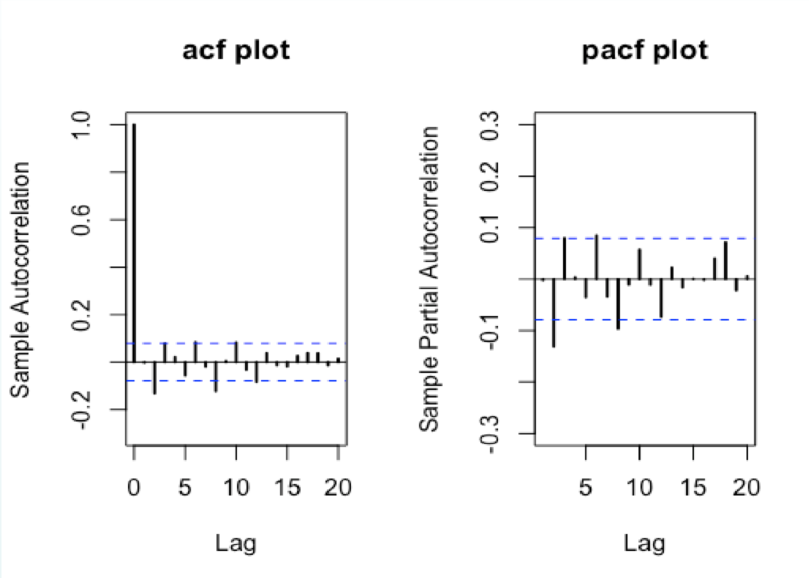


Figure 4: The log returns of CRIX index



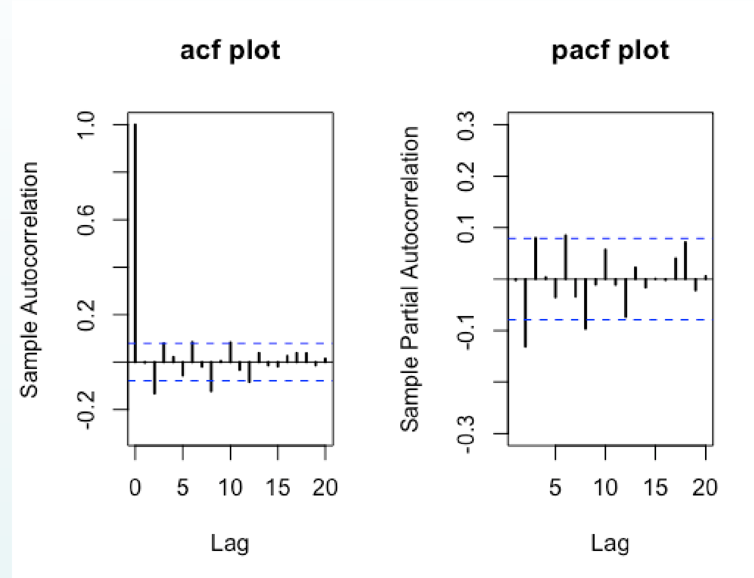


Figure 5: Histogram and QQ plot of CRIX returns

Figure 6: The sample ACF and PACF of CRIX returns

```
rm(list = ls(all = TRUE))
graphics.off()
# install and load packages
libraries = c("zoo", "tseries", "xts","ccgarch")
lapply(libraries, function(x) if (!(x %in% installed.packages())) { install.packages(x)}
```

```r
lapply(libraries, library, quietly = TRUE, character.only = TRUE)

# load dataset
load(file.choose())
load(file.choose())
load(file.choose())

# three indices return
ecrix1 = zoo(ecrix, order.by = index(crix1))
efcrix1 = zoo(efcrix, order.by = index(crix1))

# plot with different x-axis scales with zoo
my.panel <- function(x, ...) {
  lines(x, ...)
  lines(ecrix1, col = "blue")
  lines(efcrix1, col = "red")
}
plot.zoo(crix1, plot.type = "multiple", type = "l", lwd = 1.5, panel = my.panel,
         main = "Indices in the CRIX family", xlab = "Date")
```

```r
# plot of crix
# plot(as.xts(crix), type="l", auto.grid=FALSE, main = NA)
plot(crix1, ylab = "Price of CRIX", xlab = "Date")

# plot of crix return
ret   = diff(log(crix1))
# plot(as.xts(ret), type="l", auto.grid=FALSE, main = NA)
plot(ret, ylab = "Return of CRIX", xlab = "Date")

# stationary test
adf.test(ret, alternative = "stationary")
kpss.test(ret, null = "Trend")

par(mfrow = c(1, 2))
# histogram of returns
hist(ret, col = "grey", breaks = 20, freq = FALSE, ylim = c(0, 25), xlab = "Return of CRIX")
lines(density(ret), lwd = 2)
mu = mean(ret)
sigma = sd(ret)
x = seq(-4, 4, length = 100)
curve(dnorm(x, mean = mean(ret), sd = sd(ret)), add = TRUE, col = "red",
      lwd = 2)
```

```
# qq-plot
qqnorm(ret)
qqline(ret, col = "blue", lwd = 3)

# acf plot
autocorr = acf(ret, lag.max = 20, ylab = "Sample Autocorrelation", main = "acf plot",
              lwd = 2, ylim = c(-0.3, 1))

# pacf plot
autopcorr = pacf(ret, lag.max = 20, ylab = "Sample Partial Autocorrelation",
                main = "pacf plot", ylim = c(-0.3, 0.3), lwd = 2)
```

Homewor4

(Question2)

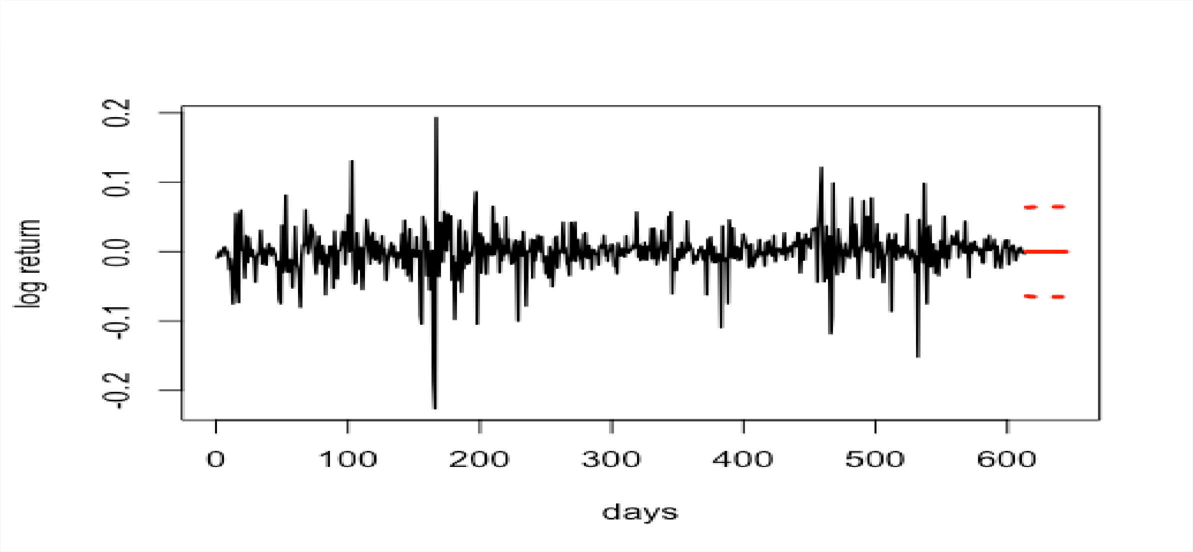


Figure 7: CRIX returns and predicted values.

Codes:

```
# arima model
par(mfrow = c(1, 1))
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
  }
}

# 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
  }
}

# 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)
```

Homewor4
(Question2)

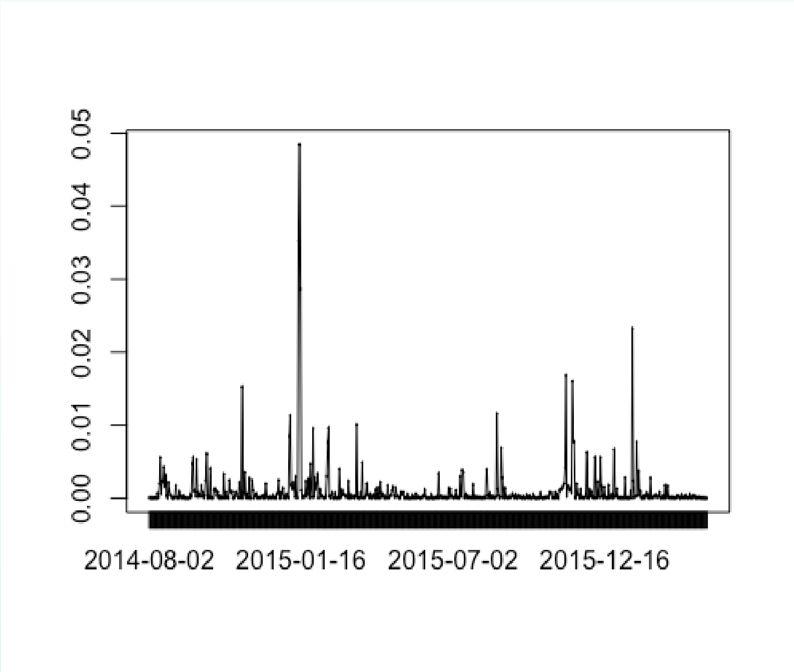


Figure 8: The squared ARIMA(2,0,2) residuals of CRIX returns.

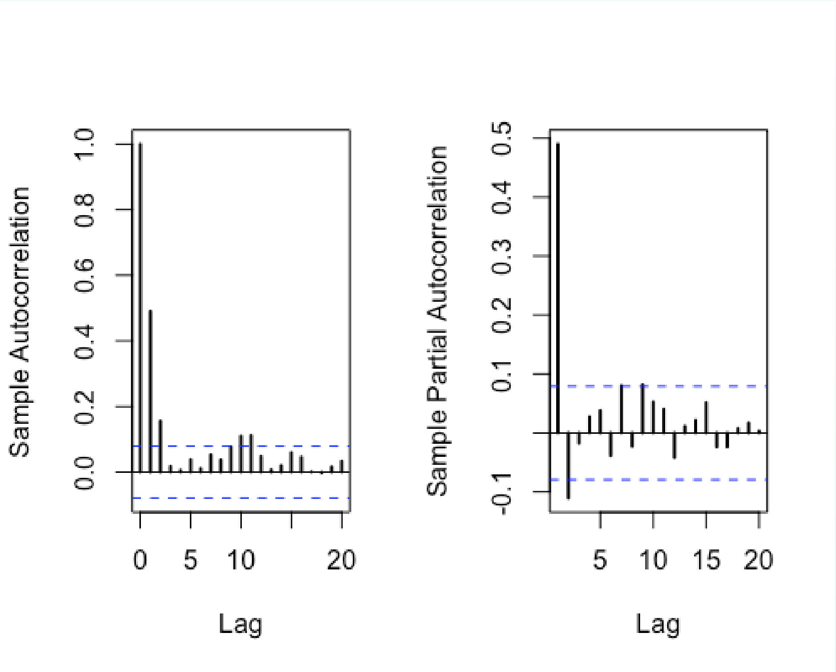


Figure 9: The ACF and PACF of squared ARIMA(2,0,2) residuals

Codes:

```
rm(list = ls(all = TRUE))
graphics.off()

# install and load packages
libraries = c("tseries")
lapply(libraries, function(x) if (!(x %in% installed.packages())) {
  install.packages(x)
})
lapply(libraries, library, quietly = TRUE, character.only = TRUE)

# please change your working directory
setwd()
load(file.choose())
Pr = as.numeric(crix)
Da = factor(date1)
crx = data.frame(Da, Pr)
# plot of crix return
ret = diff(log(crx$Pr))
Dare = factor(date1[-1])
retts = data.frame(Dare, ret)
# arima202 predict
fit202 = arima(ret, order = c(2, 0, 2))
```

```
# vola cluster
par(mfrow = c(1, 1))
res = fit202$residuals
res2 = fit202$residuals^2
tsres202 = data.frame(Dare,
res2)
plot(tsres202$Dare,
tsres202$res2, type = "o", ylab
= NA)
lines(tsres202$res2)

# plot(res2, ylab='Squared
residuals', main=NA)
par(mfrow = c(1, 2))
acfres2 = acf(res2, main = NA,
lag.max = 20, ylab = "Sample
Autocorrelation", lwd = 2)
pacfres2 = pacf(res2, lag.max
= 20, ylab = "Sample Partial
Autocorrelation", lwd = 2, main
= NA)
```

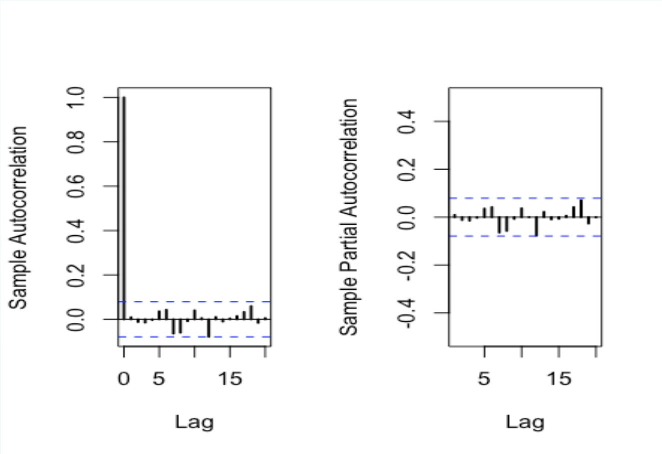


Figure 10: The ACF and PACF of squared ARIMA(2,0,2) residuals

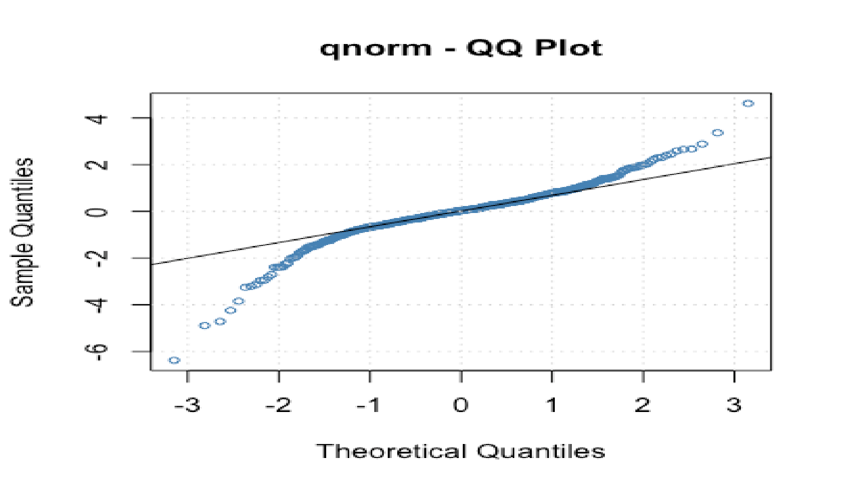


Figure 11: The QQ plots of model residuals of ARIMA-GARCH process.

Codes:

```
rm(list = ls(all = TRUE))
graphics.off()

# install and load packages
libraries = c("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.choose())
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
```

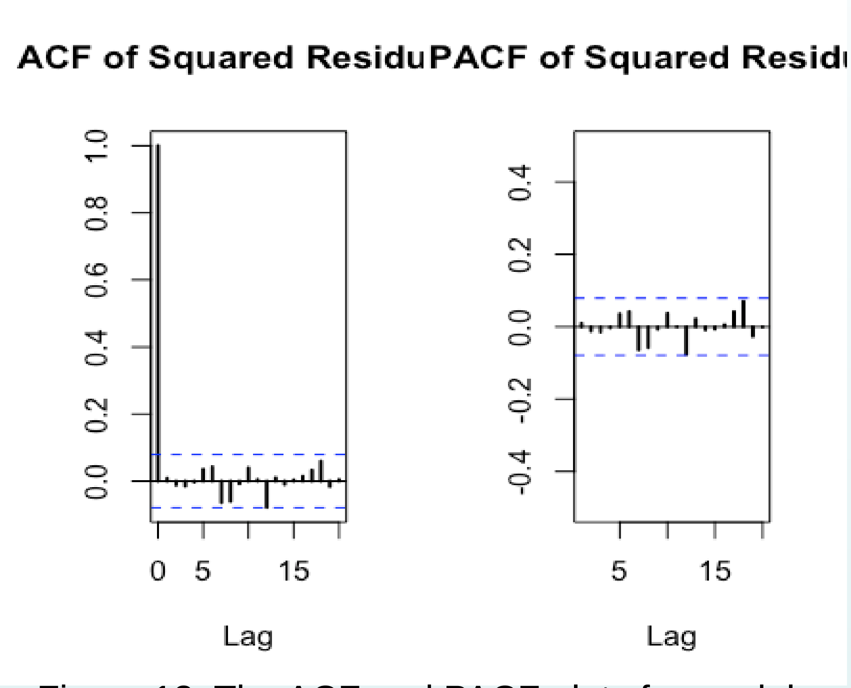
```
# different garch model
fg11 = garchFit(data = res, data
~ garch(1, 1))
summary(fg11)
fg12 = garchFit(data = res, data
~ garch(1, 2))
summary(fg12)
fg21 = garchFit(data = res, data
~ garch(2, 1))
summary(fg21)
fg22 = garchFit(data = res, data
~ garch(2, 2))
summary(fg22)

# residual plot
reszo = zoo(fg11@residuals,
order.by = index(crix1))
plot(reszo, ylab = NA, lwd = 2)
```

```r
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
```



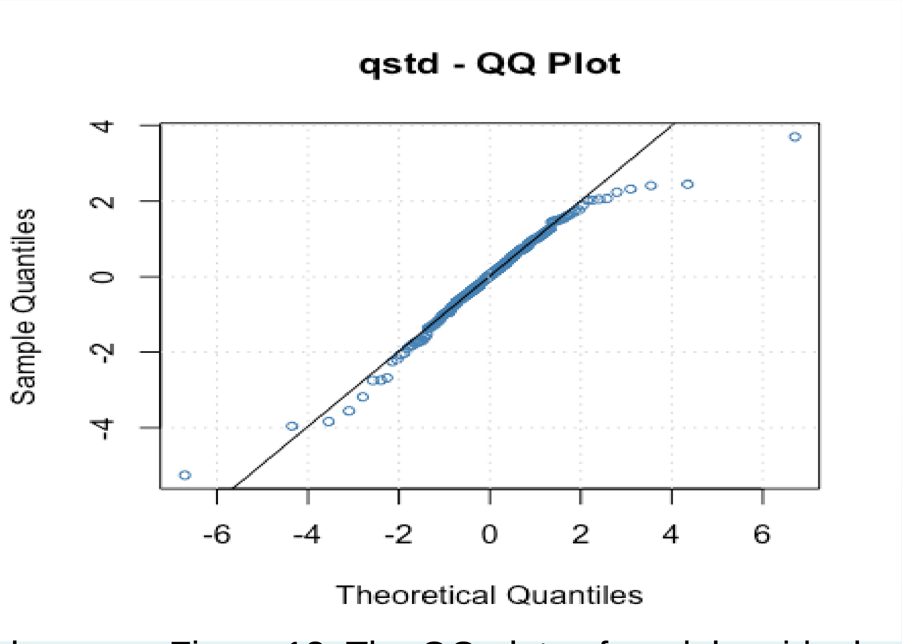


Figure 12: The ACF and PACF plots for model residuals of ARIMA(2,0,2)- t-GARCH(1,1) process.

Figure 13: The QQ plots of model residuals of ARIMA-t-GARCH process.

Codes:

```
fg11stu = garchFit(data = res, data ~ garch(1, 1), cond.dist = "std")

# different forecast with t-garch
# fg11stufore = predict(fg11stu, n.ahead = 30, plot=TRUE, mse='uncond', auto.grid=FALSE)
fg11stufore = predict(fg11stu, n.ahead = 30, plot = TRUE, cond.dist = "QMLE",
                      auto.grid = FALSE)

par(mfrow = c(1, 2))
stu.fg11res2 = fg11stu@residuals

# acf and pacf for t-garch
stu.acfres2 = acf(stu.fg11res2, ylab = NA, lag.max = 20, main = "ACF of Squared Residuals",
                  lwd = 2)
stu.pacfres2 = pacf(stu.fg11res2, lag.max = 20, main = "PACF of Squared Residuals",
                    lwd = 2, ylab = NA, ylim = c(-0.5, 0.5))

# ARIMA-t-GARCH qq plot
par(mfrow = c(1, 1))
plot(fg11stu, which = 13)
```