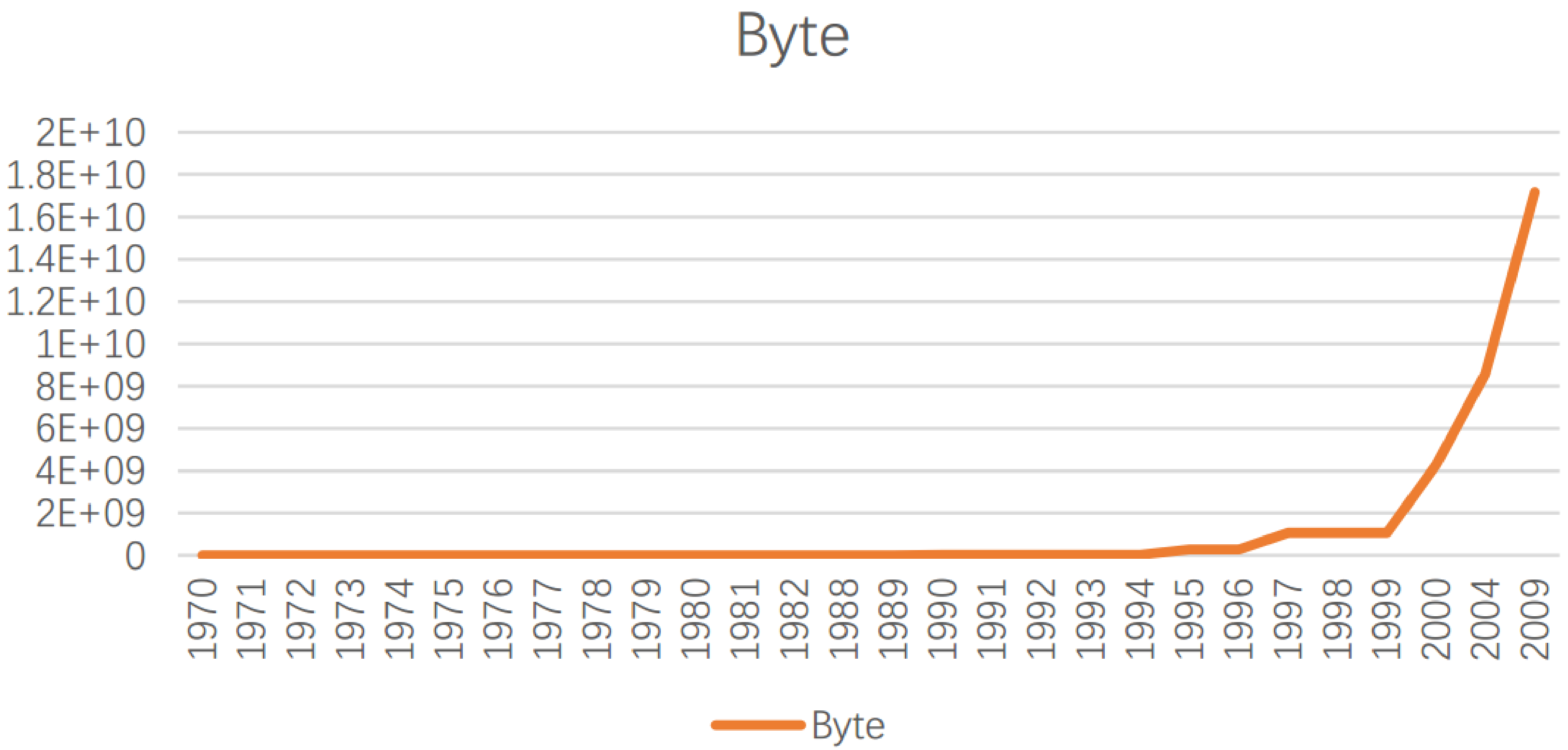
Final Exam

JingYan
15620161152292

HW1_1

This is the development path of memory of PCs



HW1_2

Logistic regression:

In statistics, logistic regression, or logit regression, or logit model 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. In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

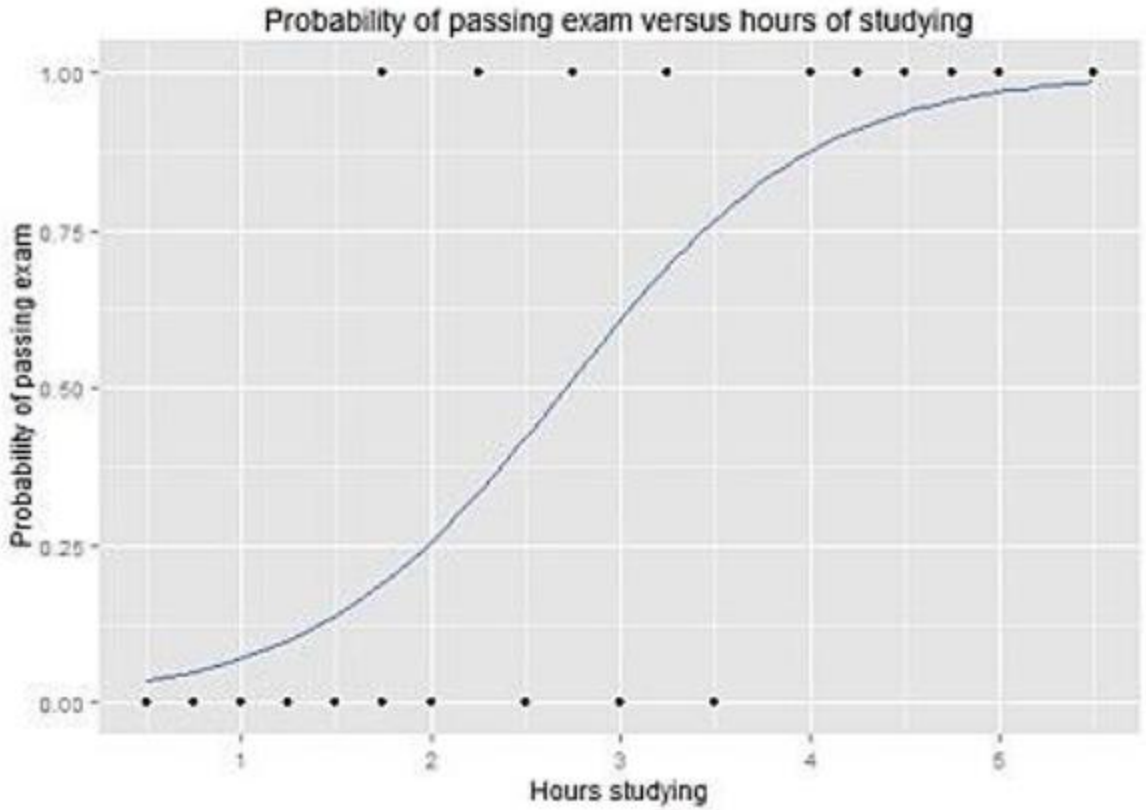
HW1_2

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

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

HW1_2



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

HW2_1

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

HW2_2

```
f<-read.csv("hw unit2.csv")
year<-f$Year;RAM<-f$RAM
plot(year,RAM,type ="o",col="black",main = "RAM of computer")

require(datasets)
require(class)
require(grDevices)
require(lattice)

x= year
y = RAM

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

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
```

HW2_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 )
```

HW3_1

```
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")
```

HW3_2

WHAT IS DSA

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.

HW3_2

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 。

HW3_3

```
>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))
```

HW3_3

```
[ {
"letter": "A",
"country": "China"
},
{
"letter": "B",
"country": "the US"
},
{
"letter": "C",
"country": "the UK"
},
{
"letter": "D",
"country": "Russia"
},
{
"letter": "E",
"country": "Korea"
},
```

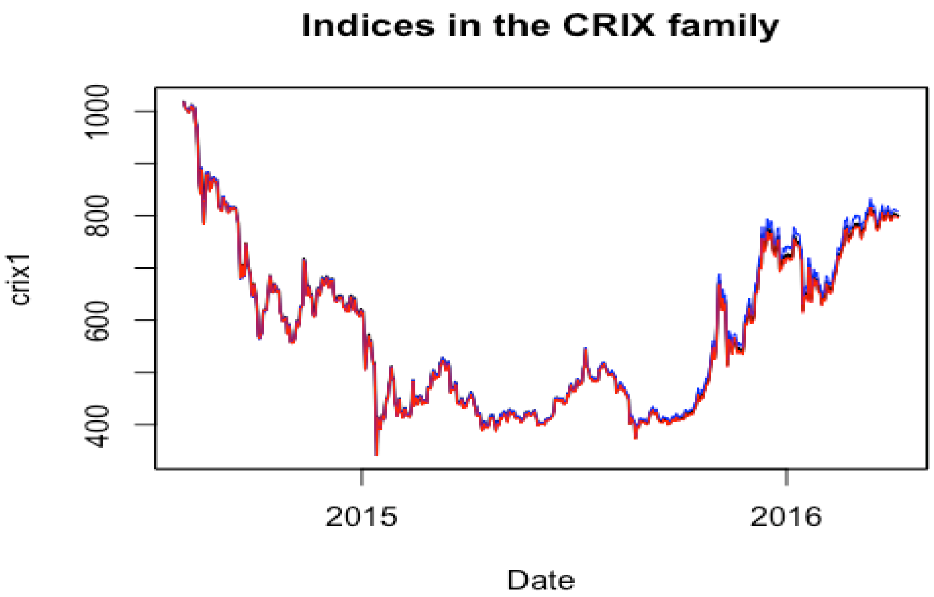
HW3_3

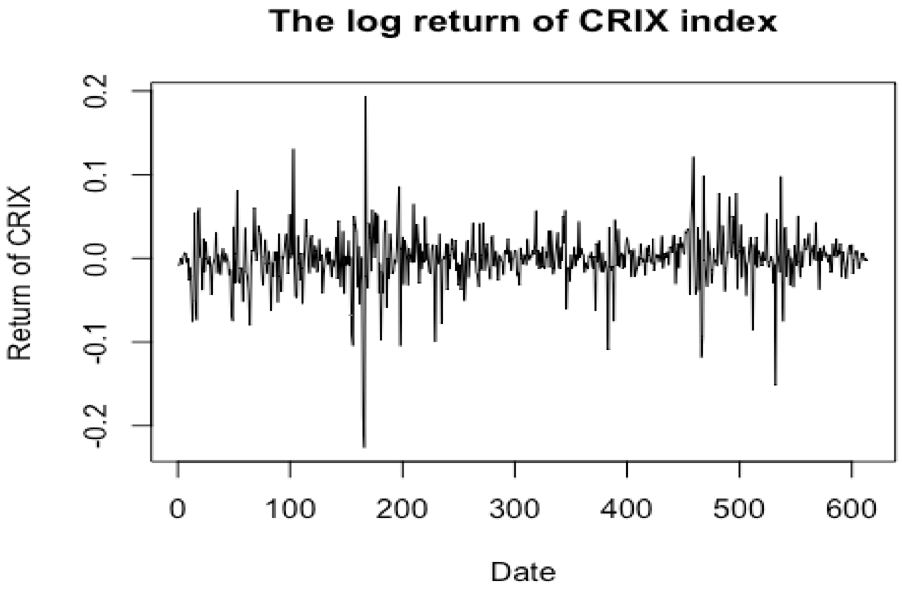
{
"letter": "F",
"country": "Japan"
},
{
"letter": "G",
"country": "Italy"
},
{
"letter": "H",
"country": "Brazil"
},
{
"letter": "I",
"country": "India"
},
{
"letter": "J",
"country": "Germany"
}]

HW3_4

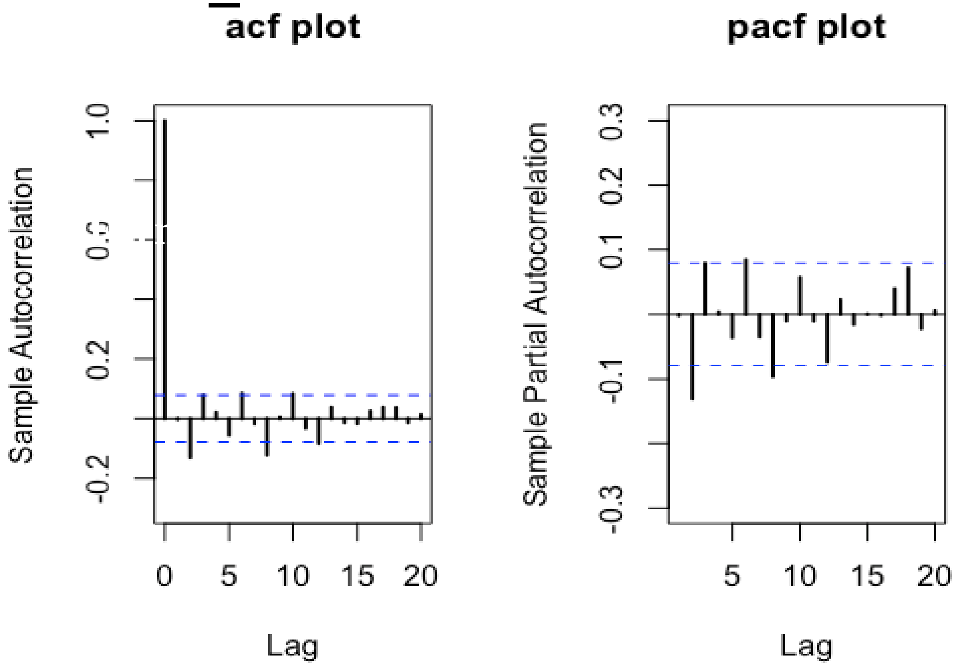
```
#install.packages("rjson", repos="http://cran.us.r-project.org")
library(rjson)
json_file = "http://crix.hu-berlin.de/data/crix.json"
json_data = fromJSON(file=json_file)
crix <- Reduce(rbind,json_data)
crix_data_frame <- as.data.frame(crix)
lst <- lapply(json_data,function(x)
{
df<-data.frame(date=x$date,price=x$price)
return(df)
})
crix_data_frame <- Reduce(rbind,lst)
plot(crix_data_frame$date,crix_data_frame$price)
#library(forecast)
#library(tseries)
plot(crix_data_frame)
```

HW4 1

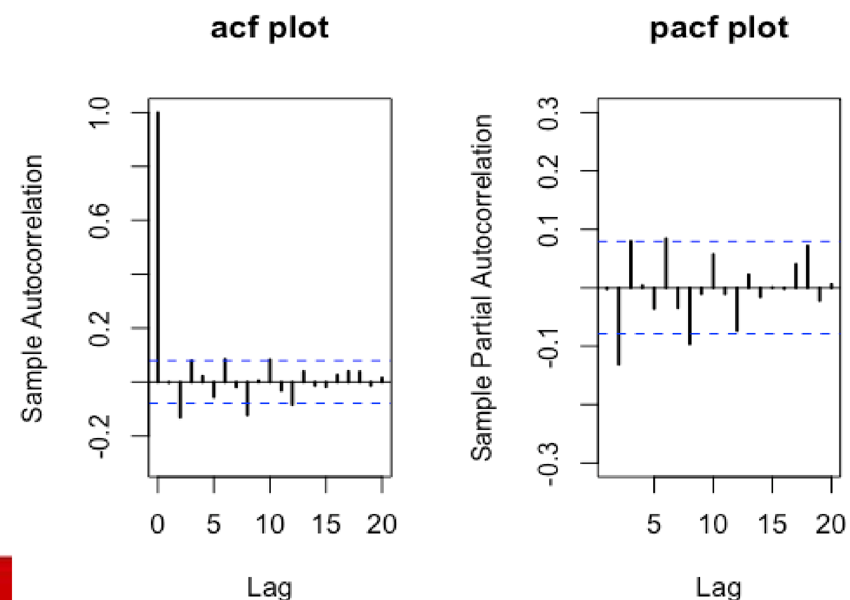

Indices in the CRIX family
crix1
1000
800
600
400
2015
2016
Date


The log return of CRIX index
Return of CRIX
0.2
0.1
0.0
-0.1
-0.2
0
100
200
300
400
500
600
Date

HW4_1




```
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)}
```



HW4_1

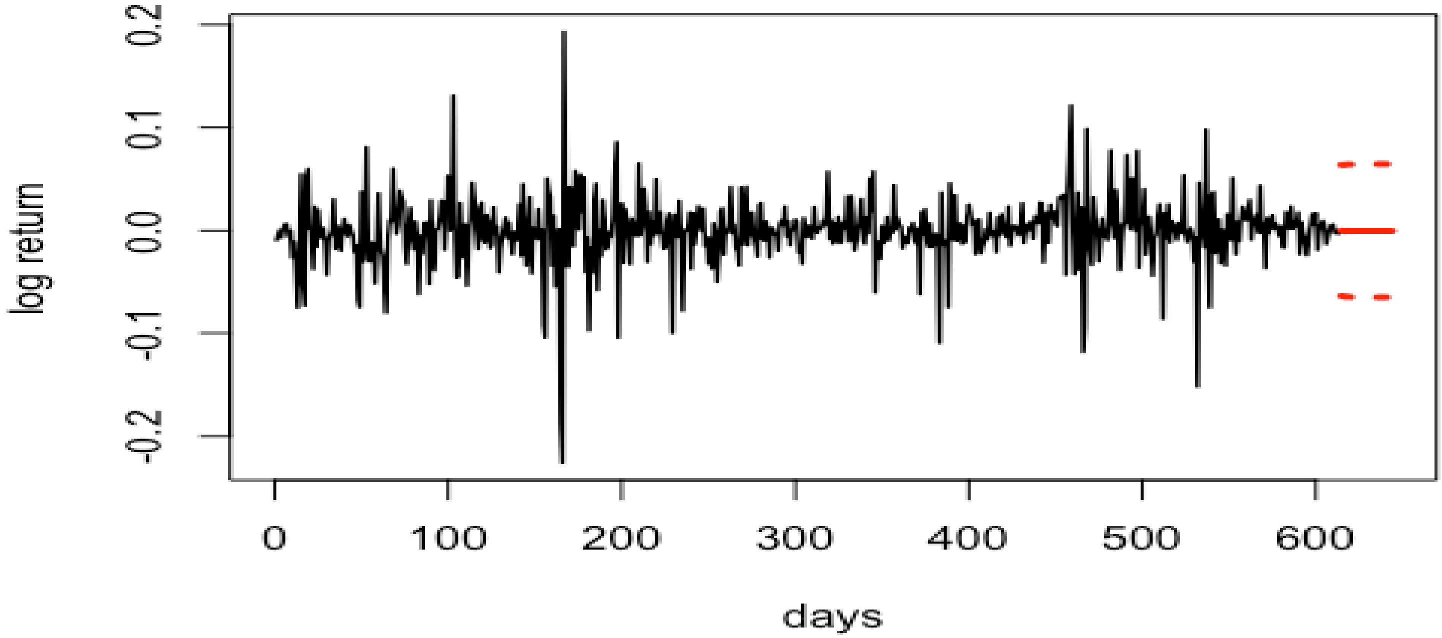
```r
lapply(libraries, library, quietly = TRUE, character.only = TRUE)

# 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")
```

```
# 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)
```

HW4_1

```
# 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)
```

HW4_2



HW4_2

Codes:

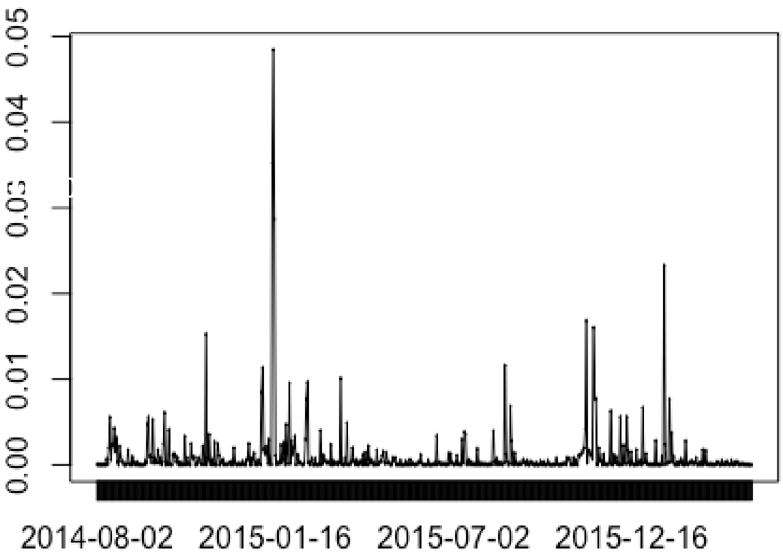
```
# arima model
par(mfrow = c(1, 1))
fit1 = arima(ret, order = c(1, 0, 1))
tsdiag(fit1)
Box.test(fit1$residuals, lag = 1)

# 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
  }
}
```

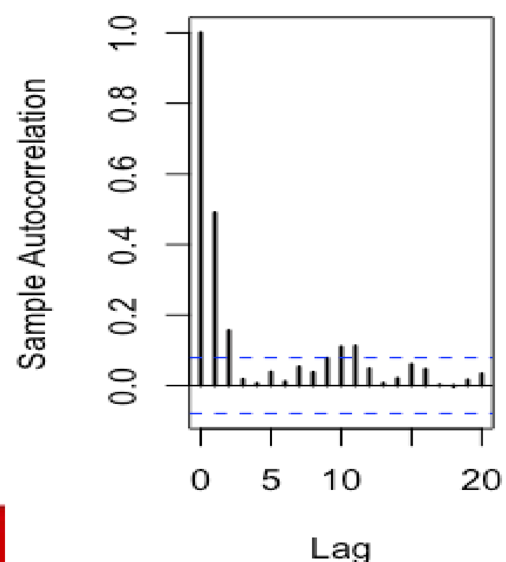
HW4_2

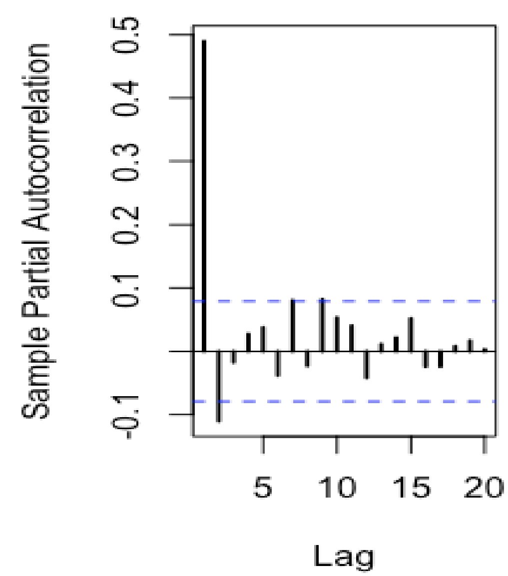
```
# bic
bic = matrix(NA, 6, 6)
for (p in 0:4) {
  for (q in 0:3) {
    b.p.q = arima(ret, order = c(p, 0, q))
    bic.p.q = AIC(b.p.q, k = log(length(ret)))
    bic[p + 1, q + 1] = bic.p.q
  }
}

# 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)
```

```
# to conclude, 202 is better than 213
fit202 = arima(ret, order = c(2, 0, 2))

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
predict_num = 30
fit202 = arima(ret, order = c(2, 0, 2))
crpre = predict(fit202, n.ahead = predict_num)

dates = seq(as.Date("02/08/2014", format = "%d/%m/%Y"), by = "days", length =
length(ret))
plot(ret, type = "l", xlim = c(0, length(ret)+predict_num), ylab = "log return", xlab
= "days",
    lwd = 1.5, col = "black")
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)
```

HW4_3


0.00
0.01
0.02
0.03
0.04
0.05
2014-08-02
2015-01-16
2015-07-02
2015-12-16


Sample Autocorrelation
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
20
Lag


Sample Partial Autocorrelation
-0.1
0.1
0.2
0.3
0.4
0.5
5
10
15
20
Lag

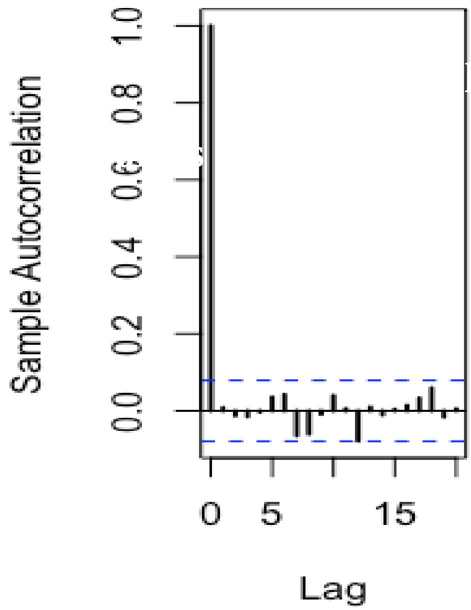
Codes:

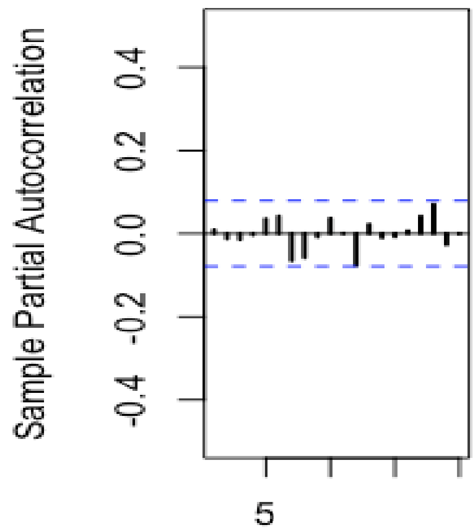
```
rm(list = ls(all = TRUE))
graphics.off()

# 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))
```

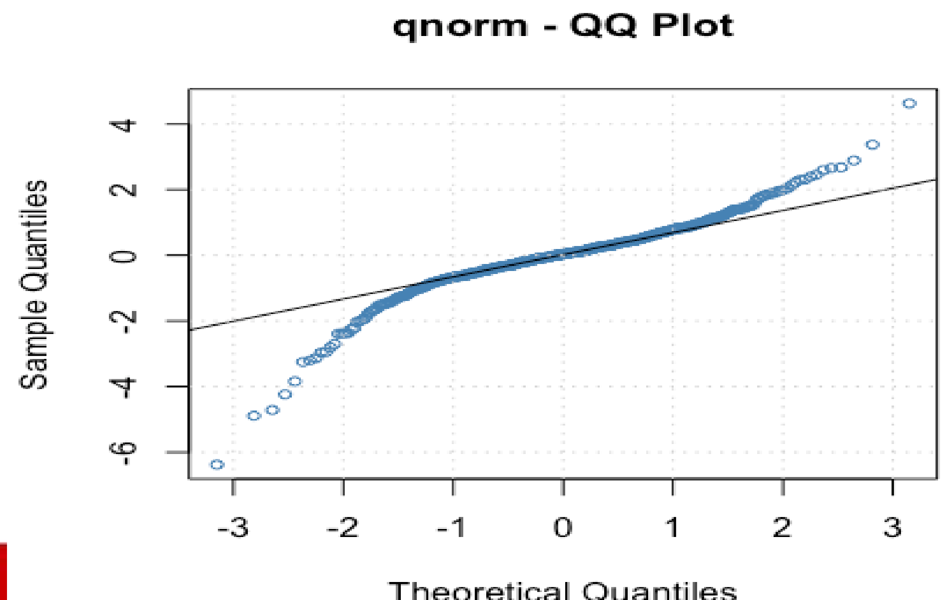
HW4_3

```
# 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)
```

HW4_3


Sample Autocorrelation
0.0
0.2
0.4
0.6
0.8
1.0
0
5
15
Lag


Sample Partial Autocorrelation
-0.4
-0.2
0.0
0.2
0.4
5


qnorm - QQ Plot
Sample Quantiles
-6
-4
-2
0
2
4
-3
-2
-1
0
1
2
3
Theoretical Quantiles

HW4_3

```
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
```

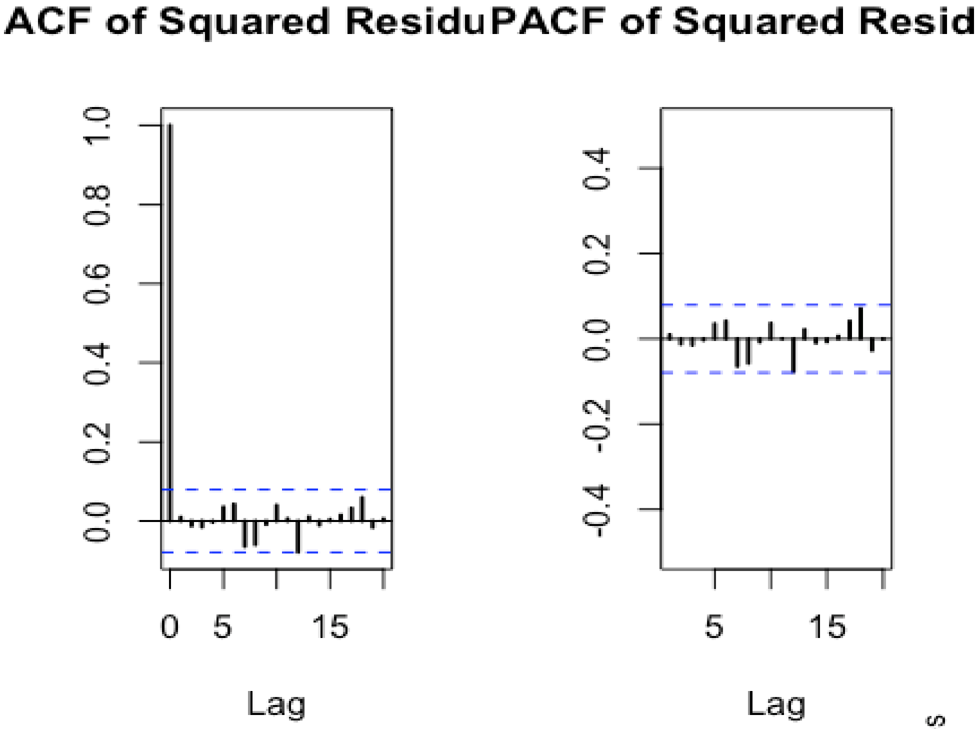
HW4_3

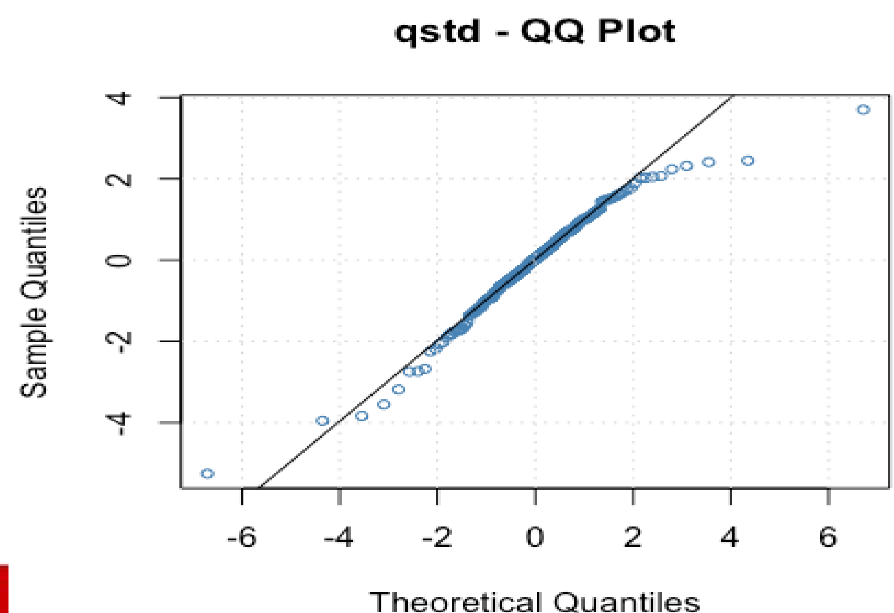
```
# different garch model
fg11 = garchFit(data = res, data ~ garch(1, 1))
summary(fg11)
fg12 = garchFit(data = res, data ~ garch(1, 2))
summary(fg12)
fg21 = garchFit(data = res, data ~ garch(2, 1))
summary(fg21)
fg22 = garchFit(data = res, data ~ garch(2, 2))
summary(fg22)

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

HW4_3

```
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
```

HW4_3


ACF of Squared Residu
PACF of Squared Resid
Lag
Lag


qstd - QQ Plot
Sample Quantiles
Theoretical Quantiles

```
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)
```