Final Exam

Yongshi JIE
Xiamen University
Master Finance

UNIT 1

1. Calculate the increase of memory of PCs over the last 30 years and check whether the FMRI analysis could have been done 20 years ago.

year	Byte
1970	262144
1971	262144
1972	262144
1973	262144
1974	262144
1975	262144
1976	262144
1977	262144
1978	262144
1979	262144
1980	262144
1981	262144
1982	262144
1988	2097152
1989	2097152
1990	2097152
1991	16777216
1992	16777216
1993	16777216
1994	16777216
1995	16777216
1996	268435456
1997	268435456
1998	1073741824
1999	1073741824
2000	1073741824
2004	4294967296
2009	8589934592
2014	17179869184

2. Logistic regression

In statistics, logistic regression, or logit regression, is a mathematical model used in statistics to estimate the probability of an event occuring having been given some previous data. Logistic Regression works with binary data, where either the event happens (1) or the event does not happen (0). So given some feature x it tries to find out whether some event y happens or not.

For example, if y represents whether a sports team wins a match, then y will be 1 if they win the match or y will be 0 if they do not. This is known as Binomial Logistic Regression. There is also another form of Logistic Regression which uses multiple values for the variable y. This form of Logistic Regression is known as Multinomial Logistic Regression.

Logistic regression does not look at the relationship between the two variables as a straight line. Instead, Logistic regression uses the natural logarithm function to find the relationship between the variables and uses test data to find the coefficients. The function can then predict the future results using these coefficients in the logistic equation.

Logistic regression uses the concept of odds ratios to calculate the probability. This is defined as the ratio of the odds of an event happening to its not happening.

$$Odds = \frac{P(y=1|x)}{1-P(y=1|x)}$$

The natural logarithm of the odds ratio is then taken in order to create the logistic equation is know as the logit:

$$Logit(P(x)) = \ln(\frac{P(y=1|x)}{1-P(y=1|x)})$$

In Logistic regression the Logit of the probability is said to be linear with respect to x, so the logit becomes:

$$Logit(P(x)) = a + bx$$

Using the two equations together then gives the following:

$$\ln\left(\frac{P(y=1|x)}{1-P(y=1|x)}\right) = a + bx$$

Using the two equations together then gives the following:

$$\frac{P(y=1|x)}{1-P(y=1|x)} = e^{a+bx}$$

This then leads to the probability:

$$P(y=1|x) = \frac{e^{a+bx}}{1+e^{a+bx}} = \frac{1}{1+e^{-(a+bx)}}$$

The final equation is the logistic regression.

We can use the vector form and the logistic equation can be changed to:

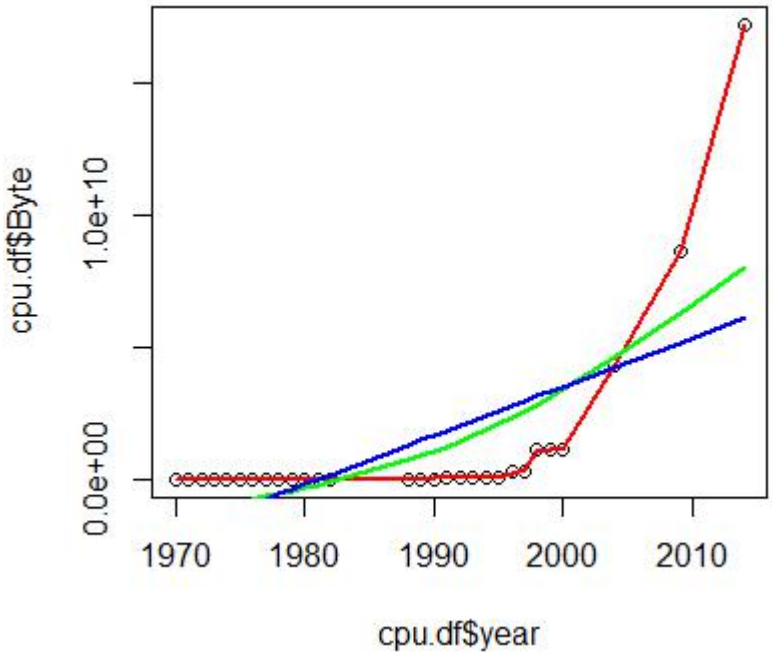
$$P(y=1|x) = \frac{1}{1+e^{-(w^T x)}}$$

UNIT 1

UNIT 2

Code and Result

```
##################HW2#####################

# EX1&EX2
cpu.df = read.csv("byte.csv",header = TRUE)
plot(cpu.df$Byte~cpu.df$year)

splines.reg.l1 = smooth.spline(x = cpu.df$year, y = cpu.df$Byte, spar = 0.2)
splines.reg.l2 = smooth.spline(x = cpu.df$year, y = cpu.df$Byte, spar = 1)
splines.reg.l3= smooth.spline(x = cpu.df$year, y = cpu.df$Byte, spar = 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
```




```
# EX3
lambda=4
x=6
probex1=exp(-lambda)*lambda^x/factorial(x)
lambda=5
x=0
probex2=exp(-lambda)*lambda^x/factorial(x)
```

```
> lambda=2
> x=3
> probex1=exp(-lambda)*lambda^x/factorial(x)
> probex1
[1] 0.180447

> lambda=5
> x=0
> probex2=exp(-lambda)*lambda^x/factorial(x)
> probex2
[1] 0.006737947
```

UNIT 1

UNIT 2

UNIT 3

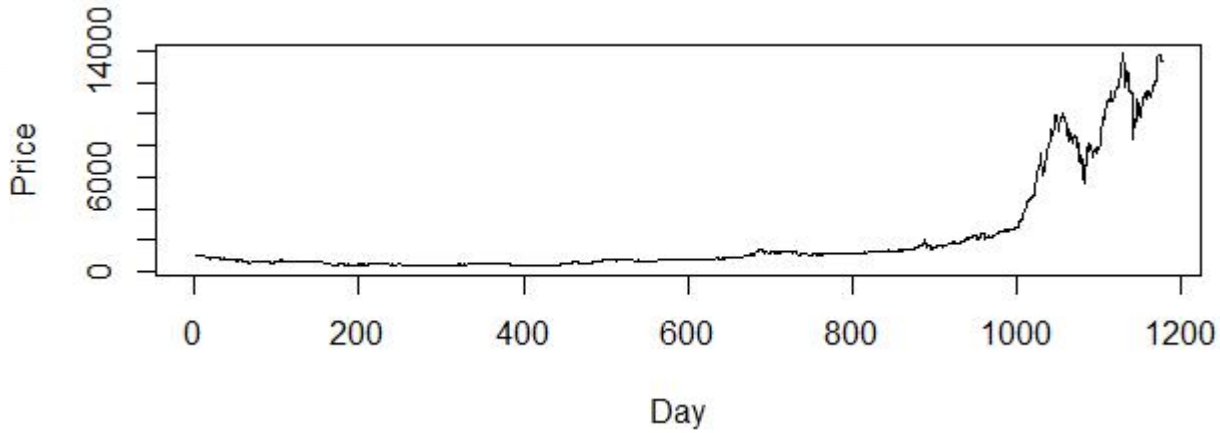
1. Code and Result

```
############## HW3 ###############
#1
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")
```

```
> digest("I learn a lot from this class when I am proper listening to the professor"
,"sha256")
[1] "c16700de5a5c1961e279135f2be7dcf9c187cb6b21ac8032308c715e1ce9964c"
> digest("I do not learn a lot from this class when I am absent and playing on my Ip
hone","sha256")
[1] "2533d529768409d1c09d50451d9125fdbaa6e5fd4efdeb45c04e3c68bcb3a63e"
```

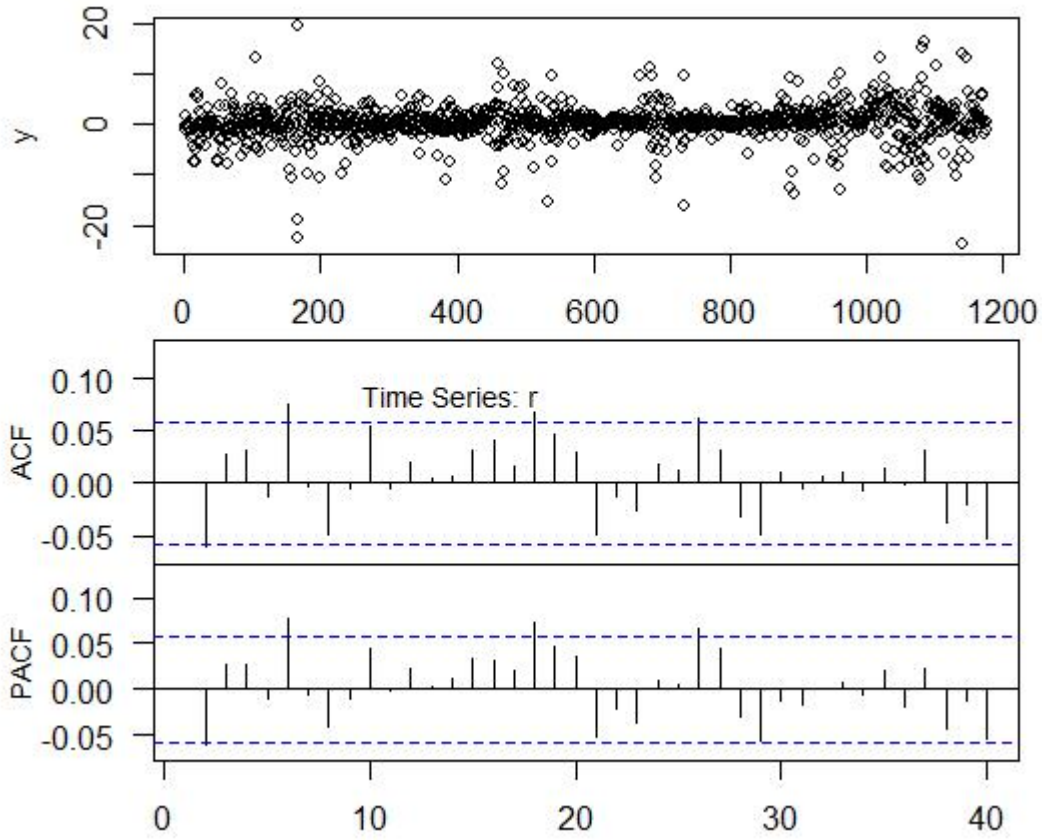
```
#3
library(rjson)
json_file = "http://crix.hu-berlin.de/data/crix.json"
json_data = fromJSON(file=json_file)
x = as.data.frame(json_data)
date1=c(json_data[[1]]$date)
for (i in 1:50){
  date1[i]=c(json_data[[i]]$date)
}
price1=c(json_data[[1]]$price)
for (i in 1:50){
    price1[i]=c(json_data[[i]]$price)
}
```

```
date=date1
price=price1
crix=data.frame(date,price)

plot(crix$price~as.Date(crix$date))
plot(crix$price~crix$date,type="b")
plot(ts(crix$price,freq=1),type='l',xlab='Day',ylab='Price')
```



Code and Result

```
library(caschrono)
library(TTR)
library(fGarch)
library(rugarch)
library(forecast)
library(TSA)

#*******************ARIMA model******************
xy.acfb(crix$price,numer=FALSE)
adf.test(crix$price)
#Augmented Dickey-Fuller Test:not stationary

##*****1)return
r=diff(log(crix$price))*100
plot(r,type="b")
abline(h = 0)
plot(r,type="l")
xy.acfb(r,numer=FALSE)
```



Code and Result

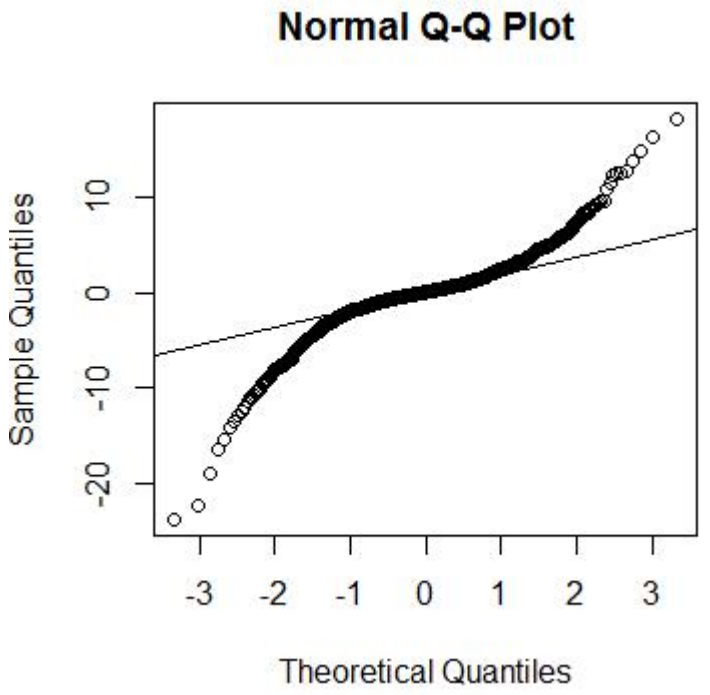
```
#*****2)Parameter Estimation
#estimation of p and q

a.fin2=arima(r,order=c(2,0,2))
summary(a.fin2)
f=forecast(a.fin2,h=3,level=c(99.5))
acf(f$residuals,lag.max = 20)
Box.test(f$residuals,lag=20,type='Ljung-Box')
#the residuals follow Gaussian distribution
plot.ts(f$residuals)

#****3)some evidence to GARCH model
#get ACF and PACF of the residuals
xy.acfb(residuals(a.fin2),numer=FALSE)
xy.acfb((residuals(a.fin2))^2,numer=FALSE)+
xy.acfb(abs(residuals(a.fin2)),numer=FALSE)

#get the Conditional heteroskedasticity test
McLeod.Li.test(y=residuals(a.fin2))
#p-values are all included in the test, it formally
shows strong evidence for ARCH in this data.
```

```
#**Normality of the Residuals
qqnorm(residuals(a.fin2))
qqline(residuals(a.fin2))
shapiro.test(residuals(a.fin2))
#The QQ plot suggest that the distribution of returns may have a tail
thicker that of a
#normal distribution and maybe somewhat skewed to the right
#p-value<0.05 reject the normality hypothesis

g1=garchFit(~garch(1,1),data=residuals(a.fin2),trace=FALSE,include.me
an=TRUE, na.action=na.pass)
summary(g1)
g2=garchFit(~garch(1,2),data=residuals(a.fin2),trace=FALSE,include.me
an=TRUE, na.action=na.pass)
summary(g2)
g3=garchFit(~garch(2,1),data=residuals(a.fin2),trace=FALSE,include.me
an=TRUE, na.action=na.pass)
summary(g3)
g4=garchFit(~garch(2,2),data=residuals(a.fin2),trace=FALSE,include.me
an=TRUE, na.action=na.pass)
summary(g4)
#The best one is Garch(1,1) model which has the smallest AIC.
```

Code and Result

```
#******2)Parameter Estimation
#estimation of p and q

a.fin2=arima(r,order=c(2,0,2))
summary(a.fin2)
f=forecast(a.fin2,h=3,level=c(99.5))
acf(f$residuals,lag.max = 20)
Box.test(f$residuals,lag=20,type='Ljung-Box')
#the residuals follow Gaussian distribution
plot.ts(f$residuals)

#*****3)some evidence to GARCH model
#get ACF and PACF of the residuals
xy.acfb(residuals(a.fin2),numer=FALSE)
xy.acfb((residuals(a.fin2))^2,numer=FALSE)+
xy.acfb(abs(residuals(a.fin2)),numer=FALSE)

#get the Conditional heteroskedasticity test
McLeod.Li.test(y=residuals(a.fin2))
#p-values are all included in the test, it formally
shows strong evidence for ARCH in this data.
```

```
#**Normality of the Residuals
qqnorm(residuals(a.fin2))
qqline(residuals(a.fin2))
shapiro.test(residuals(a.fin2))
#The QQ plot suggest that the distribution of returns may have a tail
thicker that of a
#normal distribution and maybe somewhat skewed to the right
#p-value<0.05 reject the normality hypothesis
```



2. DSA

The Digital Signature Algorithm (DSA) is a **United States Federal Government standard for digital signatures.**

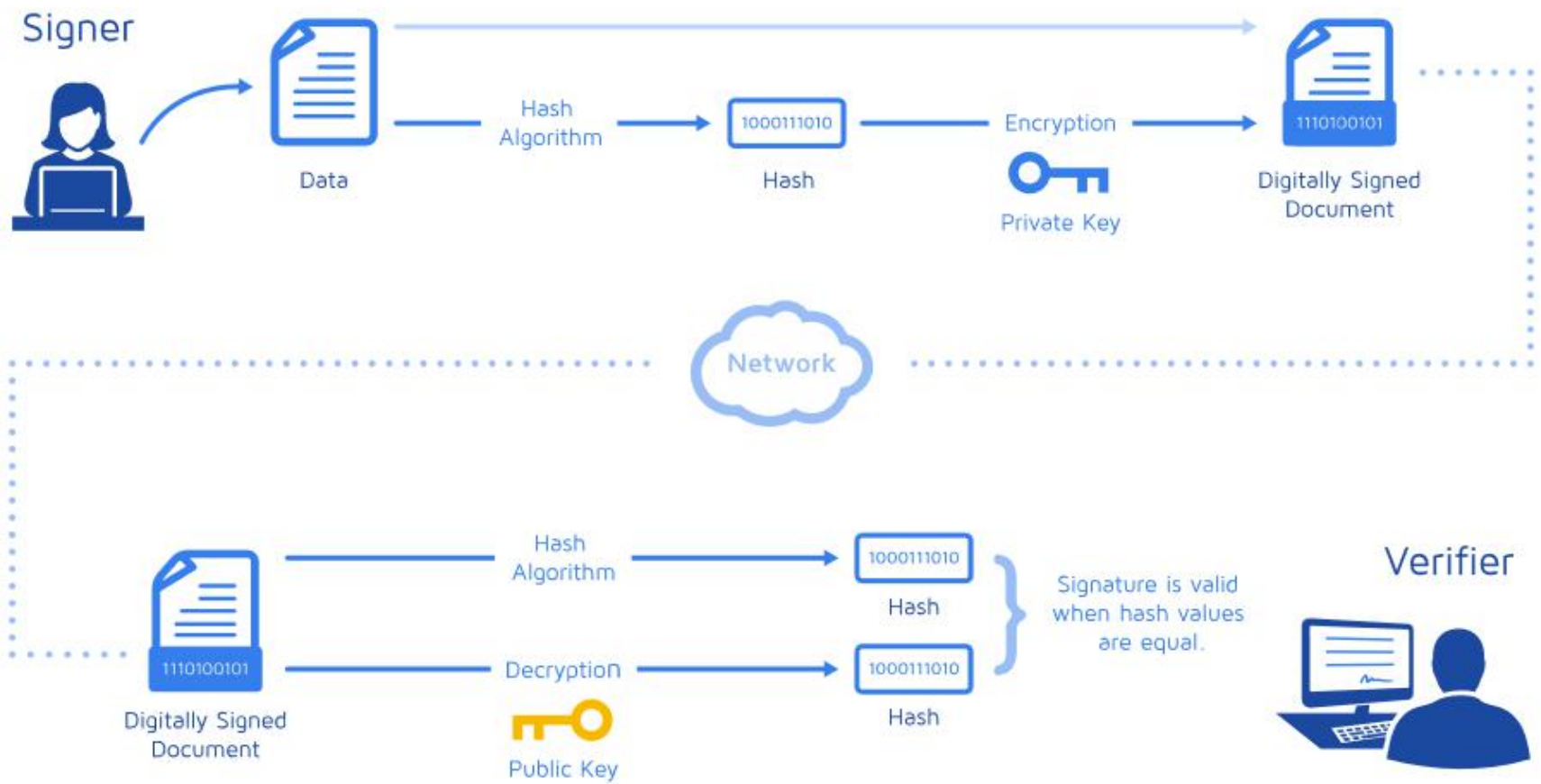
DSA was proposed by the National Institute of Standards and Technology (NIST) in August 1991 for use in their Digital Signature Standard (DSS), specified in FIPS 186.

DSA can be used by the recipient of a message to **verify that the message has not been altered during transit** as well as as **certain the originator's identity**

How do digital signatures work?

2. DSA

How do digital signatures work?



2. DSA

The Digital Signature is usually performed in several steps:

1. **Calculate the Message Digest**(hash-value of the message)

In the first step of the process, a hash-value of the message (often called the message digest) is calculated by applying some cryptographic hashing algorithm

2. Calculate the Digital Signature

In the second step of digitally signing a message, the information obtained in the first step hash-value of the message (the message digest) is encrypted with the private key of the person who signs the message and thus an encrypted hash-value, also called digital signature, is obtained. For this purpose, some mathematical cryptographic encrypting algorithm for calculating digital signatures from given message digest is used, which includes **DSA, TSA, ECDSA and so on**.

3. **Verifying Digital Signatures**

The public key is used in the signature verification process to verify the authenticity of the signature

2. DSA

The Digital Signature is usually performed in several steps:

1. **Calculate the Message Digest**(hash-value of the message)

In the first step of the process, a hash-value of the message (often called the message digest) is calculated by applying some cryptographic hashing algorithm

2. Calculate the Digital Signature

In the second step of digitally signing a message, the information obtained in the first step hash-value of the message (the message digest) is encrypted with the private key of the person who signs the message and thus an encrypted hash-value, also called digital signature, is obtained. For this purpose, some mathematical cryptographic encrypting algorithm for calculating digital signatures from given message digest is used, which includes **DSA, TSA, ECDSA and so on**.

3. **Verifying Digital Signatures**

The public key is used in the signature verification process to verify the authenticity of the signature

UNIT 1

UNIT 2

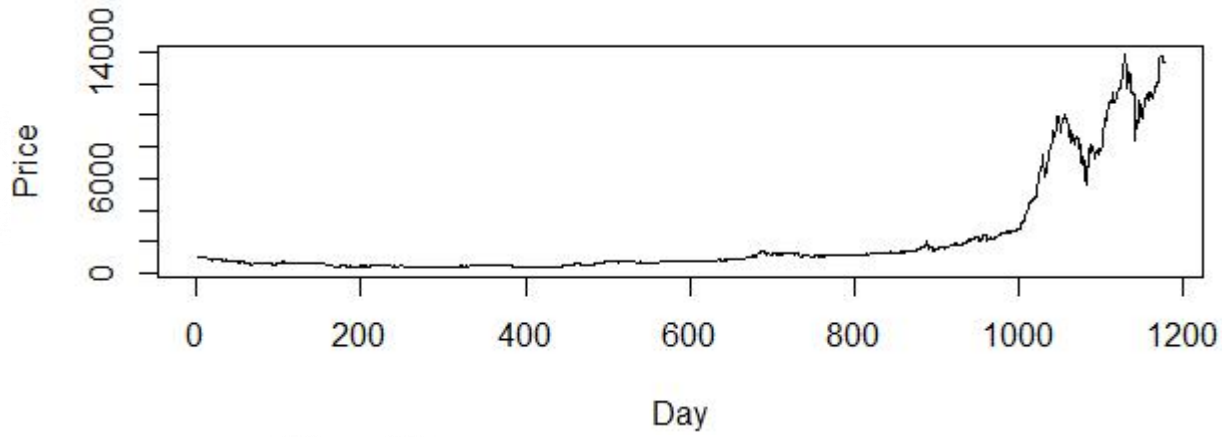
UNIT 3

UNIT 4

1. Code and Result

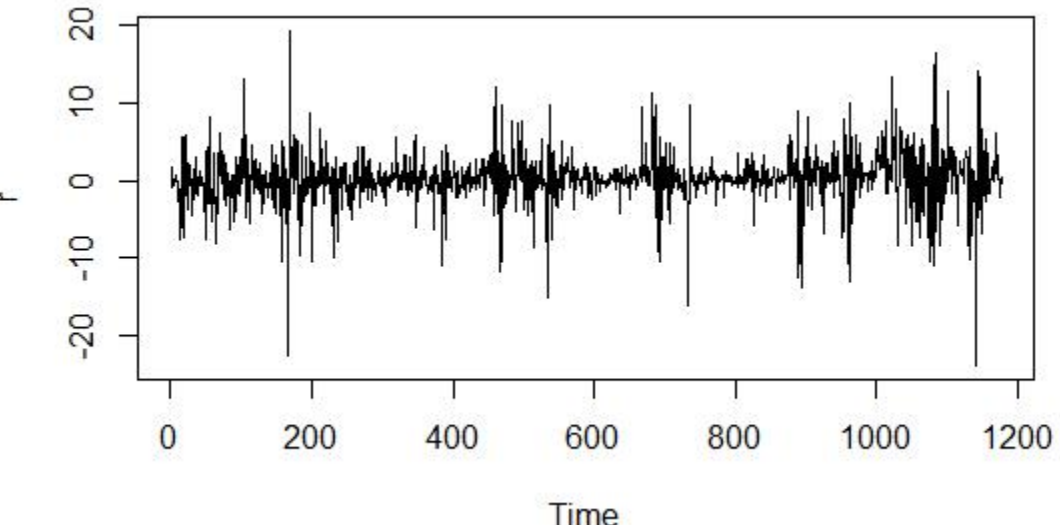
```
############## HW4 ###############
#1&2
library(rjson)
json_file = "http://crix.hu-berlin.de/data/crix.json"
json_data = fromJSON(file=json_file)
x = as.data.frame(json_data)
date1=c(json_data[[1]]$date)
for (i in 1:1177){
  date1[i]=c(json_data[[i]]$date)
}
price1=c(json_data[[1]]$price)
for (i in 1:1177){
  price1[i]=c(json_data[[i]]$price)
}
date=date1
price=price1
crix=data.frame(date,price)
```

```
##figure 3:crix&ecrix%efcrix
plot(ts(crix$price,freq=1),type='l',xlab='Day',ylab='Price')

library(caschrono)
library(TTR)
library(fGarch)
library(rugarch)
library(forecast)
library(TSA)
```

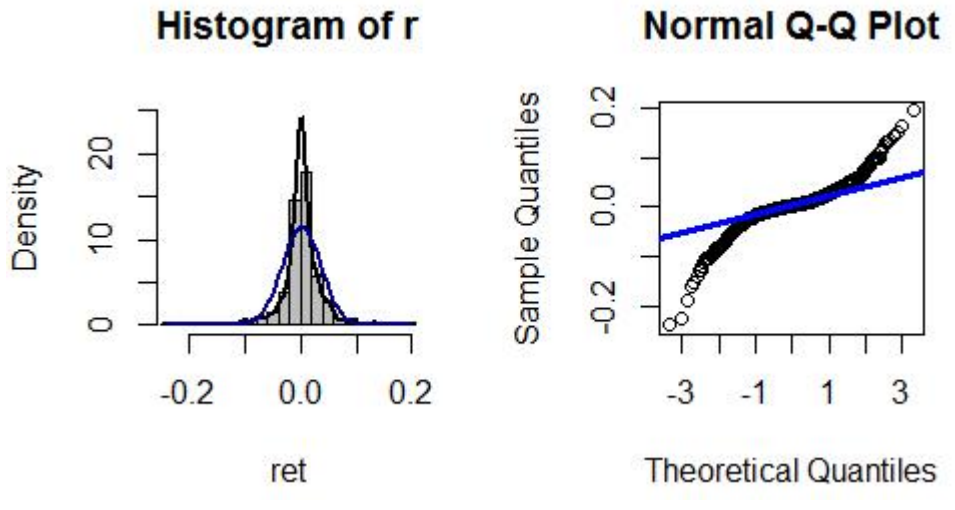


1. Code and Result

```
#*********ARIMA medel**********
xy.acfb(crix$price,numer=FALSE)
adf.test(crix$price)
#Augmented Dickey-Fuller Test:not stationary

##******1)return
r=diff(log(crix$price))
plot(r,type="b")
abline(h = 0)
plot(r,type="l")

#figure4
ts.plot(r)
```

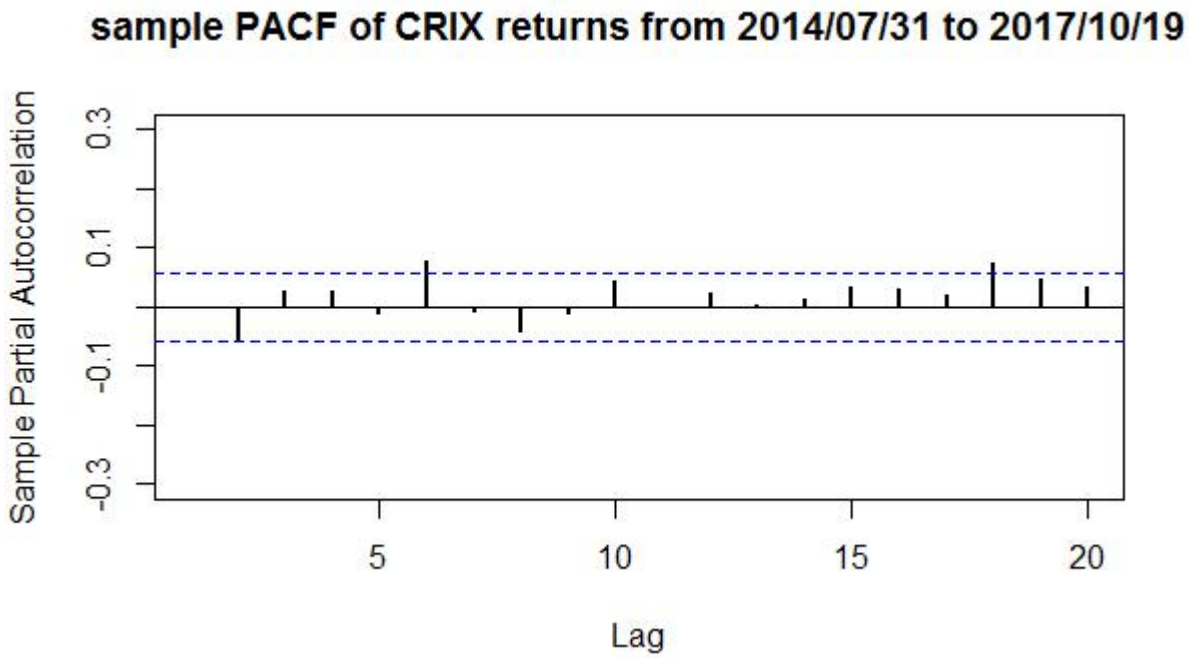



```
#figure5
mean(r)
var(r)
sd(r)
hist(r, col = "grey", breaks = 20, freq = FALSE, ylim = c(0, 25),
xlab = "ret")
lines(density(r), lwd = 2)

mu = mean(r)
sigma = sd(r)
x = seq(-4, 4, length = 100)
curve(dnorm(x, mean = mean(r), sd = sd(r)), add = TRUE,
 col = "darkblue",  lwd = 2)
qqnorm(r)
qqline(r, col = "blue", lwd = 3)
```



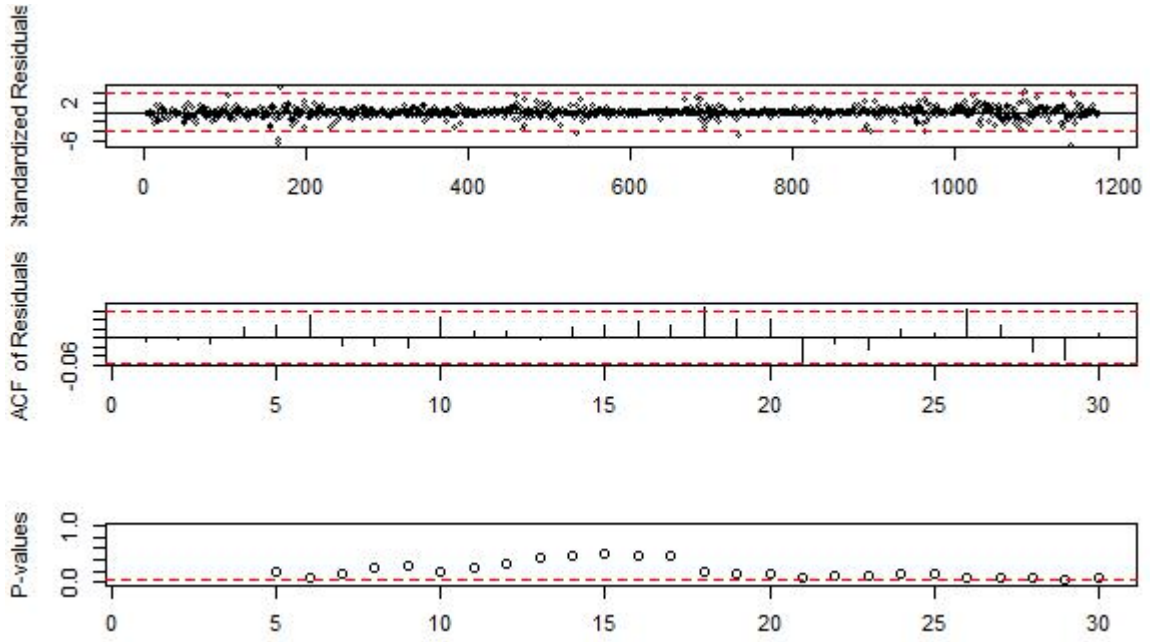
1. Code and Result

```
#figure6
libraries = c("zoo", "tseries")
autocorr = acf(r, lag.max = 20, ylab = "Sample Autocorrelation", main = "sample ACF of CRIX returns from 2014/07/31 to 2017/10/19" ,
          lwd = 2, ylim = c(-0.3, 1))
autopcorr = pacf(r, lag.max = 20, ylab = "Sample Partial Autocorrelation",
           main = "sample PACF of CRIX returns from 2014/07/31 to 2017/10/19" , ylim = c(-0.3, 0.3), lwd = 2)
```



1. Code and Result

```
#figure7
par(mfrow = c(1, 1))
auto.arima(r)
fit202 = arima(r, order = c(2, 0, 2))
tsdiag(fit202)
fit202 = arima(r, order = c(2, 0, 2))
crpre = predict(fit202, n.ahead = 30)

dates = seq(as.Date("31/07/2014", format = "%d/%m/%Y"), by = "days", length = length(ret))

plot(ret, type = "l", ylab = "log return", xlab = "days",
     lwd = 1.5, main = "CRIX returns and predicted values")
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)
```



Thanks for your attention!