厦门大学

报告题目： **Final Exam Project**

姓　　名： 陈　睿 15620161152244

任课老师： Wolf 教授

日　　期： 2017 年 10 月 21 日

HW Unit1

The development of computer memory:

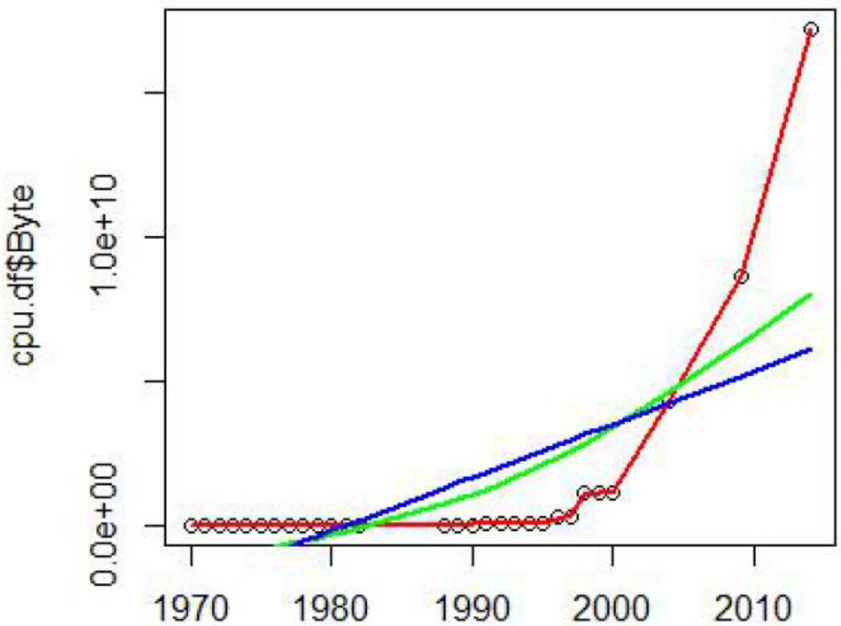
year	Byte
1970	262144
1971	262144
1972	262144
1973	262144
1974	262144
1975	262144
1976	262144
1977	262144
1978	262144
1979	262144
1980	262144
1981	262144
1982	262144
1988	2097152
1989	2097152
1990	2097152
1991	16777216
1992	16777216
1993	16777216
1994	16777216
1995	16777216
1996	268435456
1997	268435456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

HW Unit 2

2.1 Make an R quantlet to solve HW #1 from unit 1 with R and show it on Github (GH). Hint: use the CMB Qs for this work.

```
#hw2

# EX1&EX2

cpu.df = read.csv("byte.csv",header = TRUE)

plot(cpu.df$Byte~cpu.df$year,title(main = "The development of computer memory",cex.main= 0.8))
```

```
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
```



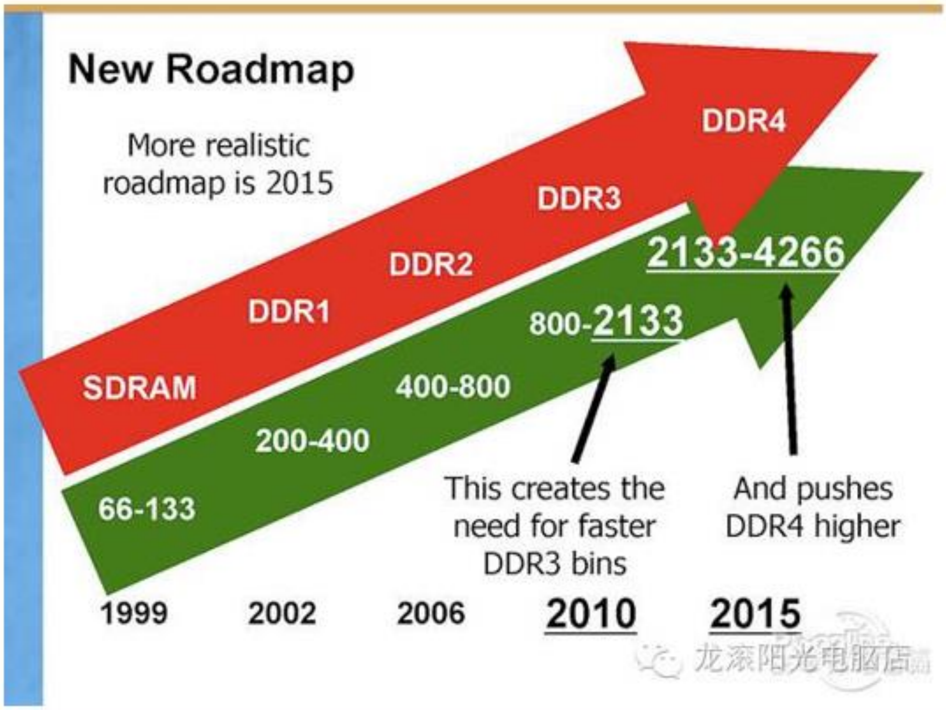
2.2 Use R with B-spline code to solve HW#1, any comments?

```
#the code of question 2
data<-read.table("C:/Users/Administrator/Desktop/workstation/hw1.csv",header=TRUE,sep="")
x<-data$Year
y<-data$RAM
plot(x,y)
lines(spline(x,y))
lines(spline(x, y, n = 201), col = 2)
```

2.3 logistic regression ppt


Logistics regression model
Rui CHEN
2017/10/21
1

The development of home computer inter storage



Generalized Linear Models

- First of all, we briefly review the concept of generalized linear models (GLMs). Logistic regression is just one example of this type of model.

- All generalized linear models have the following three characteristics:

1. A probability distribution describing the outcome variable;
2. A linear model: $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$;
3. A link function that relates the linear model to the parameter of the outcome distribution:

$$g(p) = y \; or \; p = g^{-1}(y)$$

Logistic Regression

- Logistic regression is a GLM used to model a binary categorical variable using numerical and predictors.

- We assume a binomial distribution produced the outcome variable and we therefore want to model *p* the probability of success for a given set of predictors.

- To finish specifying the logistic model we just need to establish a reasonable link function that connects *y* to *p.* There are a variety of options but the most commonly used is the logit function.

$$logit(p) = log\left(\frac{p}{1-p}\right), \qquad for\ 0 \le p \le 1$$

Properties of the Logit

- The logit function takes a value between 0 and 1, mapping it to a value between $-\infty$ and ∞.

- Inverse logit (logistic) function

$$g^{-1}(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

- The inverse logit function takes a value between $-\infty$ and ∞ and maps it to a value between 0 and 1.

- This formulation also has some use when it comes to interpreting the model as logit can be interpreted as the log odds for a success.

The logistic regression model

- The three GLM characteristics give us:
- 1. $y_i \sim Binom(p_i)$

  2. $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$

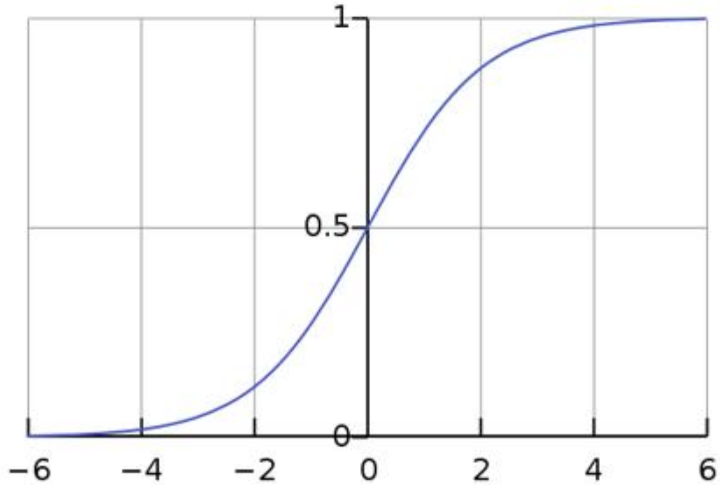
  3. $logit(p) = y$

- From which we arrive at,

$$p_i = P(y = 1|\boldsymbol{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_{1,t} + \cdots + \beta_n x_{n,t})}},$$

$$where\ \boldsymbol{x} = (x_{1,t}, x_{2,t}, \ldots, x_{n,t})$$

Graph



2.4 Suppose you observe that in n=1000 mails (in 1 week) you have about 2 scams. Use the LvB /Possion pdf to calculate that you have 6 scam emails in 2 weeks. In Scammyland you have 5 scams on average, what is the probability to have no scam mail.

```
# EX3

lambda=2

x=3

probex1=exp(-lambda)*lambda^x/factorial(x)

probex1


lambda=5

x=0

probex2=exp(-lambda)*lambda^x/factorial(x)

probex2
```

```
> 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
```

HW Unit 3

3.1 Make an R quantlet on GH to produce hash code for the 2 sentences: „I learn a lot from this class when I am proper listening to the professor“, „I do not learn a lot from this class when I am absent and playing on my Iphone“. Compare the 2 hash sequences.

install stuff for hash calculation

install.packages("digest")

call the library doing the hashes

library("digest")

digest("I learn a lot from this class when I am proper listening to the professor")

#"a8d3e4701672195e5dcd16ea9b062279"

digest("I do not learn a lot from this class when I am absent and playing on my phone")

#"059ab10d478614d2eab3d70cfccd3fcc"

digest("I learn a lot from this class when I am proper listening to the professor","sha256")

#"c16700de5a5c1961e279135f2be7dcf9c187cb6b21ac8032308c715e1ce9964c"

digest("I do not learn a lot from this class when I am absent and playing on my phone","sha256")

#"f5e2cba48dac097355d0bb310fdbd5bd38a22a5c8e8215cd1ae67014cfc35b91"

3.2 Make 3-5 slides (in PPTX) on the DSA (Digital Signature Algorithms)

Q2： DSA
(Digital Signature Algorithms)

Presented by Rui Chen 15620161152244

Department of Finance, SOE

Definition of DSA

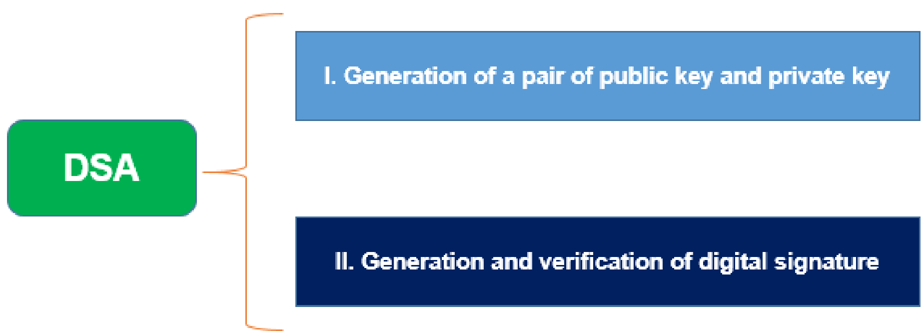
The Digital Signature Algorithm (DSA) is a Federal Information Processing Standard for digital signatures. In August 1991 the National Institute of Standards and Technology (NIST) proposed DSA for use in their Digital Signature Standard (DSS) and adopted it as FIPS 186 in 1993. Four revisions to the initial specification have been released: FIPS 186-1 in 1996, FIPS 186-2 in 2000, FIPS 186-3 in 2009, and FIPS 186-4 in 2013.

Definition of DSA

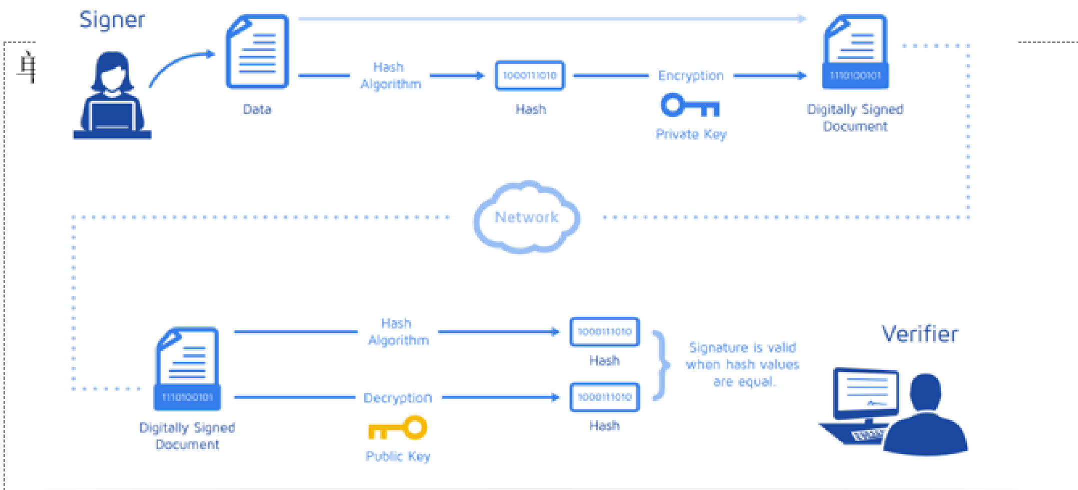
DSA is covered by U.S. Patent 5,231,668, filed July 26, 1991 and attributed to David W. Kravitz, a former NSA employee. This patent was given to "The United States of America as represented by the Secretary of Commerce, Washington, D.C.", and NIST has made this patent available worldwide royalty-free. Claus P. Schnorr claims that his U.S. Patent 4,995,082 (expired) covered DSA; this claim is disputed. DSA is a variant of the ElGamal signature scheme.

——From Wikipedia

DSA consists of the following two parts：



How do digital signatures work?



Key generation has two phases：

Two phases

- **I. A choice of algorithm parameters which may be shared between different users of the system**
- **II. Computing public and private keys for a single user**

The steps of performing the digital signature

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.

Reference：

https://en.wikipedia.org/wiki/Digital_Signature_Algorithm

3.3 Make slides with R code where you create a JSON data set that you save and read again.

```
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)
```

```
load("H:/大数据与互联网金融/crix.RData")
```

3.4 Download the CRIX data and make a plot of the time series, analyse its properties, i.e. fit ARMA, ARIMA etc. Is there a GARCH effect?

crix_data_frame = as.data.frame(json_data)

x=crix_data_frame

n=dim(x)

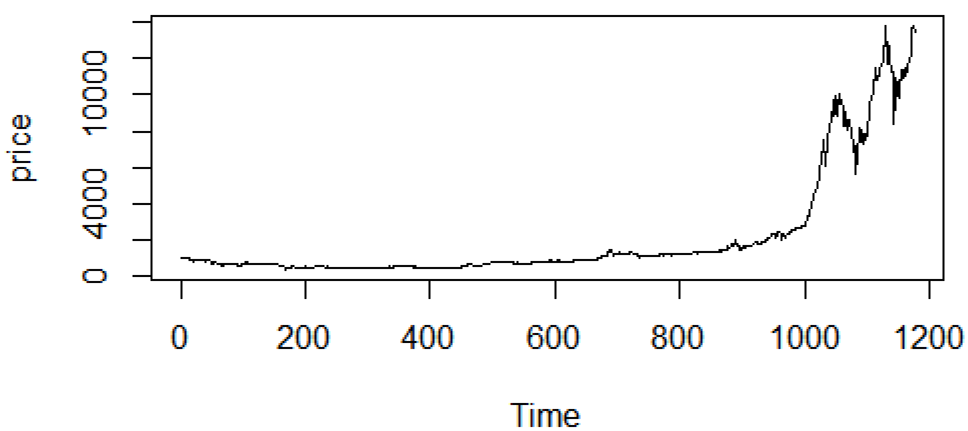
a=seq(1,n[2],2)

b=seq(2,n[2],2)

date=t(x[1,a])

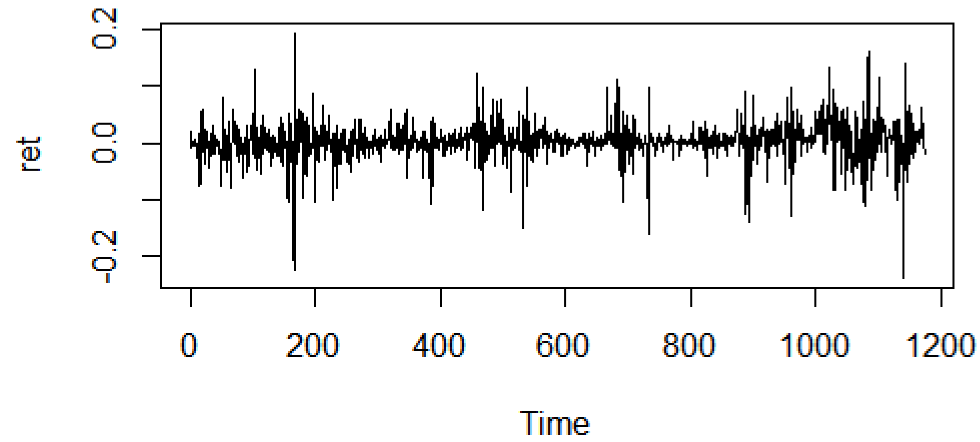
price=t(x[1,b])

ts.plot(price)

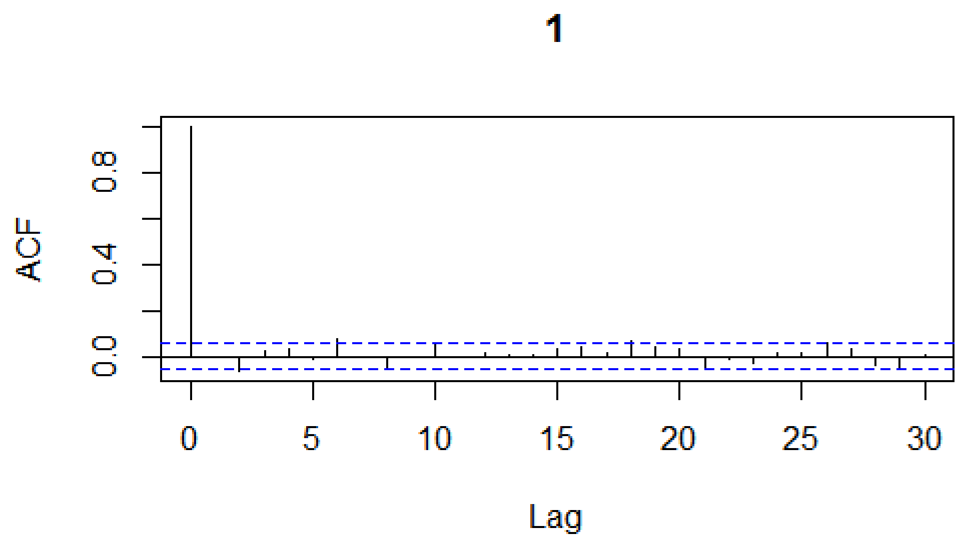


ret=diff(log(price))

ts.plot(ret)



acf(ret)



#auto.arima(ret)

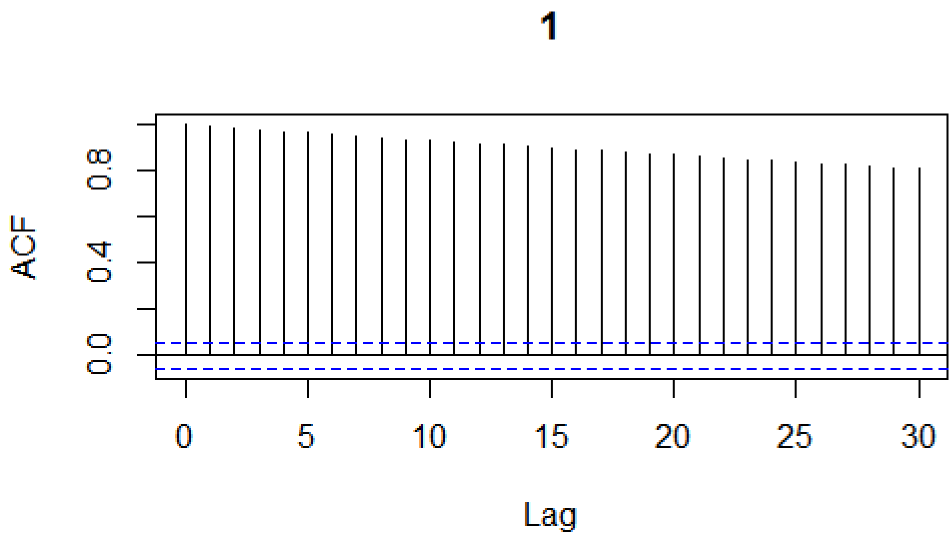
fit1 = arima(ret, order=c(1,0,1))

tsdiag(fit1)

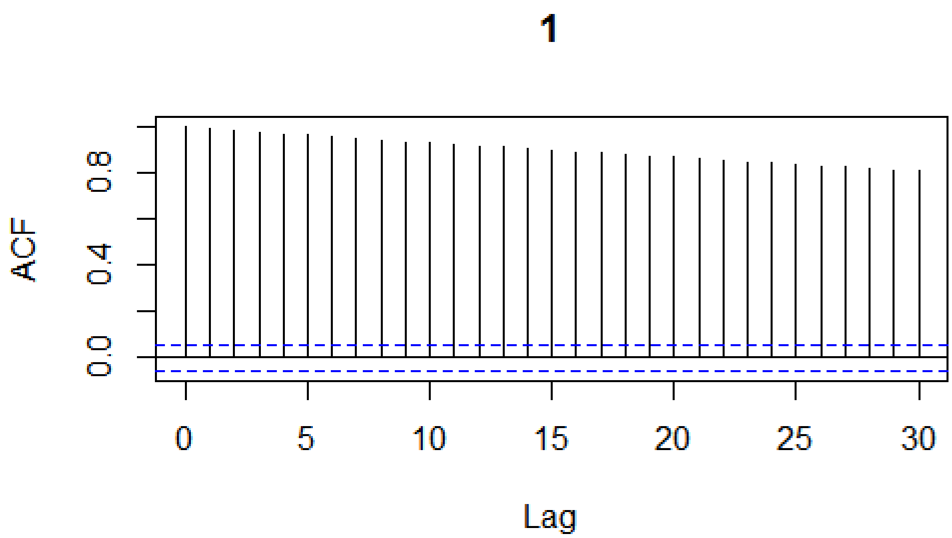
Box.test(fit1$residuals,lag=1)

```
#Box-Pierce test

data:  fit1$residuals
X-squared = 5.7598e-07, df = 1, p-value = 0.9994, not reject  H0: price is nonstationary #
```

acf(price)



pacf(price)



install.packages("forecast")

library(forecast)

```
#arima model
par(mfrow=c(1,1))
#auto.arima(ret)
fit1 = arima(ret, order=c(1,0,1))
tsdiag(fit1)
Box.test(fit1$residuals,lag=1)
#*****3)Parameter Estimation
#estimation of p and q
a.fin1=auto.arima(dr)
summary(a.fin1)
#ARMA(0,0) therefore r fits ARIMA(0,1,0)
a.fin2=arima(r,order=c(0,1,0))
summary(a.fin2)
f=forecast(a.fin2,h=3,level=c(99.5))
acf(f$residuals,lag.max = 20)
Box.test(f$residuals,lag=20,type='Ljung-Box')
#the residuals follow Gaussian distribution
plot.ts(f$residuals)
#aic
aic=matrix(NA,6,6)
for(p in 0:5)
{
  for(q in 0:5)
  {
    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

```
         [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] -4542.536 -4535.468 -4532.309 -4525.945 -4520.869 -4513.908
[2,] -4535.468 -4528.400 -4525.535 -4523.076 -4516.134 -4509.428
[3,] -4532.553 -4525.909 -4524.000 -4512.366 -4511.991 -4505.668
[4,] -4526.383 -4523.684 -4512.524 -4513.948 -4507.626 -4498.932
[5,] -4520.178 -4513.111 -4511.576 -4507.557 -4503.693 -4496.750
[6,] -4513.197 -4509.276 -4506.483 -4500.723 -4496.761 -4491.485
```

```
#bic
bic=matrix(NA,6,6)
for(p in 0:5)
{
  for(q in 0:5)
  {
    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)
```

```
Box-Pierce test

data:  fit4$residuals
X-squared = 0.00046506, df = 1, p-value = 0.9828
```

```
fitr4 = arima(ret, order=c(2,1,3))
tsdiag(fitr4)
Box.test(fitr4$residuals,lag=1)
```

```
Box-Pierce test
```

```
data:  fitr4$residuals
X-squared = 0.26615, df = 1, p-value = 0.6059
```

#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)))

```
[1] -4524
```

AIC(fit4, k=log(length(ret)))

```
[1] -4512.366
```

AIC(fitr4, k=log(length(ret)))

```
[1] -4519.232
```

fit202$aic

```
[1] -4554.409
```

fit4$aic

```
[1] -4547.843
```

fitr4$aic

```
[1] -4549.641
```

#arima202 predict

fit202 = arima(ret,order=c(2,0,2))

crpre = predict(fit202, n.ahead = 30)

tsret = ts(ret)

plot(retts$Dare, retts$ret, type="o", xlim=c(0,644))

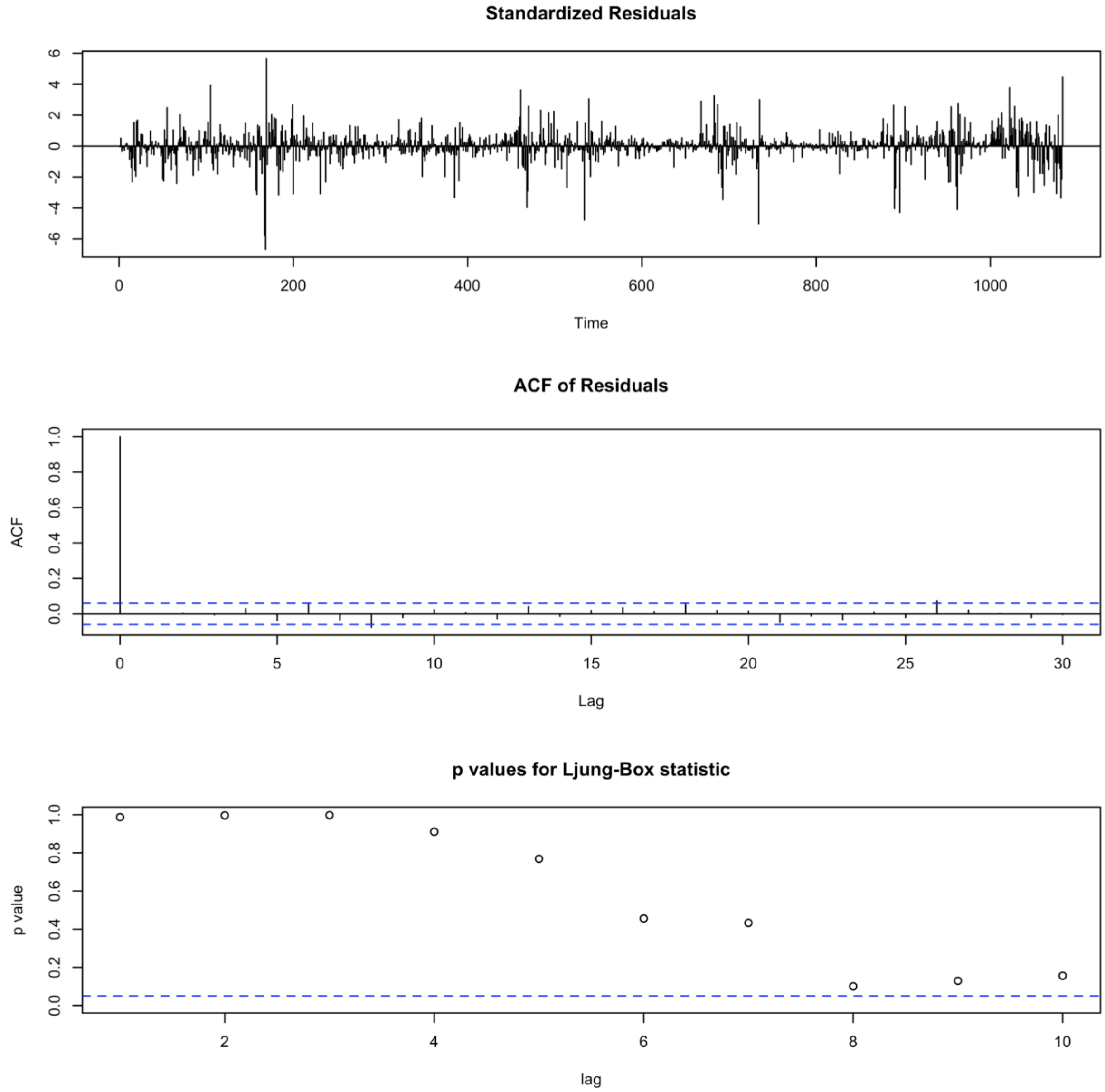
lines(retts$ret)

lines(crpre$pred,col="red",lwd=3)

lines(crpre$pred+2*crpre$se,col="red",lty=3, lwd=3)

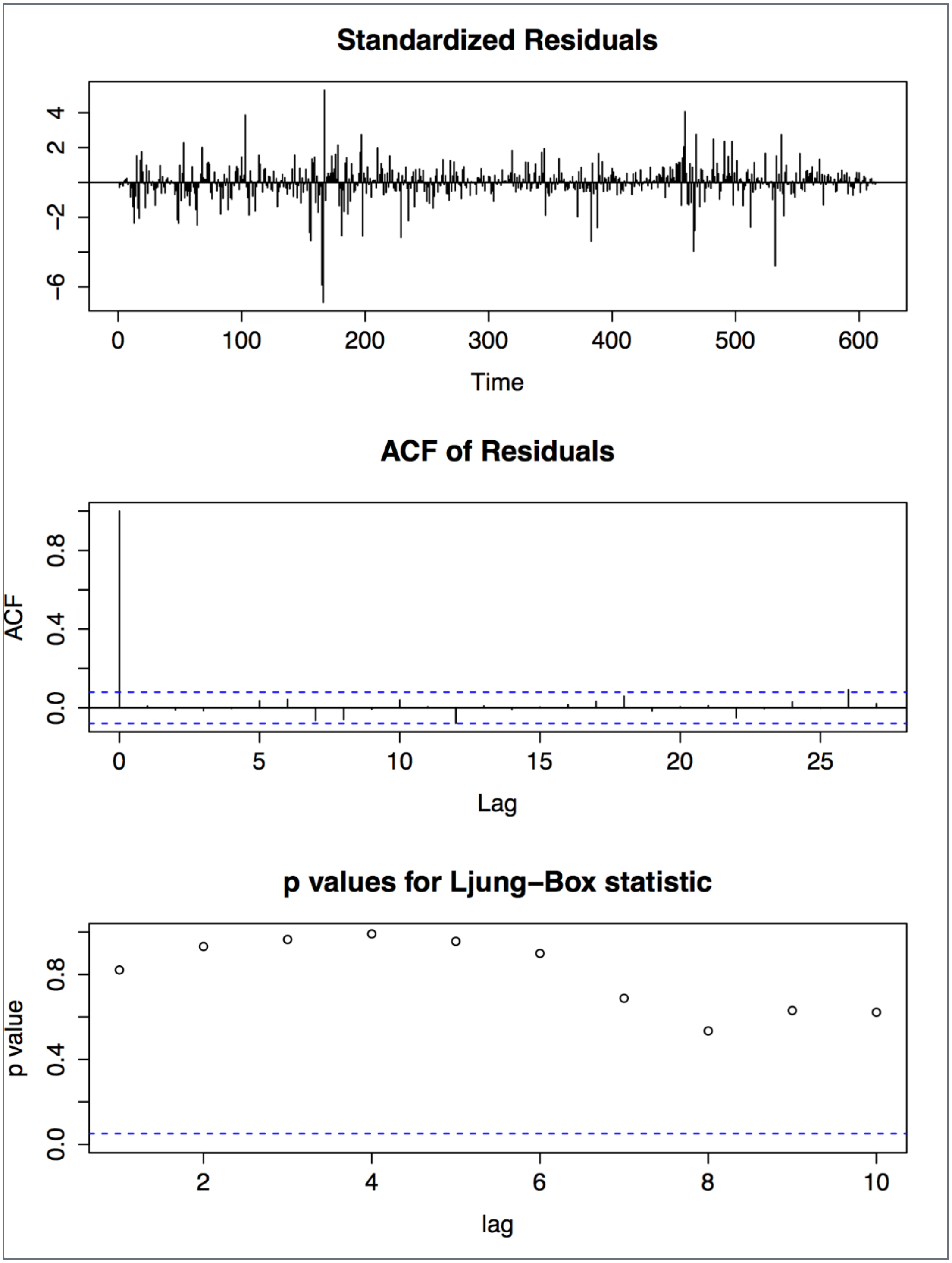
lines(crpre$pred-2*crpre$se,col="red",lty=3,lwd=3)

ARIMA(1，0，1)



Standardized Residuals
6
4
2
0
-2
-4
-6
0
200
400
600
800
1000
Time
ACF of Residuals
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
Lag
p values for Ljung-Box statistic
p value
1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
10
lag

ARIMA(2，0，2)



AS A WHOLE:

#3&4

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')




library(caschrono)
```

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

#******************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))*100
plot(r,type="b")
abline(h = 0)
plot(r,type="l")

#*****2)Model Specification ARIMA(p,d,q)
#ADF test-H0:unit root H1:no unit root(test for stationarity)
adf.test(r)
#p-value=0.27,not stationary.
dr=diff(r)
plot(dr,type="b")
abline(h = 0)
adf.test(dr)
#p-value=0.01,stationary.(d=1)

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

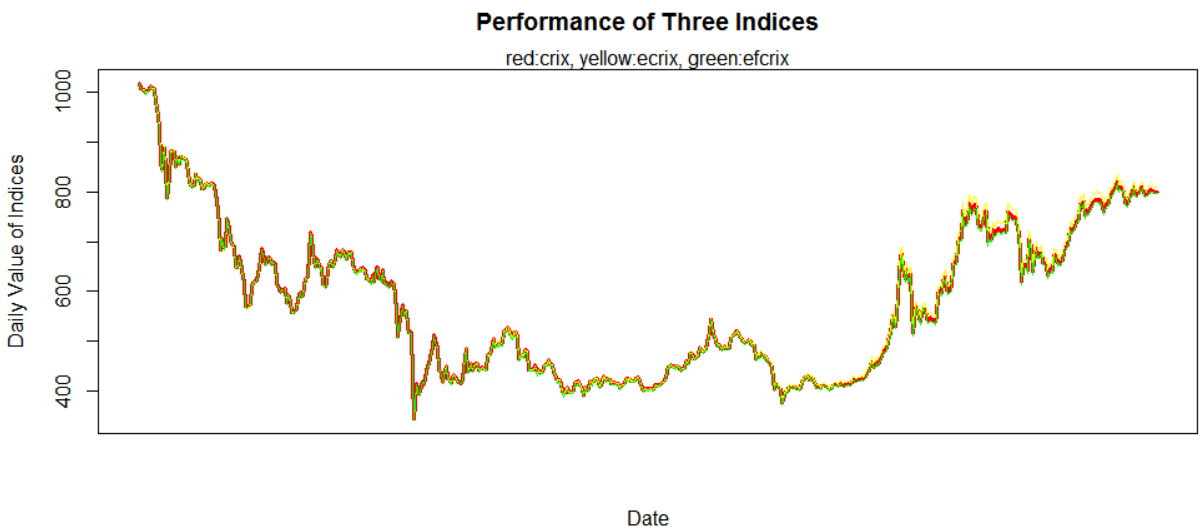
```
a.fin1=auto.arima(dr)

summary(a.fin1)

#ARMA(0,0) therefore r fits ARIMA(0,1,0)

a.fin2=arima(r,order=c(0,1,0))

summary(a.fin2)

help("forecast.Arima")

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)


#****4)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
```

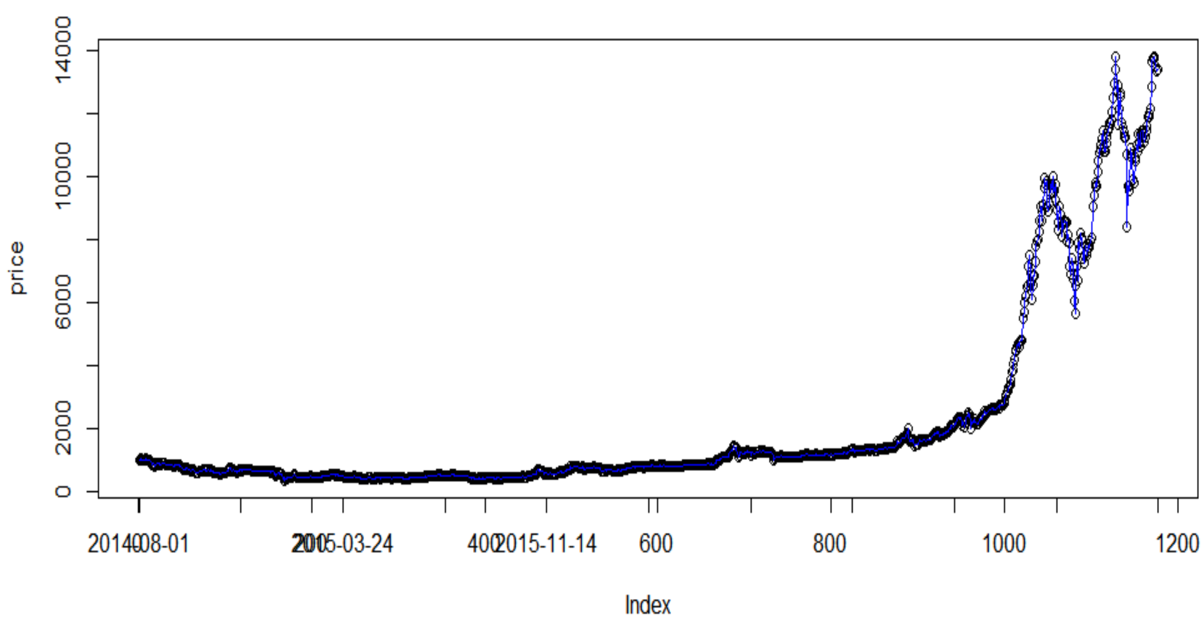
```
g1=garchFit(~garch(1,1),data=residuals(a.fin2),trace=FALSE,include.mean=TRUE, na.action=na.pass)

summary(g1)

g2=garchFit(~garch(1,2),data=residuals(a.fin2),trace=FALSE,include.mean=TRUE, na.action=na.pass)

summary(g2)

g3=garchFit(~garch(2,1),data=residuals(a.fin2),trace=FALSE,include.mean=TRUE, na.action=na.pass)

summary(g3)

g4=garchFit(~garch(2,2),data=residuals(a.fin2),trace=FALSE,include.mean=TRUE, na.action=na.pass)

summary(g4)

#The best one is Garch(1,1) model which has the smallest AIC.
```

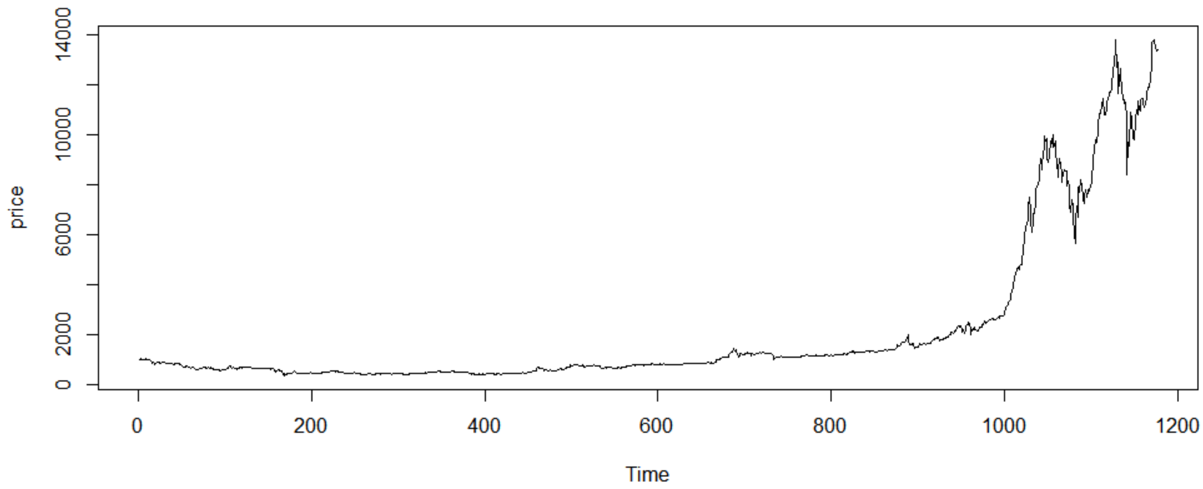
Unit 4 HW

```
#figure 3:crix&ecrix%efcrix

setwd("C:/Users/Administrator/Desktop/workstation")

load("crix.RData")

load("ecrix.RData")

load("efcrix.RData")

plot(crix, type = "l", col = "red", xaxt = "n", lwd = 3, main = "Performance of Three Indices",  xlab =
"Date", ylab = "Daily Value of Indices")

lines(ecrix, col = "yellow")

lines(efcrix, col = "green")

mtext("red:crix, yellow:ecrix, green:efcrix")

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)

x=crix_data_frame

dim(x)

n=dim(x)

a=seq(1,n[2],2)
```

```
b=seq(2,n[2],2)
data=t(x[1,a])
price=t(x[1,b])
ts.plot(price)
plot(price)
lines(price, col = "blue")
s=seq(1,n[2],n[2]/20)
axis(1, at = s, label = names(ecrix)[s])
```

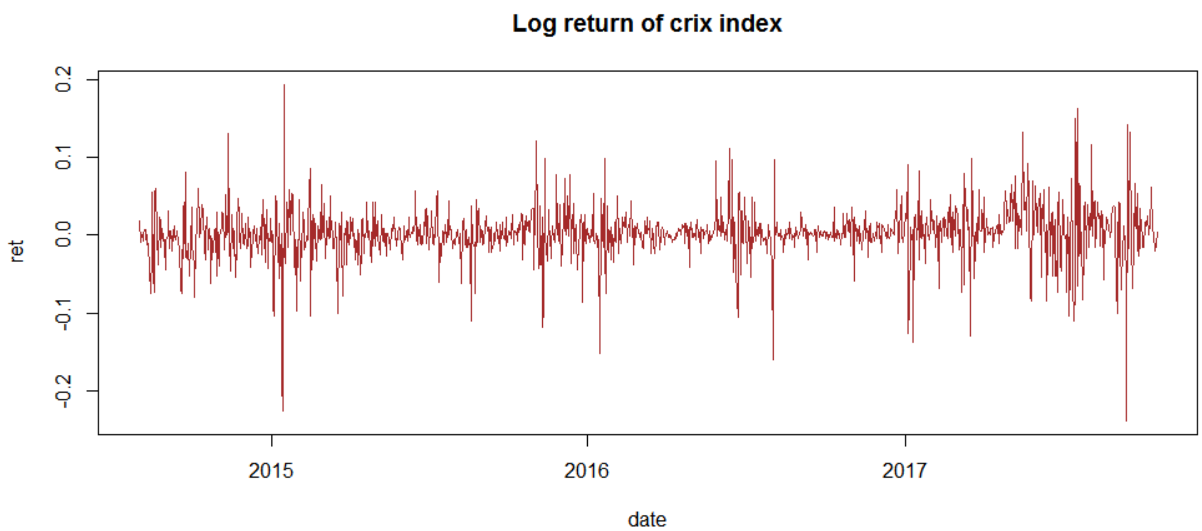

Performance of Three Indices
red:crix, yellow:ecrix, green:efcrix
Daily Value of Indices
1000
800
600
400
Date


price
14000
10000
6000
2000
0
2014-08-01
2015-03-24
2015-11-14
200
400
600
800
1000
1200
Index



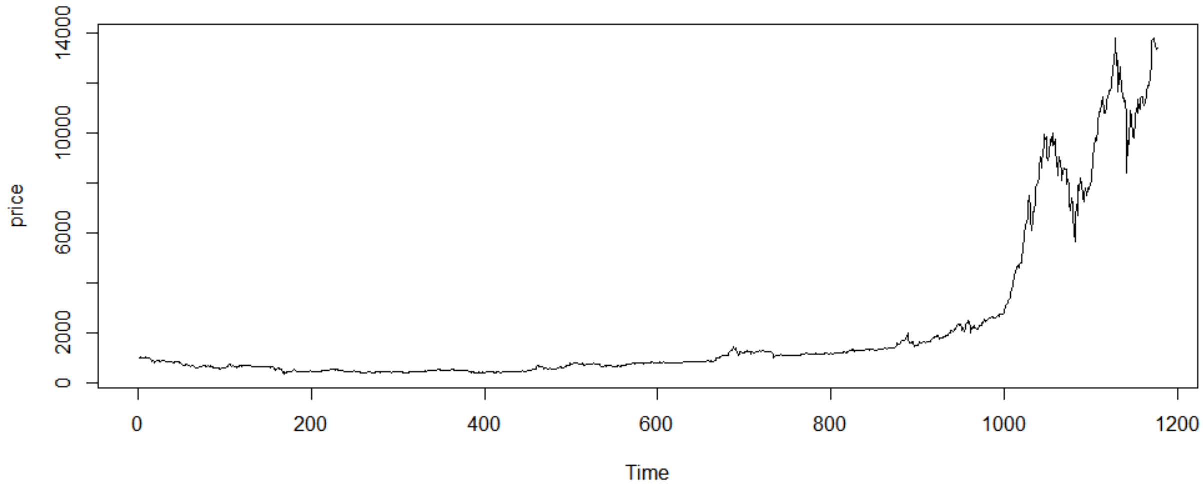

```
#figure4
library(rjson)
json_file = "http://crix.hu-berlin.de/data/crix.json"
json_data = fromJSON(file=json_file)
x = as.data.frame(json_data)



date1=c(json_data[[1]]$date)

for (i in 1:2348){

  date1[i]=c(json_data[[i]]$date)
}

price1=c(json_data[[1]]$price)
for (i in 1:2348){

  price1[i]=c(json_data[[i]]$price)
}
```

```
date=date1

price=price1

crix=data.frame(date,price)

date2=date[-1]

ret=diff(log(price))

plot(ret~as.Date(date2),type="l",col="brown",xlab="date",ylab="ret", main="Log return of crix index")
```



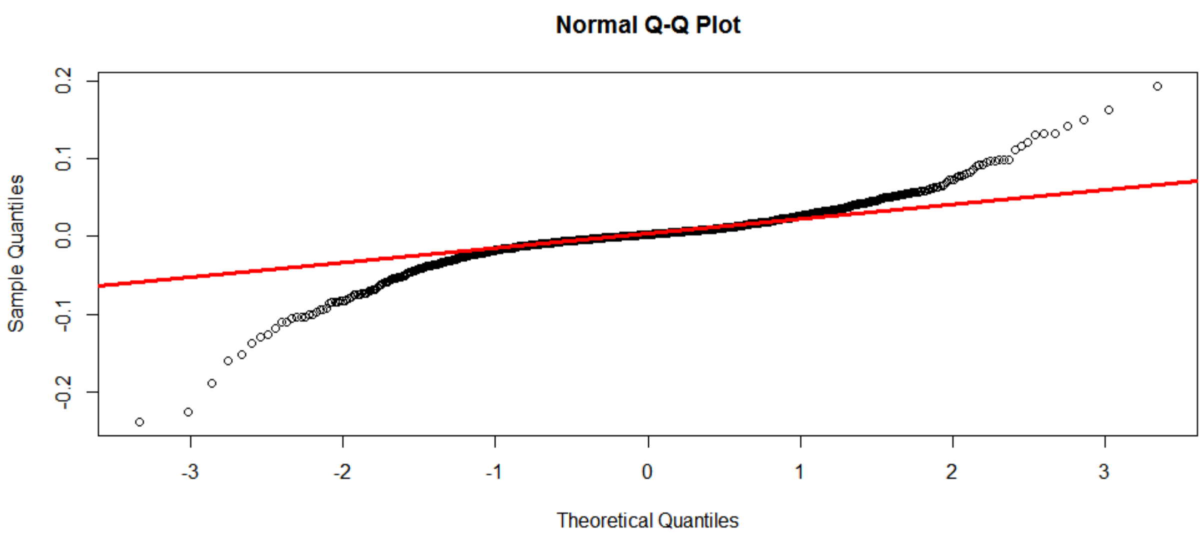

```
#figure5

mean(ret)

var(ret)

sd(ret)

hist(ret, col = "orange", breaks = 20, freq = FALSE, ylim = c(0, 25), xlab = "ret")

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 = "purple", lwd = 2)



qqnorm(ret)

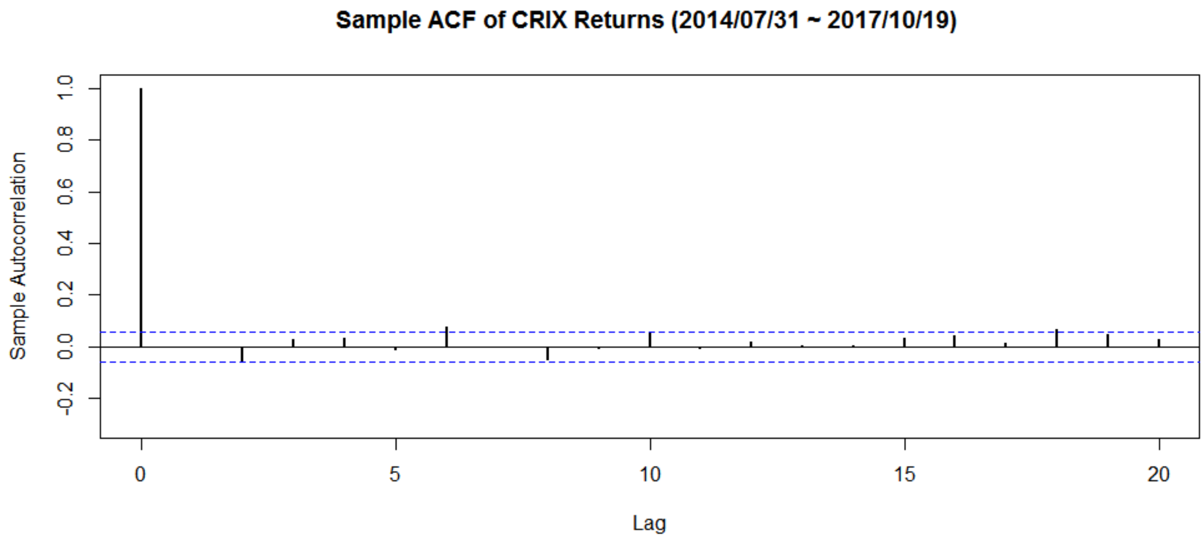
qqline(ret, col = "red", lwd = 3)



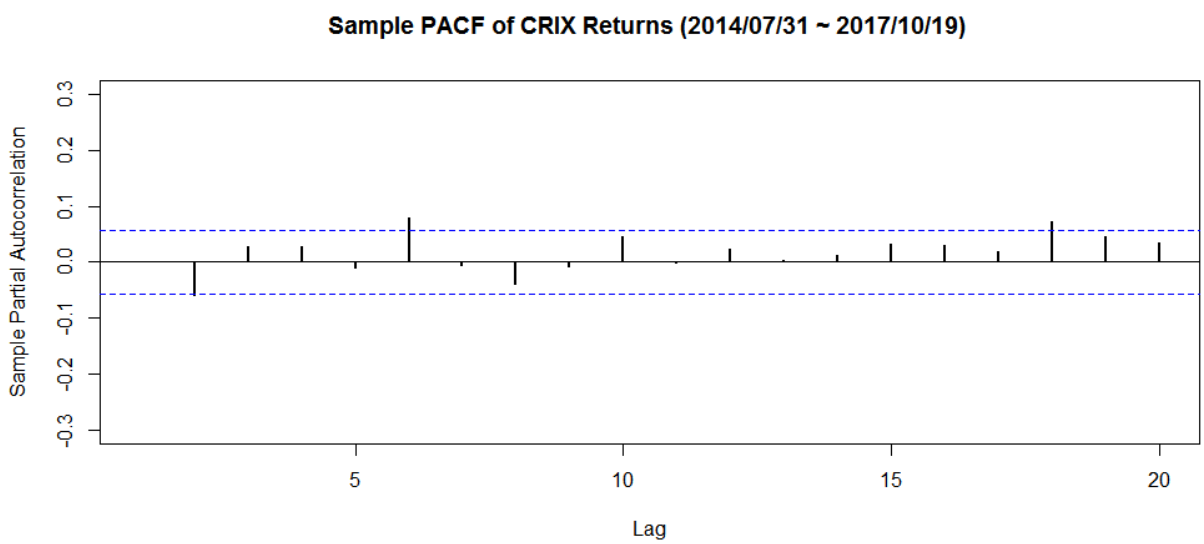
#figure6

libraries = c("zoo", "tseries")

autocorr = acf(ret, lag.max = 20, ylab = "Sample Autocorrelation", main = "Sample ACF of CRIX Returns (2014/07/31 ~ 2017/10/19) ", lwd = 2, ylim = c(-0.3, 1))



autopcorr = pacf(ret, lag.max = 20, ylab = "Sample Partial Autocorrelation", main = "Sample PACF of CRIX Returns (2014/07/31 ~ 2017/10/19) ", ylim = c(-0.3, 0.3), lwd = 2)



#figure7

arima model

library(caschrono)

library(TTR)

library(forecast)

library(TSA)

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

auto.arima(ret)

fit202 = arima(ret, order = c(2, 0, 2))

tsdiag(fit202)

fit202 = arima(ret, 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 = "blue", lty = 3, lwd = 3)

lines(crpre$pred - 2 * crpre$se, col = "grey", lty = 3, lwd = 3)
```