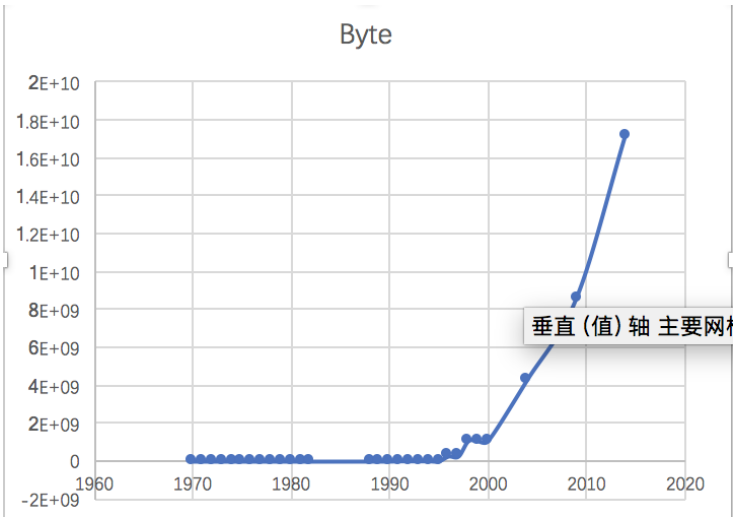
Homework 1

1.

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



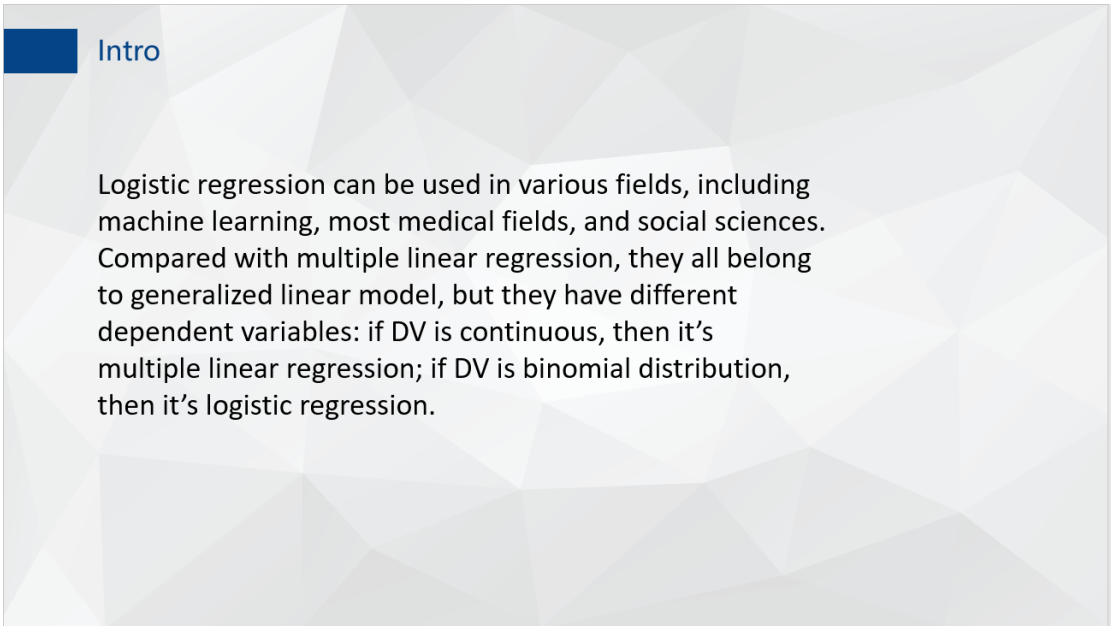
2. logistic regression

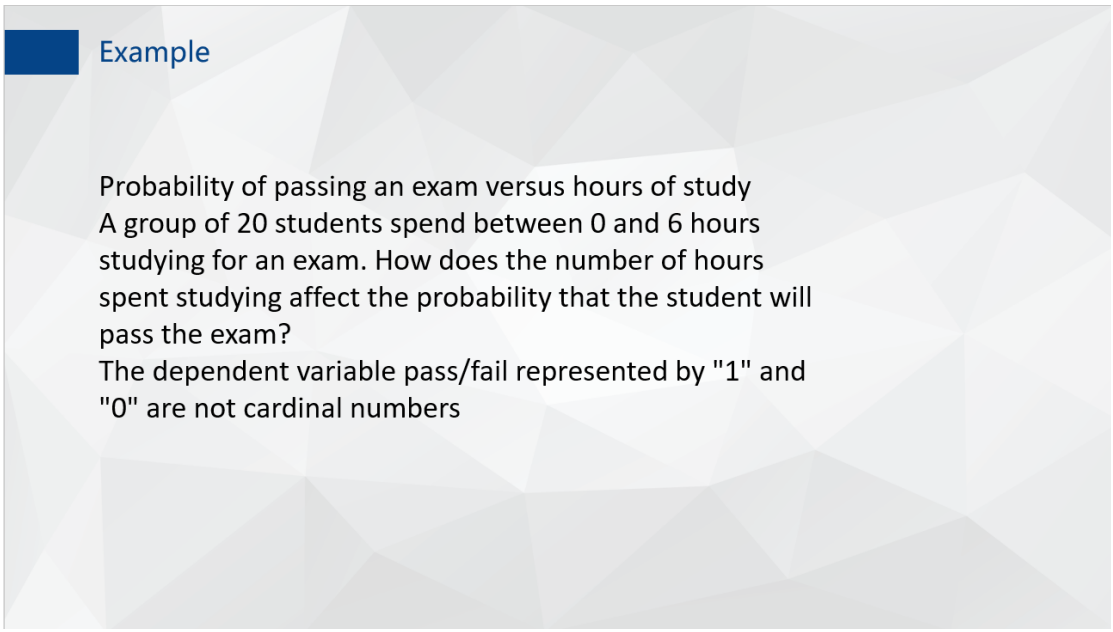
HW1 Logistic Regression

Presented by Xiaomeng Rao
ID Numb 15620161152244
Department of Finance, SOE

Intro

A regression model where the dependent variable is categorical.
Logistic regression was developed by statistician David Cox in 1958.
The binary logistic model is used to estimate the probability of a binary response based on one or more predictor (or independent) variables (features). It allows one to say that the presence of a risk factor increases the odds of a given outcome by a specific factor



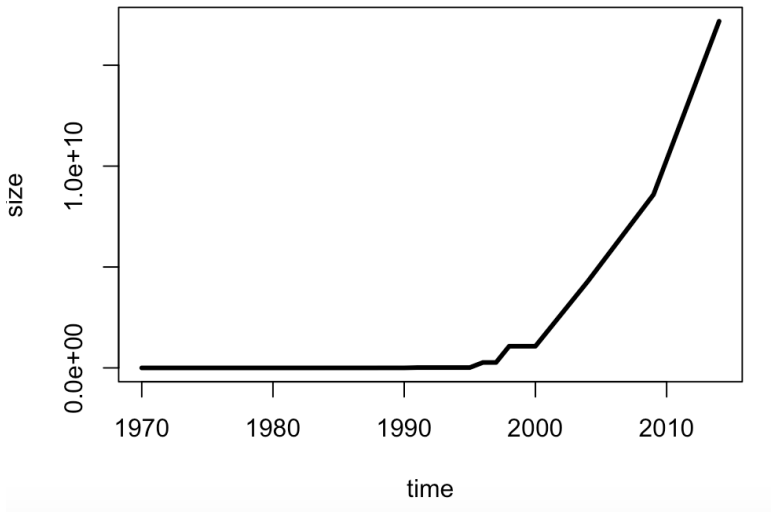


3. **Github Account:** https://github.com/Simon2274/Home-Work-for-BDIF

Homework 2

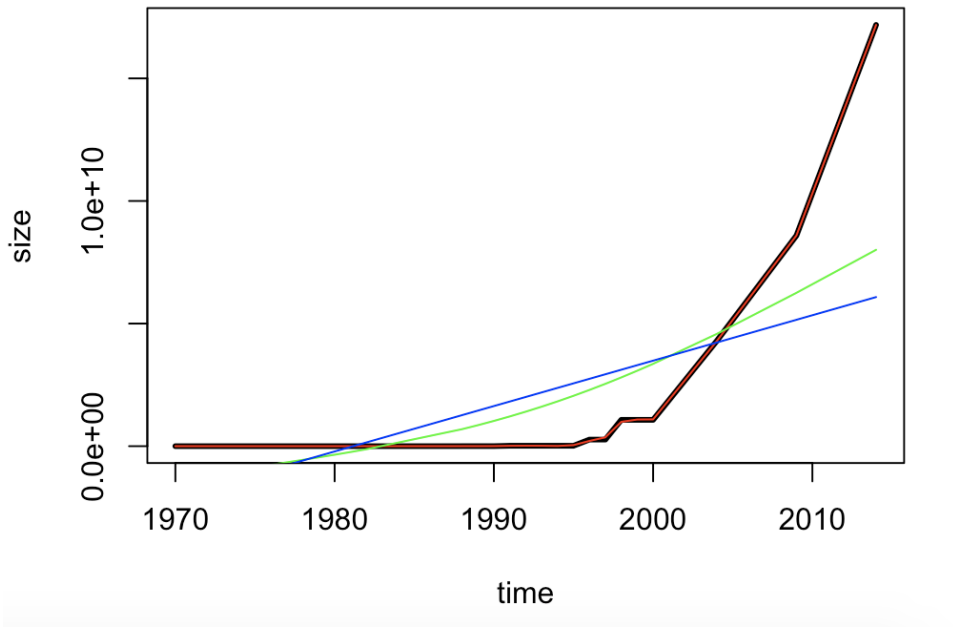
1. Use R to solve HW #1

```
library(readr)
RAM_size <- read_csv("~/R data/Home-Work-for-BDIF/RAM_size.csv")
plot(RAM_size,type="l",xlab = "time",ylab = "size",lwd=3)
```



2. use R with B-spline code to solve HW#1

```
splines.reg.l1 = smooth.spline(x = RAM_size$year, y = RAM_size$Byte, spar = 0.2)  # lambda = 0.2
splines.reg.l2 = smooth.spline(x = RAM_size$year, y = RAM_size$Byte, spar = 1)  # lambda = 1
splines.reg.l3 = smooth.spline(x = RAM_size$year, y = RAM_size$Byte, spar = 2)  # lambda = 2
lines(splines.reg.l1, col = "red", lwd = 1)  # regression line with lambda = 0.2
lines(splines.reg.l2, col = "green", lwd = 1)  # regression line with lambda = 1
lines(splines.reg.l3, col = "blue", lwd = 1)  # regression line with lambda = 2
```



Comments: The larger the spar is, more smooth the line is.

3. Poisson Distribution

lambda=4
x=6
dpois(x,lambda)

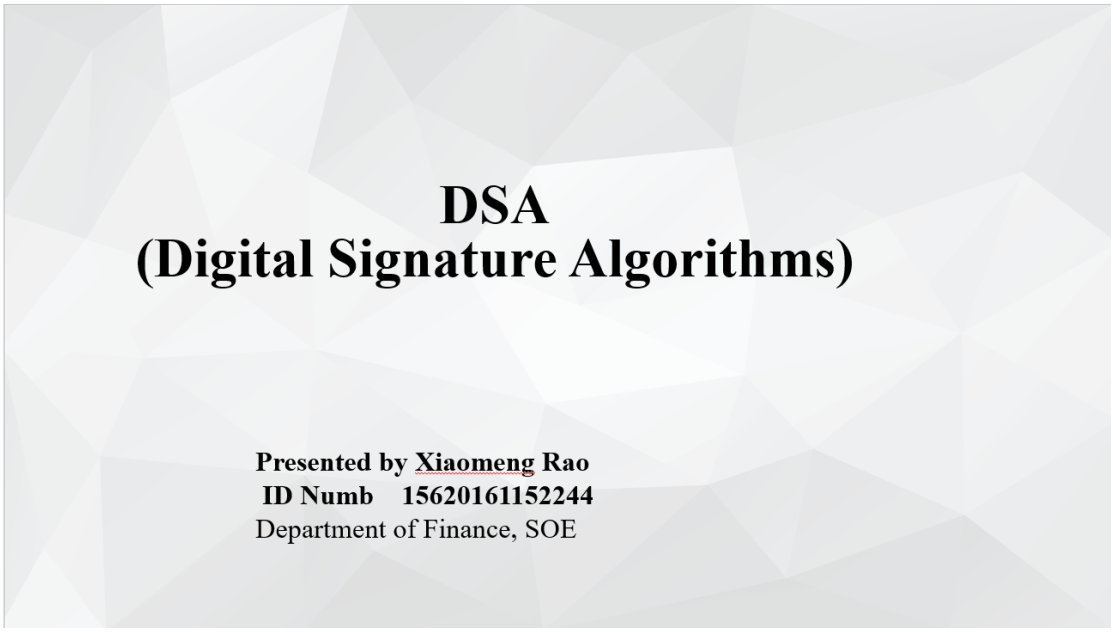
lambda=5
x=0
dpois(x,lambda)

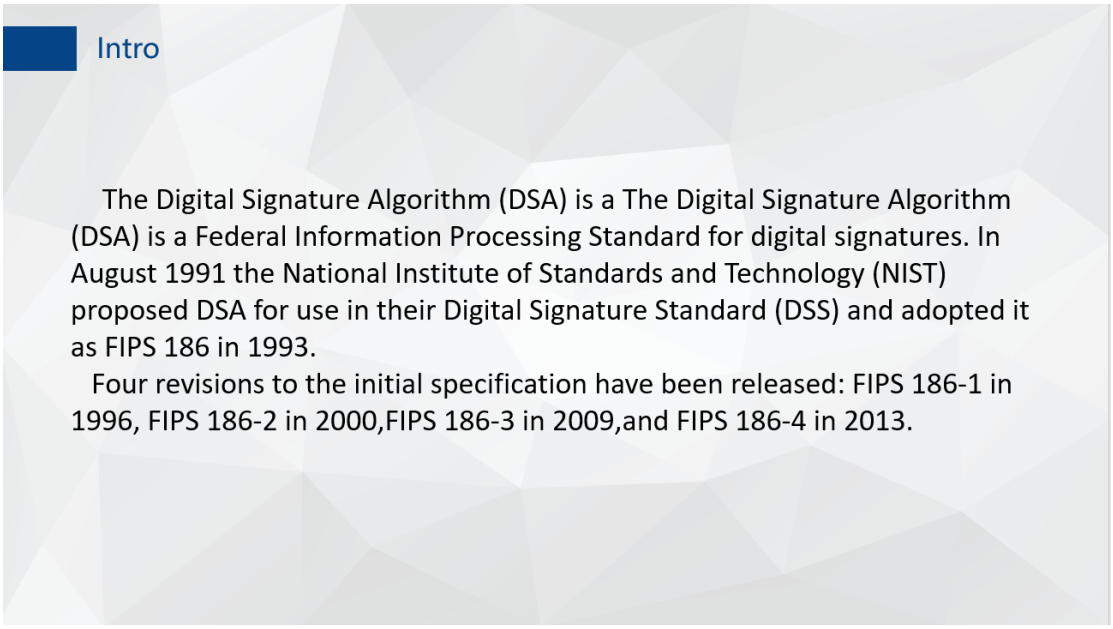
Homework 3

1. hash code

#install.packages("digest",repos='http://cran.us.r-projest.org')
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")

2. Digital Signature Algorithms





Intro

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.[8] DSA is a variant of the ElGamal signature scheme.

Key generation

Key generation has two phases. The first phase is a choice of *algorithm parameters* which may be shared between different users of the system, while the second phase computes public and private keys for a single user.

Peculiarity

Creates a 320 bit signature

With 512-1024 bit security

Smaller and faster than RSA

A digital signature scheme only

Security depends on difficulty of computing discrete logarithm

Variant of ElGamal&Schnorr schemes

3. Json data

JSON

- Dataframe in R

	year	Byte
1	1970	262144
2	1971	262144
3	1972	262144
4	1973	262144
5	1974	262144
6	1975	262144
7	1976	262144
8	1977	262144
9	1978	262144
10	1979	262144
11	1980	262144
12	1981	262144
13	1982	262144
14	1988	2097152

Json

- library(rjson)
- json_RAM <- toJSON(RAM_size,method = "C")

```
> json_RAM
[1] "{\"year\":[1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1988,1
989,1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2000,2004,2009,2014],\"Byte\":[26
2144,262144,262144,262144,262144,262144,262144,262144,262144,262144,262144,262144,26214
4,2097152,2097152,2097152,16777216,16777216,16777216,16777216,16777216,268435456,268435
456,1073741824,1073741824,1073741824,4294967296,8589934592,17179869184]}"
```

4. CRIX data

```
#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)
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)

#install.packages("forecast")
#install.packages("tseries")
library(forecast)
library(tseries)
ts.plot(crix_data_frame$price)
Acf(crix_data_frame$price)

for(i in 1:length(crix_data_frame$price)){
   crixreturn[i] <- log(crix_data_frame$price[i+1]/crix_data_frame$price[i])
}
```

ts.plot(crixreturn)

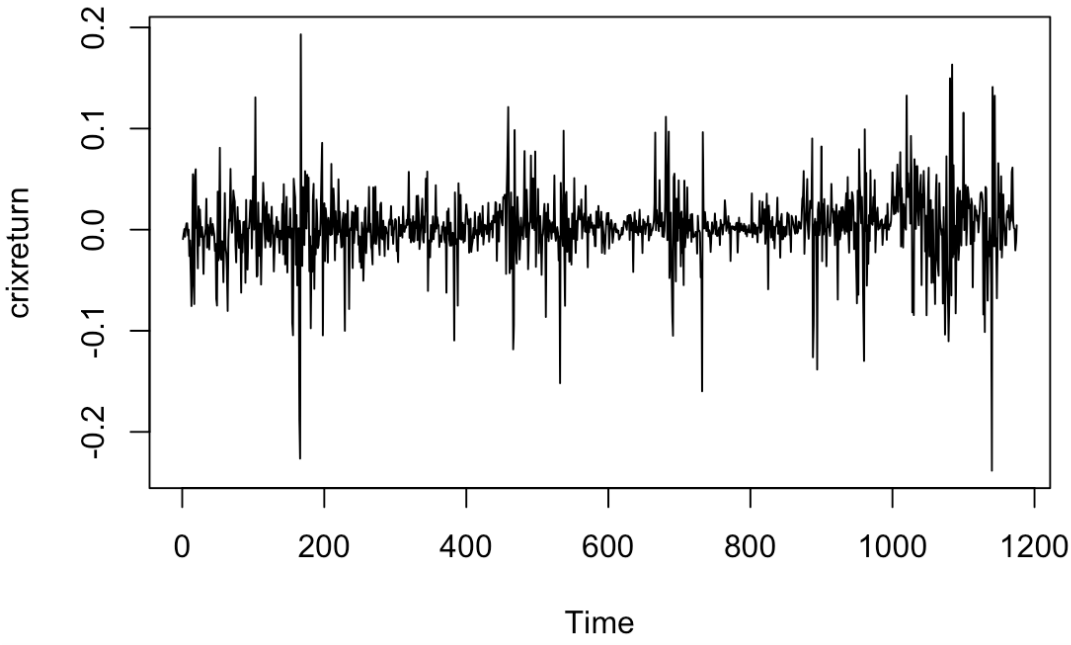
Box.test(crixreturn, type = "Ljung-Box", lag = 20)

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

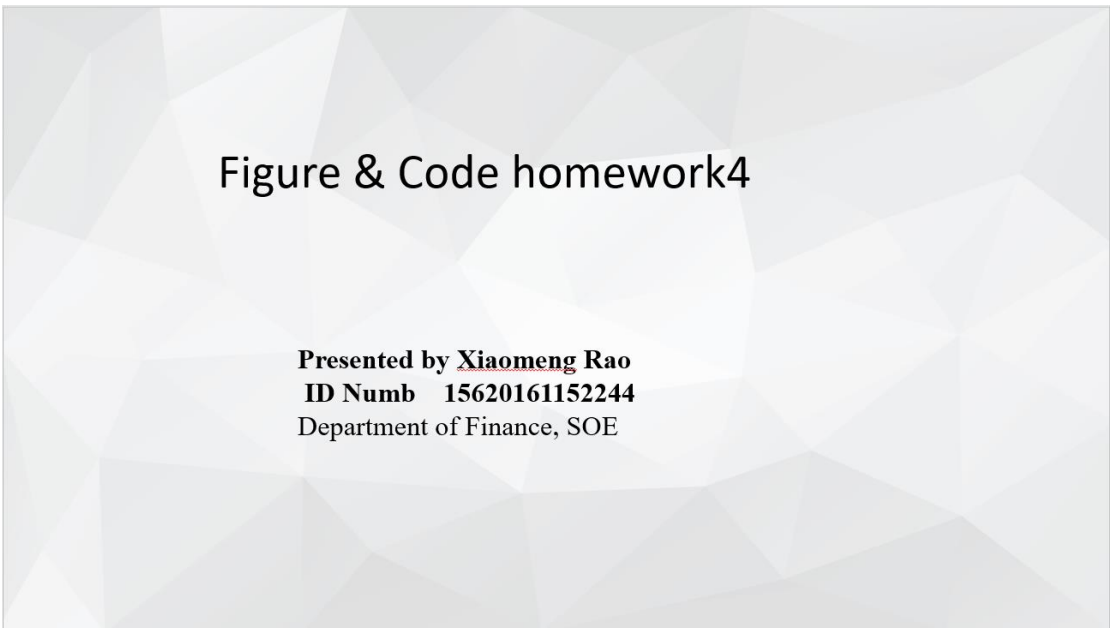
Acf(crixreturn)

Pacf(crixreturn)

arima(crixreturn,order = c(2,0,2))



Homework 4

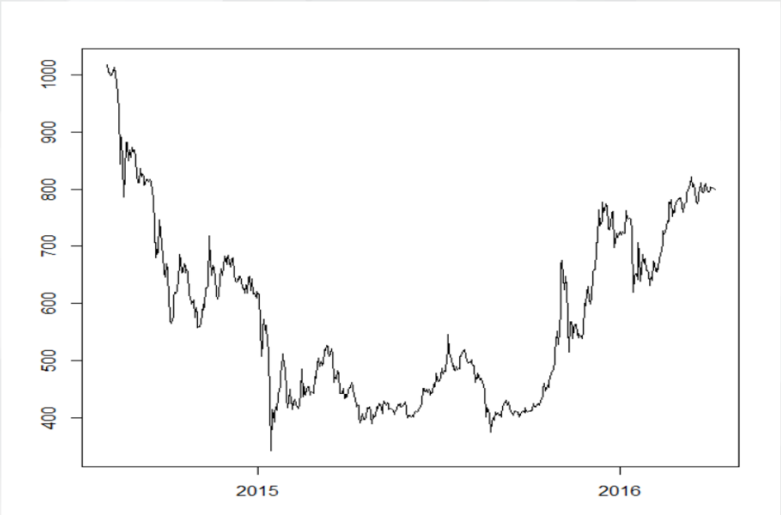


1. Figure3,4,5,6

```
#HW4.1
library(rjson)
json_file = "http://crix.hu-berlin.de/data/crix.json"
json_data = fromJSON(file=json_file)
lst <- lapply(json_data,function(x){
  df<-data.frame(date=x$date,price=x$price)
  return(df)
})
crix_data_frame <- Reduce(rbind,lst)
crix_data_frame <- crix_data_frame[-1,]
load(file = "ecrix.RData")
load(file = "efcrix.RData")
```

```
length(ecrix)=length(crix_data_frame$price)

length(efcrix)=length(crix_data_frame$price)

ecrix_data_frame <- as.data.frame(ecrix)

efcrix_data_frame <- as.data.frame(efcrix)

#install.packages("dplyr")

library(dplyr)

sum_crix <- cbind(crix_data_frame,ecrix_data_frame,efcrix_data_frame)

#figure3

ts.plot(sum_crix$price)

lines(sum_crix$price,col="black",lwd=0.5)

lines(sum_crix$ecrix,col="blue",lwd=1)

lines(sum_crix$efcrix,col="red",lwd=1)
```

Figure 3

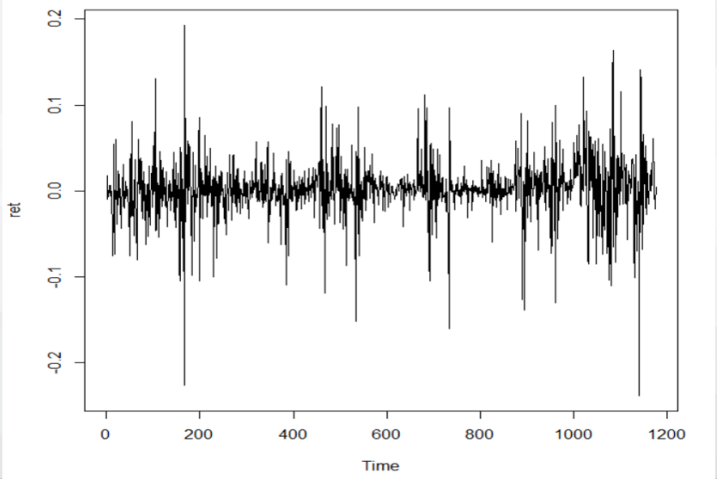


#figure4

crixreturn <- diff(log(crix_data_frame$price))

ts.plot(crixreturn)

Figure 4



#figure5

hist(crixreturn,col = "grey",breaks = 20,freq = FALSE,ylim = c(0,25),xlab = NA)

lines(density(crixreturn),lwd=1)

mu = mean(crixreturn)

sigma = sd(crixreturn)

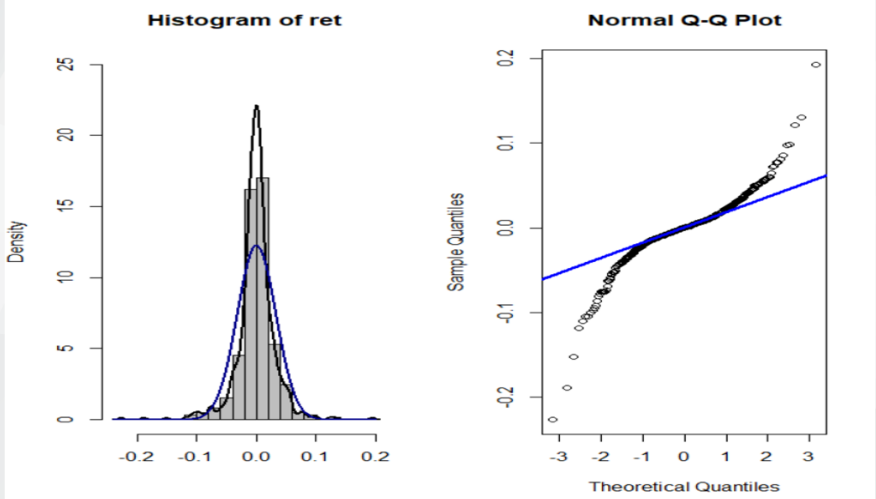
x = seq(-4, 4, length = 100)

curve(dnorm(x, mean = mean(crixreturn), sd = sd(crixreturn)), add = TRUE, col = "darkblue", lwd = 1)

qqnorm(crixreturn)

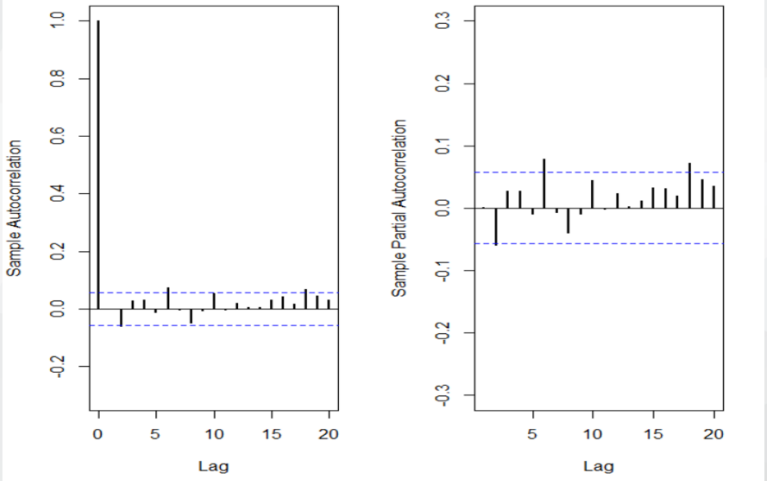
qqline(crixreturn, col = "blue", lwd = 2)

Figure 5




```
#figure6

Box.test(crixreturn, type = "Ljung-Box", lag = 20)

adf.test(crixreturn, alternative = "stationary")

kpss.test(crixreturn, null = "Trend")

par(mfrow = c(1, 2))

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

print(cbind(autocorr$lag, autocorr$acf))

Box.test(crixreturn, type = "Ljung-Box", lag = 1, fitdf = 0)

Box.test(autocorr$acf, type = "Ljung-Box")

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

Figure 6



2. Figure 7

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

auto.arima(crixreturn)

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

tsdiag(fit1)

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


aic = matrix(NA, 6, 6)

for (p in 0:4) {

   for (q in 0:3) {

      a.p.q = arima(crixreturn, order = c(p, 0, q))

      aic.p.q = a.p.q$aic

      aic[p + 1, q + 1] = aic.p.q
```

```
  }
}
aic

bic = matrix(NA, 6, 6)
for (p in 0:4) {
  for (q in 0:3) {
    b.p.q = arima(crixreturn, order = c(p, 0, q))
    bic.p.q = AIC(b.p.q, k = log(length(crixreturn)))
    bic[p + 1, q + 1] = bic.p.q
  }
}
bic
fit4 = arima(crixreturn, order = c(2, 0, 3))
tsdiag(fit4)
Box.test(fit4$residuals, lag = 1)
fitr4 = arima(crixreturn, order = c(2, 1, 3))
tsdiag(fitr4)
Box.test(fitr4$residuals, lag = 1)
fit202 = arima(crixreturn, order = c(2, 0, 2))
tsdiag(fit202)
tsdiag(fit4)
```

```
tsdiag(fitr4)

AIC(fit202, k = log(length(crixreturn)))

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

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

fit202$aic

fit4$aic

fitr4$aic

fit202 = arima(crixreturn, 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(crixreturn))

plot(crixreturn, type = "l", xlim = c(0, 1200), ylab = "log return", xlab =
"days", lwd = 1)

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)
```

Figure7