**Homework**

JIA ZHU（朱佳）, 27720151153580, WISE

**Homework 1**

**Question 1**

**The development of compute memory**

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |  |  |

**Question 2**

**Brief explanation of Logistic regression**

The linear regression model assumes that the response Y is quantitative. But in many situations, the response variable is qualitative. For example, eye color is qualitative, taking on values blue, brown, or green. It is unreasonable to assume that the dependent variable Y is normal distribution. Therefore, the outcome variable may be categorical such as binary variables(for example, yes/no, passed/failed, lived/died) and polytomous variables(for example, poor/good/excellent).

Logistic regression is applied to situations in which the response variable Y is binary (0,1). We use a linear regression model to represent the logistic or logit function of p:

, (1)

where . And we can know that is the conditional mean of Y. In this case,  is the link function.

Equation (1) can be rewritten as

 .

The logit function can take any real value, but the associated probability always lies in the required [0,1] interval. The parameters of the logistic regression model are estimated by maximum likelihood.

**Homework 2**

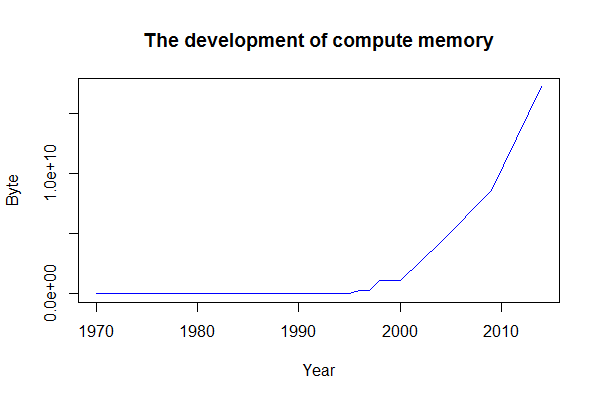
**Question 1**

**R Codes:**

## question 1 ##

Byte <- read.csv("byte.csv",header = TRUE)

plot(Byte, type="l", col="blue", main="The development of compute memory")



**Question 2**

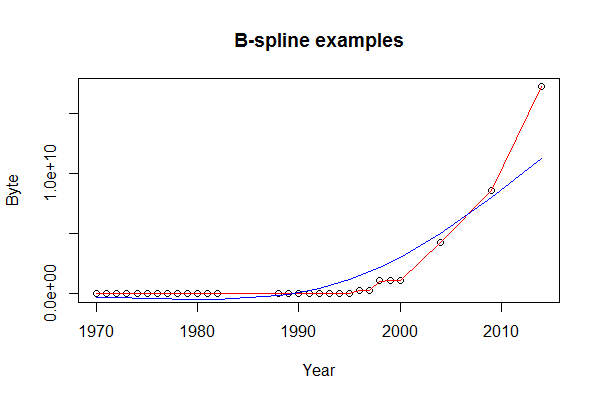
**R Codes:**

## question 2 ##

plot(Byte, type="p", main="B-spline examples")

lines(smooth.spline(Byte, spar = 0.1),type = "l", col="red")

lines(smooth.spline(Byte, spar = 0.9),type = "l", col="blue")



**Question 3**

Suppose we observe that in n=1000 mails (in 1 week) we have about 2 scams. In order to calculate that we have 6 scam emails in 2 weeks, we should calculate that we have 3 scam emails in 1 week. Therefore,

 .

In Scammyland you have 5 scams on average, what is the probability to have no scam mail:

 .

**Homework 3**

**Question 1**

**R Codes:**

## question 1 ##

# 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")

# the result is "a8d3e4701672195e5dcd16ea9b062279"

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

# the result is "059ab10d478614d2eab3d70cfccd3fcc"

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

# the result is "c16700de5a5c1961e279135f2be7dcf9c187cb6b21ac8032308c715e1ce9964c"

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

# the result is "f5e2cba48dac097355d0bb310fdbd5bd38a22a5c8e8215cd1ae67014cfc35b91"

**Question 2**

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.

DSA consists of 2 parts: generation of a pair of public key and private key; generation and verification of digital signature.

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

Reference:

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

<http://www.herongyang.com/Cryptography/DSA-Introduction-What-Is-DSA-Digital-Signature-Algorithm.html>

**Question 3**

**R Codes:**

## question 3 ##

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\_data\_frame = as.data.frame(json\_data)

a <- 1:(ncol(crix\_data\_frame)/2)

n <- 2\*a

m <- n-1

time <- t(crix\_data\_frame[m])

price <- t(crix\_data\_frame[n])

mydata <- cbind(time,price)

ts.plot(price, main="The trend of CRIX")

install.packages("tseries")

library(tseries)

adf.test(price) #p-value = 0.99, accept H0: price is nonstationary

acf(price)

pacf(price)

install.packages("forecast")

library(forecast)

auto.arima(price)# ARIMA(5,2,0)

Return <- diff(log(price))

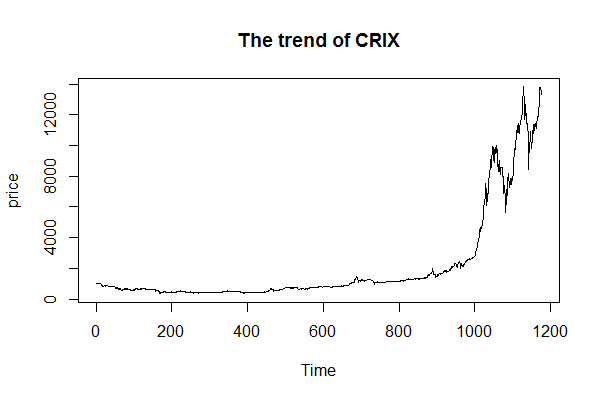
ts.plot(Return,main="The log returns of CRIX")

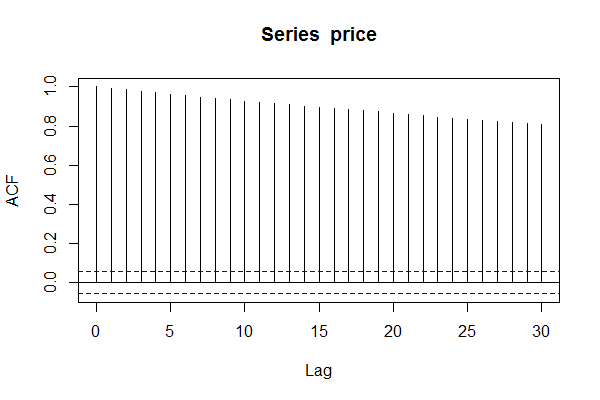
adf.test(Return)#p-value = 0.01, Return is stationary

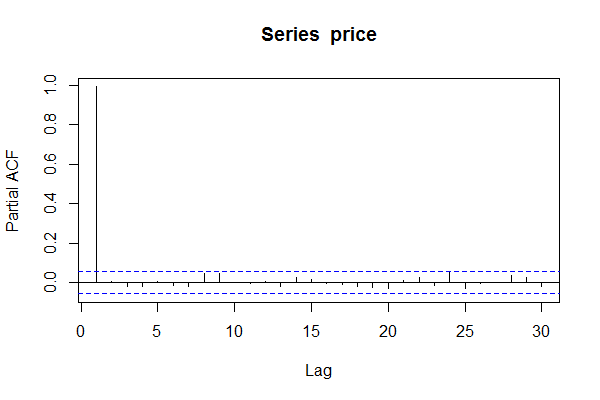
acf(Return)

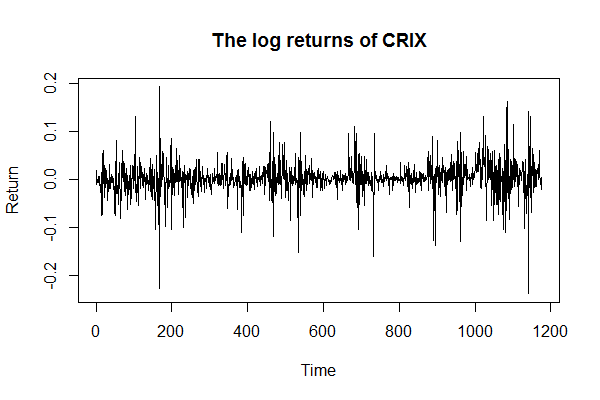
pacf(Return)

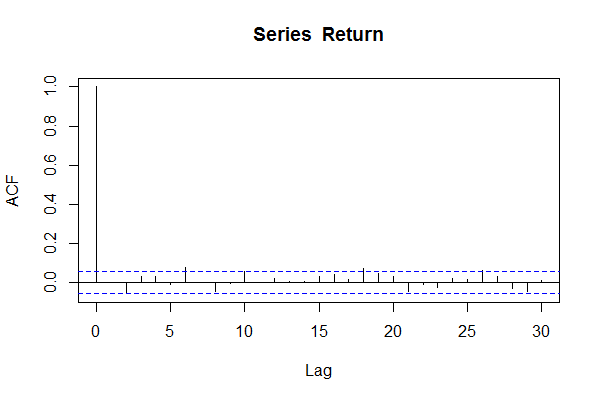
auto.arima(Return)# ARIMA(5,2,0)

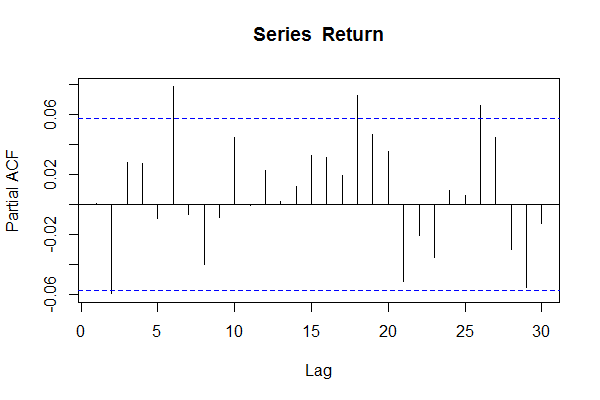












**Homework 4**

**Question 1**

**R Codes:**

# figure 3 #

load("ecrix.RData")

load("efcrix.RData")

plot(ecrix, type = "l", col = "blue", xaxt = "n",main = " Daily value of indices in the CRIX family", xlab = "Time", ylab = "Daily Value of Indices")

lines(efcrix, col = "black")

lines(price, col = "red")

axis(1,c(120,520),c("2015","2017"))

# figure 4 #

plot(Return, type = "l", xaxt='n',xlab="Time",ylab = "Price", main = "The log returns of CRIX")

axis(1,c(120,420,720,1020),c("2014","2015","2016","2017"))

# figure 5 #

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

lines(density(Return), lwd = 2)

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

curve(dnorm(x, mean = mean(Return), sd = sd(Return)), add=TRUE, col = "blue", lwd = 2)

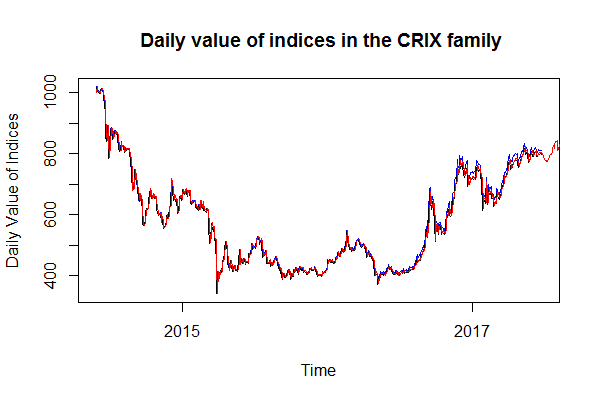
qqnorm(Return)

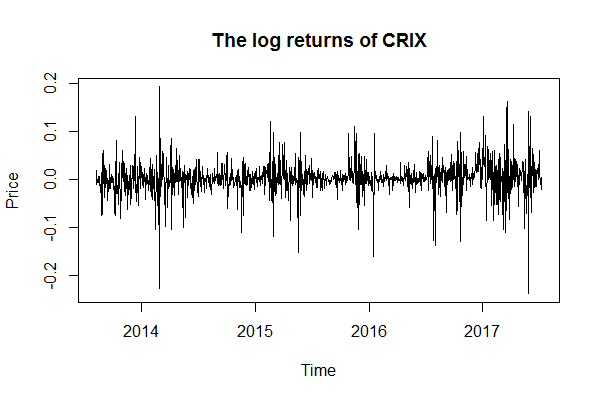
qqline(Return,col="blue",lwd=2)

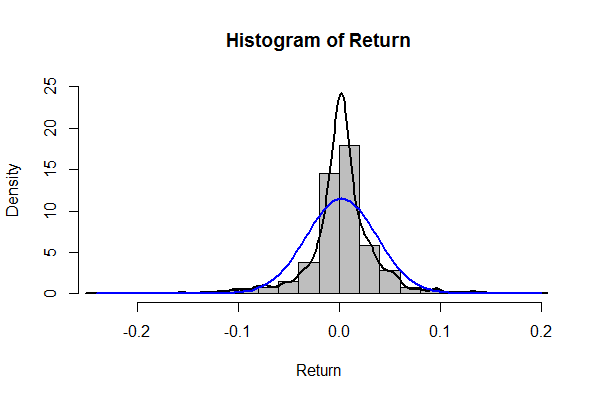
# figure 6 #

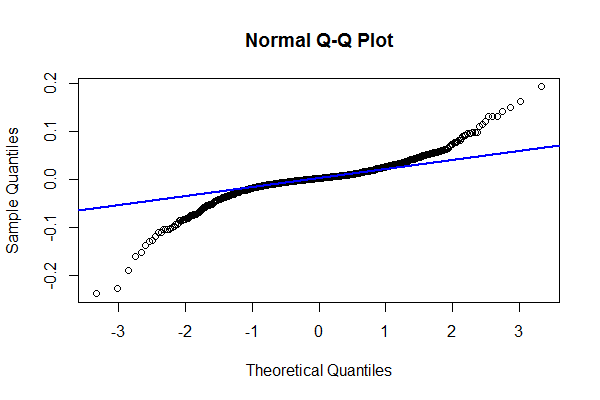
acf(Return)

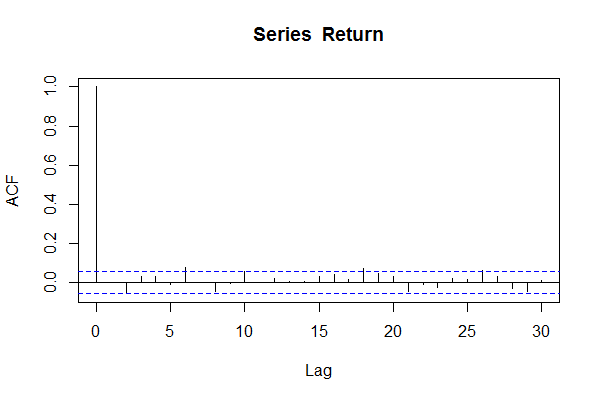
pacf(Return)

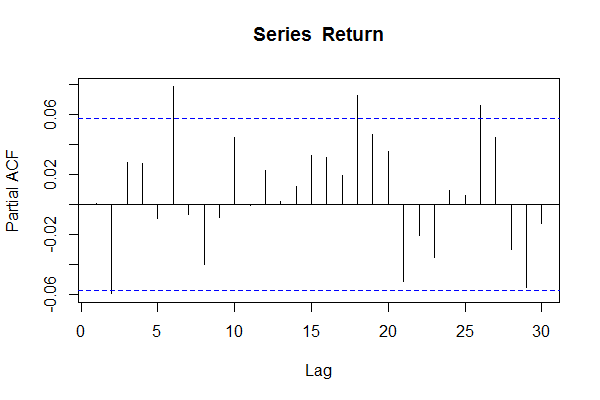












**Question 2**

**R Codes:**

aic = matrix(NA, 6, 6)

for (p in 0:4) {

for (q in 0:3) {

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

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

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

}

}

aic

# bic

bic = matrix(NA, 6, 6)

for (p in 0:4) {

for (q in 0:3) {

b.p.q = arima(Return, 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

library(TTR)

library(TSA)

library(caschrono)

library(forecast)

fit1 = arima(ret, order = c(3, 0, 3))

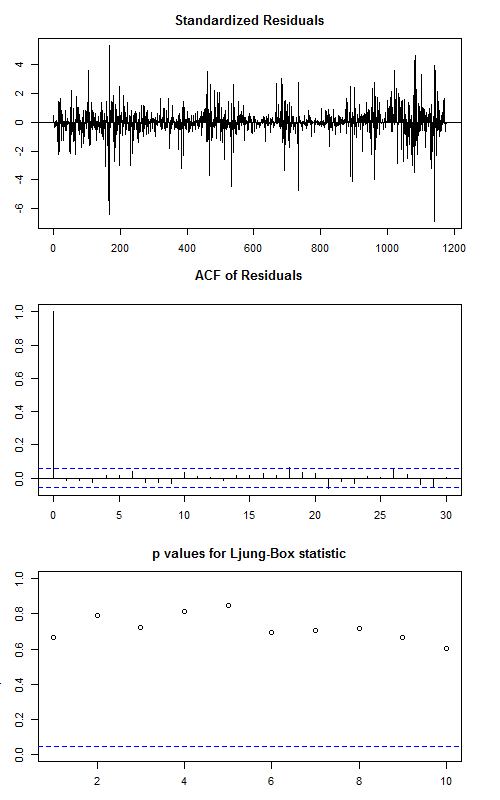
par(mar=c(3,3,3,3))

tsdiag(fit1)

#plot(fit1$residuals, xaxt='n',ylab="Residuals", main="Residuals of ARMA(3,3)")

#axis(1,c(120,420,720,1020),c("2014","2015","2016","2017"))

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



**Homework 5**

**R Codes:**

library(RCurl)

library(XML)

library(stringr)

url = paste("http://shakespeare.mit.edu/romeo\_juliet/full.html")

#url = paste("http://shakespeare.mit.edu/julius\_caesar/full.html")

#url = paste("http://shakespeare.mit.edu/hamlet/full.html")

abs = lapply(url, FUN = function(x) htmlParse(x, encoding = "ANSI"))

clean\_txt = function(x) {

cleantxt = xpathApply(x, "//body//text()

[not(ancestor :: script)][ not(ancestor :: style)]

[not(ancestor :: noscript)] " ,xmlValue)

cleantxt = paste(cleantxt, collapse="\n")

cleantxt = str\_replace\_all(cleantxt, "\n", " ")

cleantxt = str\_replace\_all(cleantxt, "\r", "")

cleantxt = str\_replace\_all(cleantxt, "\t", "")

cleantxt = str\_replace\_all(cleantxt, "<br>", "")

return(cleantxt)

}

cleantxt = lapply(abs,clean\_txt)

vec\_abs = unlist(cleantxt)

library(NLP)

library(tm)

library(SnowballC)

abs = Corpus(VectorSource(vec\_abs))

abs\_dtm = DocumentTermMatrix(abs, control = list(stemming = TRUE, stopwords = TRUE, minWordLength = 3, removeNumbers = TRUE, removePunctuation = TRUE))

dim(abs\_dtm)

inspect(abs\_dtm)

findFreqTerms(abs\_dtm, 3)

removeSparseTerms(abs\_dtm, 0.5)

inspect(removeSparseTerms(abs\_dtm, 0.5))

library(ggplot2)

library(wordcloud)

freq = colSums(as.matrix(abs\_dtm))

wf = data.frame(word=names(freq), freq=freq)

plot = ggplot(subset(wf, freq>50), aes(word, freq))

plot = plot + geom\_bar(stat="identity")

plot = plot + theme(axis.text.x=element\_text(angle=45, hjust=1))

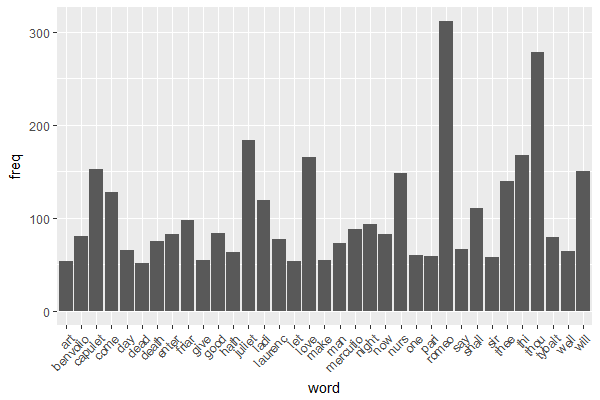
plot

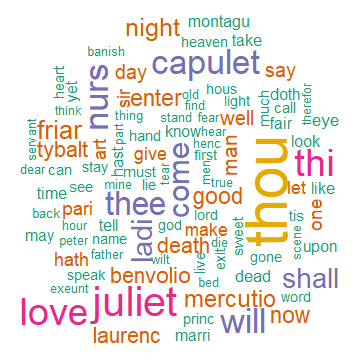
freq = colSums(as.matrix(abs\_dtm))

dark2 = brewer.pal(6, "Dark2")

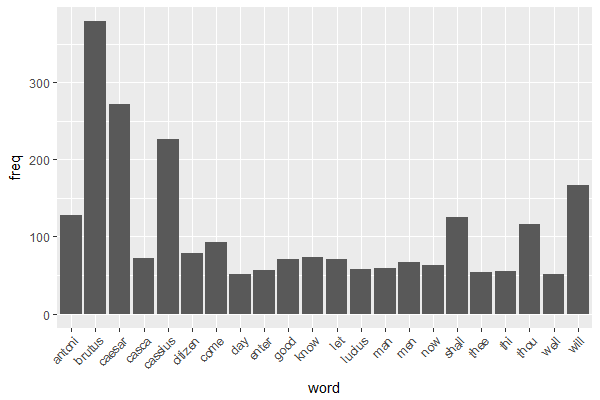
wordcloud(names(freq), freq, max.words=100, rot.per=0.2, colors=dark2)

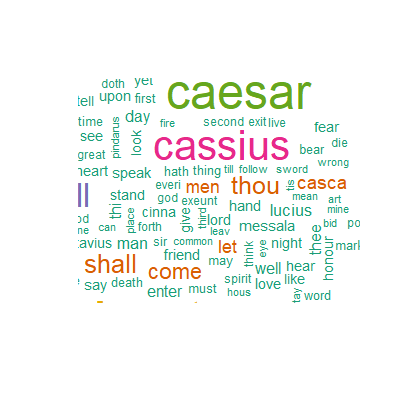
**Romeo and Juliet**





**Julius Caesar**





**Hamlet**

