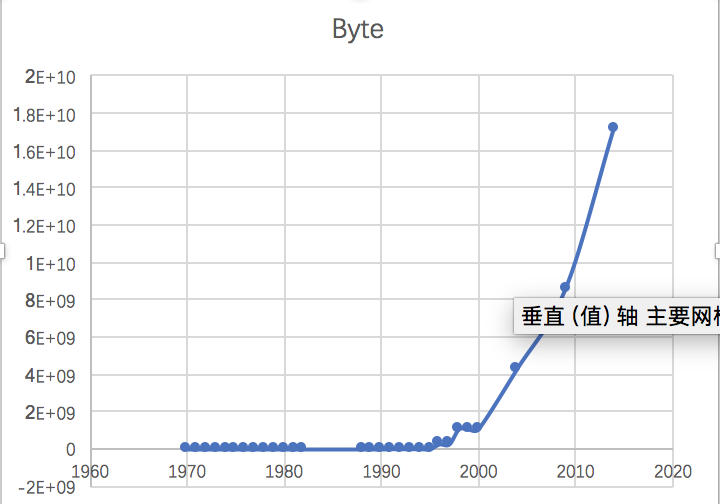
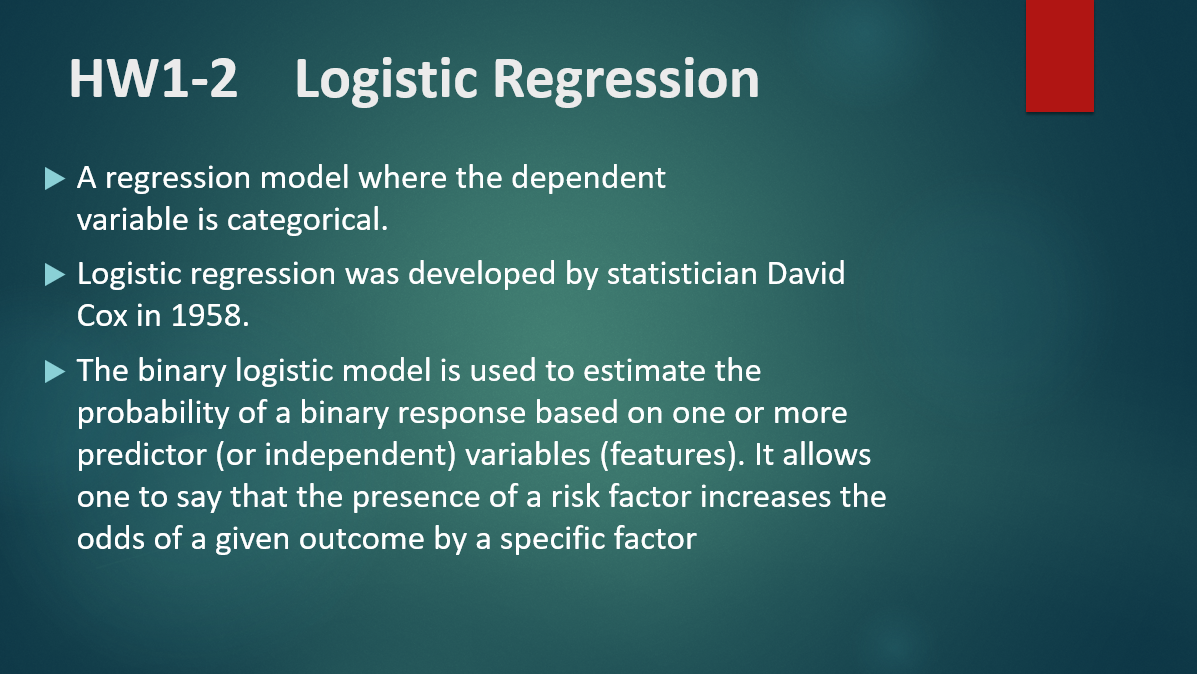
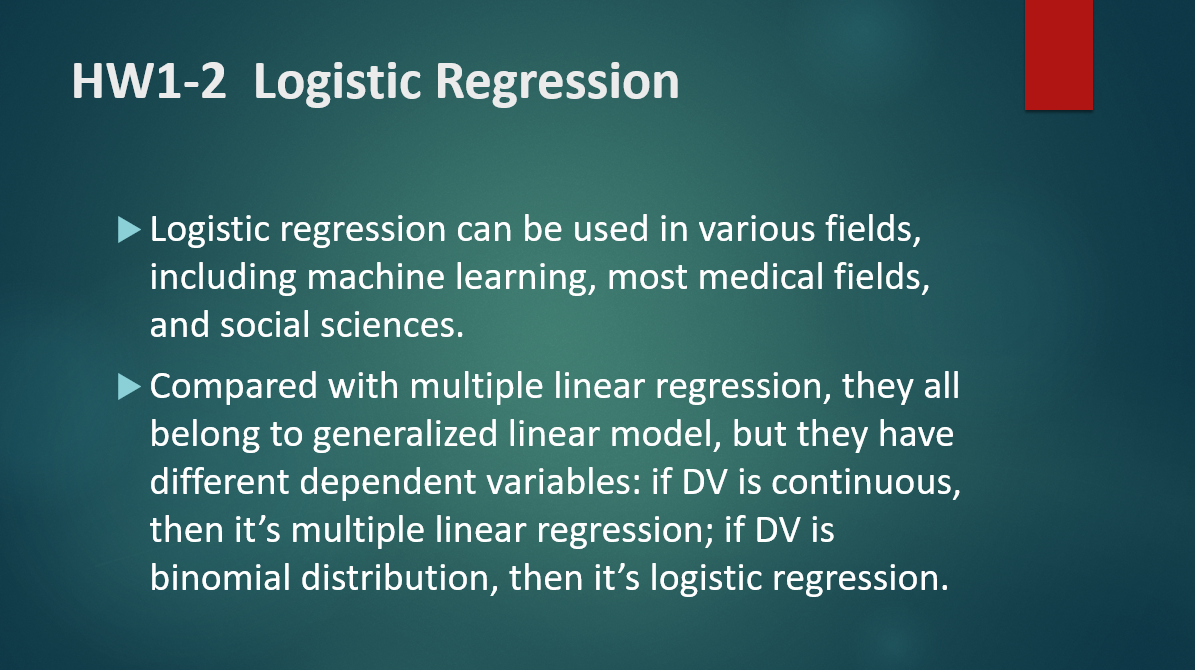
**Homework 1**

|  |  |  |  |
| --- | --- | --- | --- |
| ***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 |  |  |



1. **logistic regression**





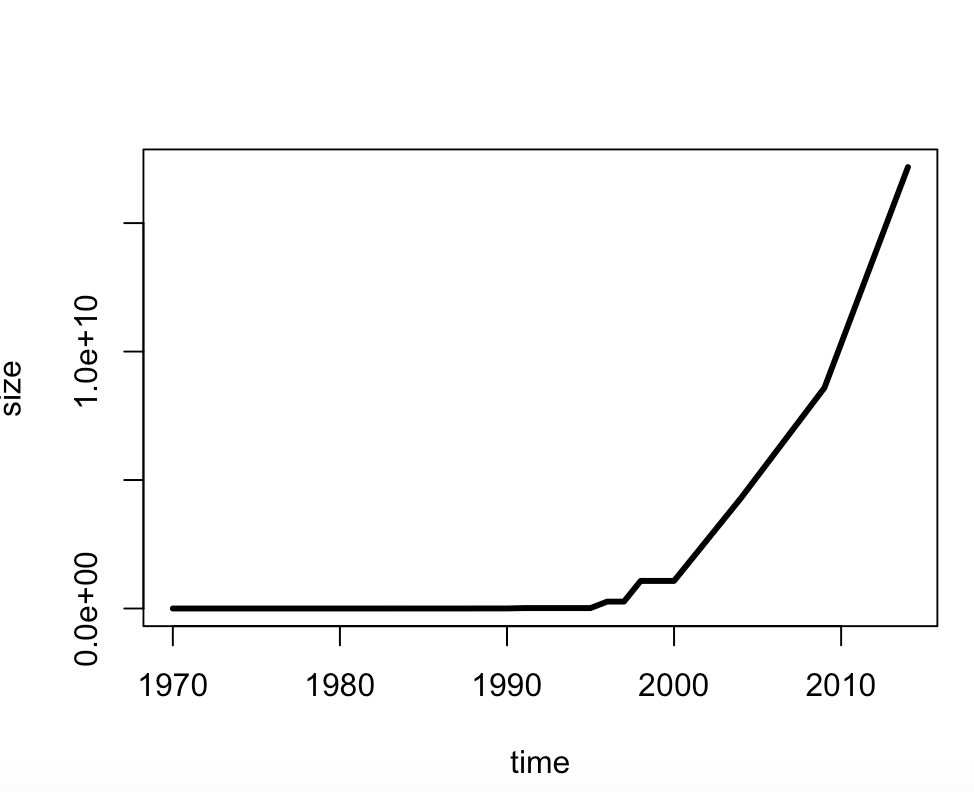
**Homework 2**

1. **Use R to solve HW #1**

x = c (1970,1988,1991,1996,1998,2004,2009,2014)

y = c(262144,2097152,16777216,268435456,1073741824,4294967296,8589934592,17179869184)

plot(x,y)

**

1. **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

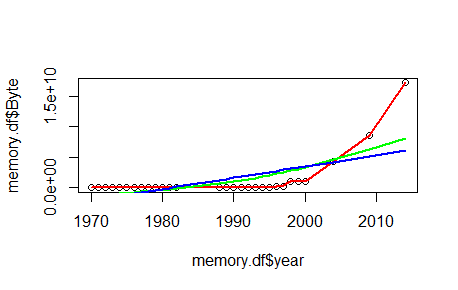
splines.reg.l2 = smooth.spline(x = RAM\_size$year, y = RAM\_size$Byte, spar = 1) # lambda = 1

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.

1. **Poisson Distribution**

lambda=4

x=6

dpois(x,lambda)

lambda=5

x=0

dpois(x,lambda)

**Homework 3**

1. **hash code**

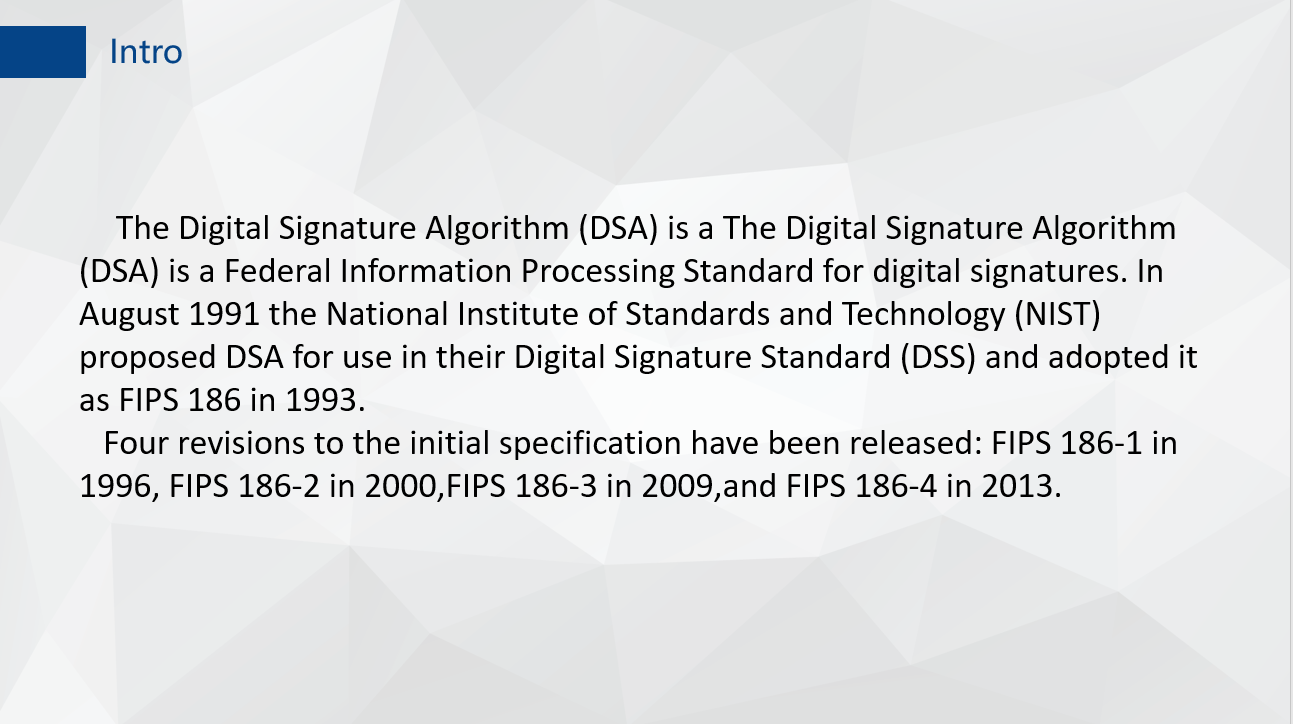
#install.packages("digest",repos='http://cran.us.r-projest.org')

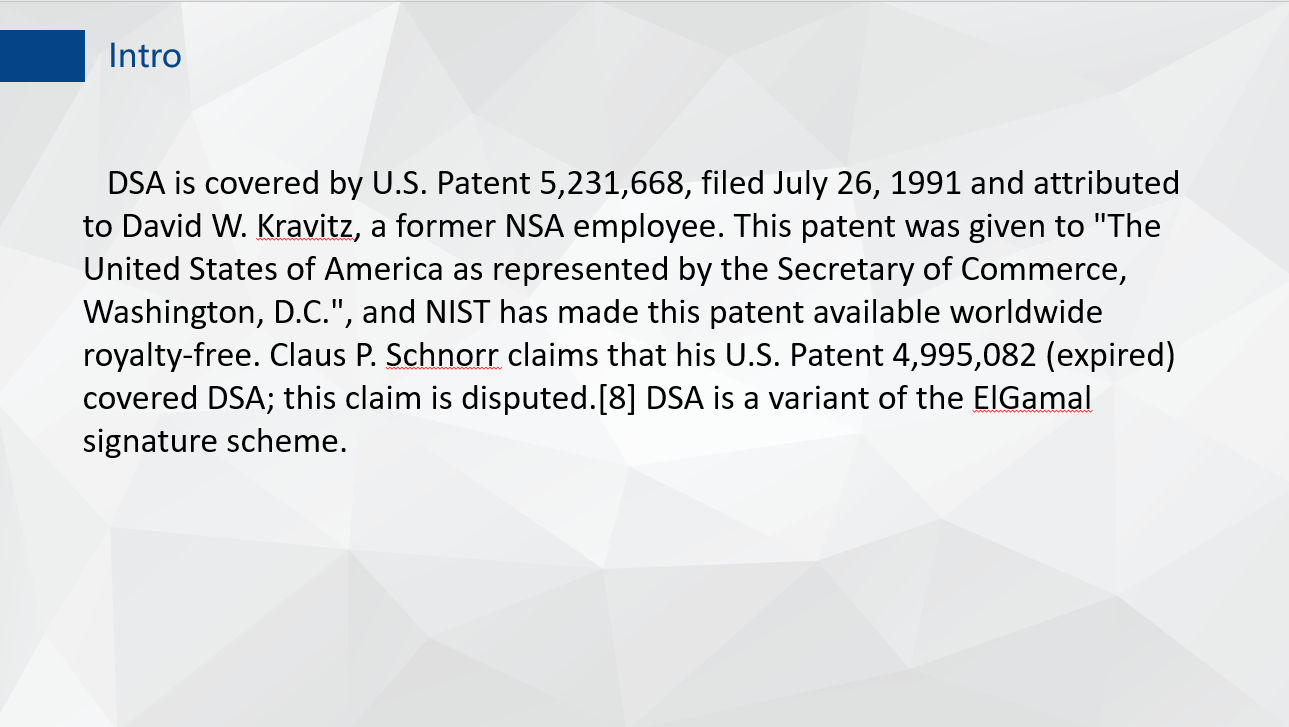
library(digest)

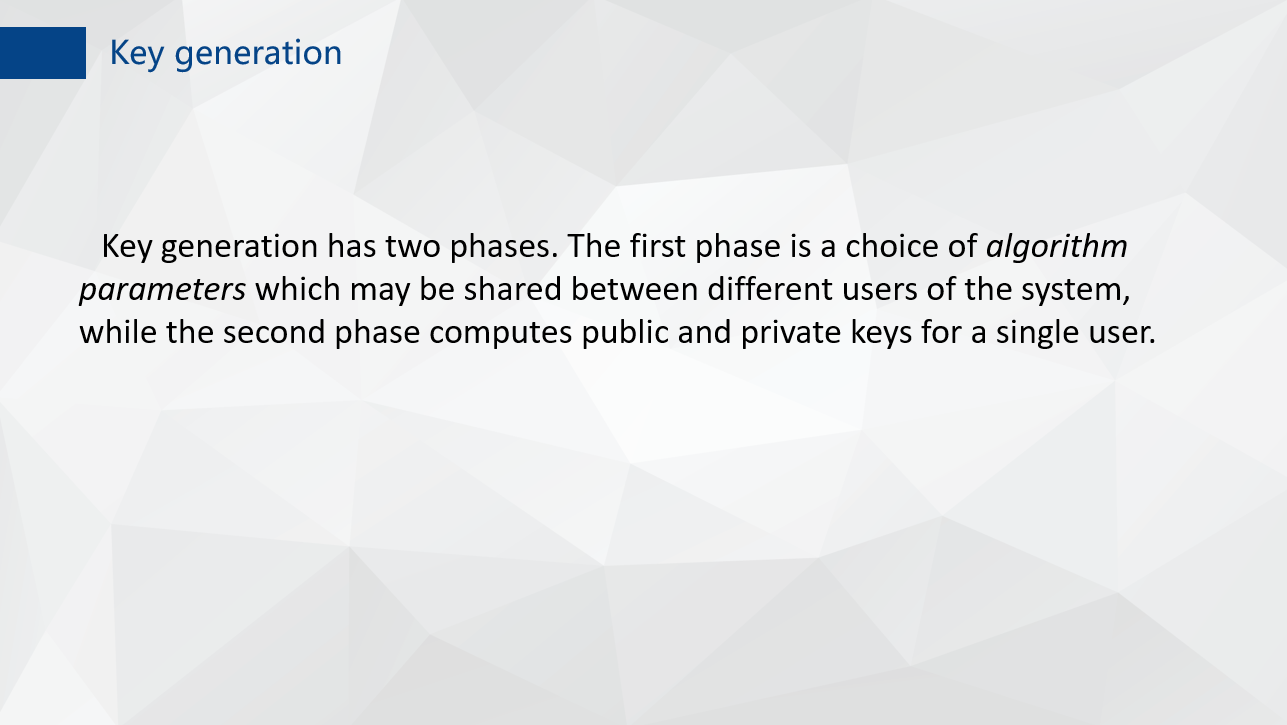
digest("I learn a lot from this class when I am proper listening to the professor","sha256")

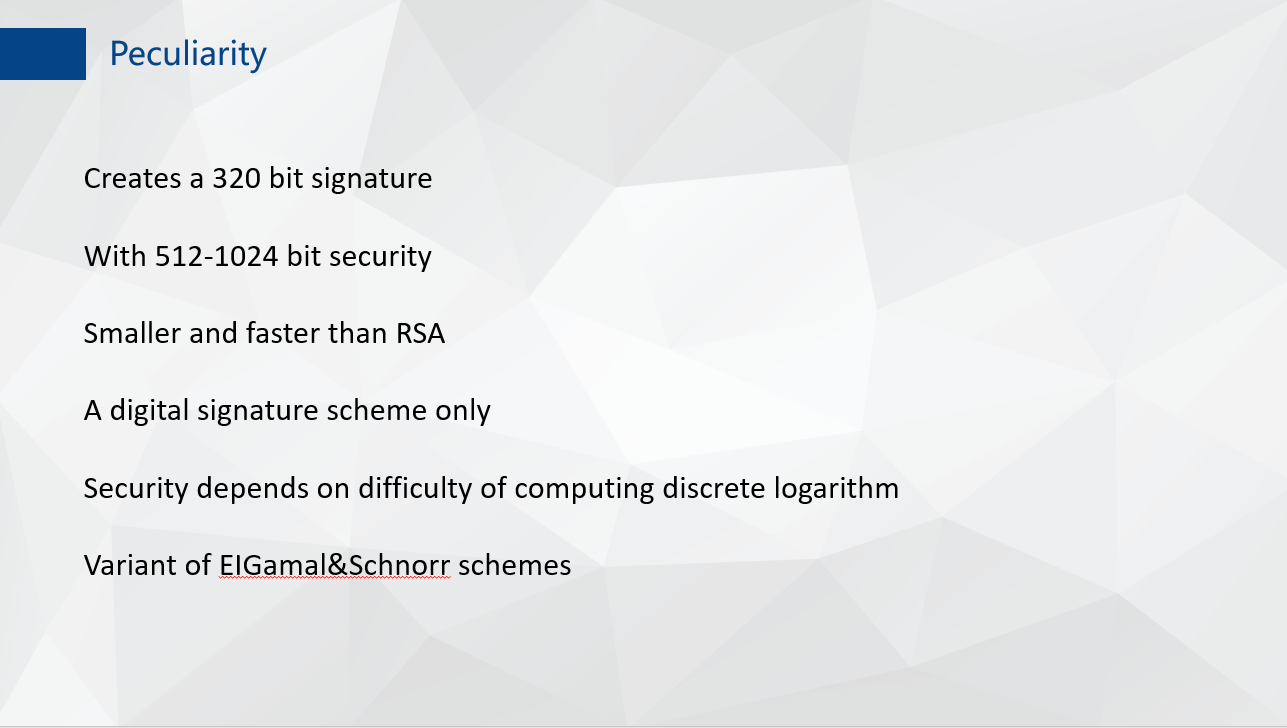
digest("I do not learn a lot from this class when I am absent and playing on my Iphone","sha256")

1. **Digital Signature Algorithms**









Q3

#20171017 JSON input from CRIX , trial done in XMNinstall.packages("rjson",repos="http://cran.us.r-project.org")

library("rjson")

=================

# create and save in txt（save as "try.json" ; "all types" ;）

{

"ID":["001","002","003","004","005","006","007","008" ],

"Salary":["600","500","600","700","800","500","600","70" ],

"StartDate":["1/1/2016","9/23/2017","11/15/2017","5/11/2017,"3/27/2017","5/21/2017",

"7/30/2016","6/17/2016"],

}

====================

install.packages("rjson")

# Load the package required to read JSON files.

library("rjson")

# Give the input file name to the function.

result <- fromJSON(file = “try.json")

# Print the result.

print(result)

Q4

#20171017 JSON input from CRIX , trial done in XMNinstall.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)

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])

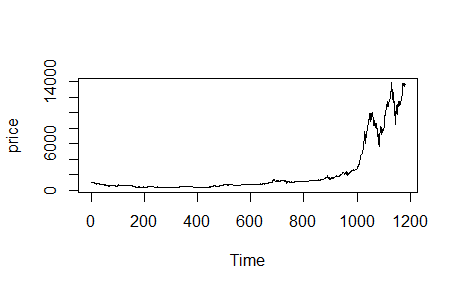
plot(price)

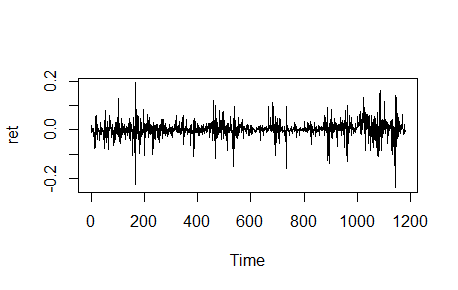
dim(price)

ts.plot(price)

ret = diff( log(price) )

ts.plot( ret )





HW 4

Q1

Q1

#20171017 JSON input from CRIX , trial done in XMN

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)

x = crix\_data\_frame

n = dim(x)

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

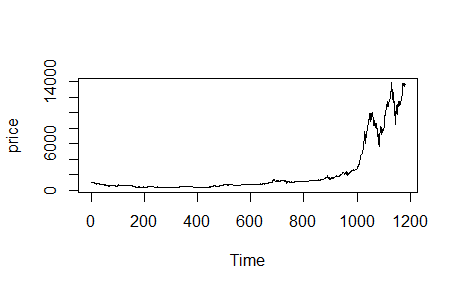
b = seq(2,n[2],2)

#figure 3 : The daily value of CRIX

date = t(x[1,a])

price = t(x[1,b])

plot(price)



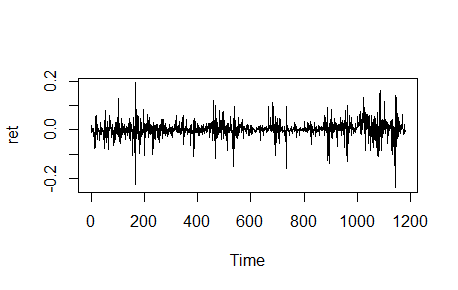
#figure 4 : The log returns of CRIX index

dim(price)

ts.plot(price)

ret = diff( log(price) )

ts.plot( ret )



#figure 5 : Histogram

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

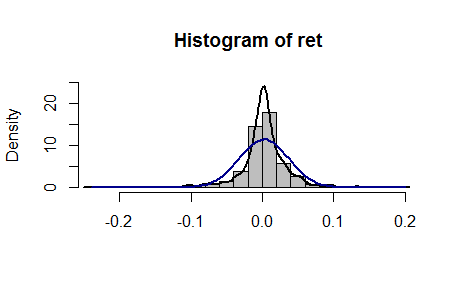
lines(density(ret), lwd = 2)

mu = mean(ret)l,

sigma = sd(ret)

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

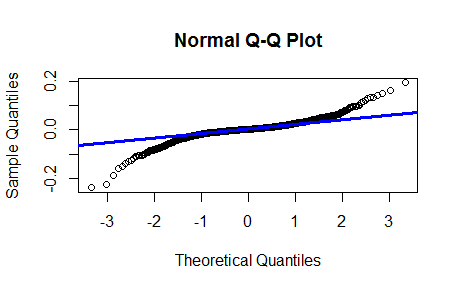
curve(dnorm(x, mean = mean(ret), sd = sd(ret)), add = TRUE, col = "darkblue", lwd = 2)



#figure 6 : QQ plot

qqnorm(ret)

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

****

Q2

rm(list = ls(all = TRUE))

graphics.off()

# install and load packages

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

lapply(libraries, function(x) if (!(x %in% installed.packages())) {

install.packages(x)

})

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

#RET

("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)

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])

dim(price)

ret = diff( log(price) )

# d order

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

# stationary test

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

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

par(mfrow = c(1, 2))

# acf plot

autocorr = acf(ret, lag.max = 20, ylab = "Sample Autocorrelation", main = NA,

lwd = 2, ylim = c(-0.3, 1))

# LB test of linear dependence

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

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

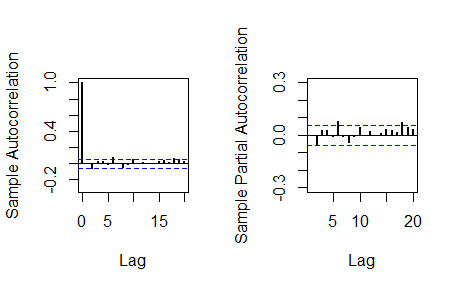
Box.test(autocorr$acf, type = "Ljung-Box")

# plot of pacf

autopcorr = pacf(ret, lag.max = 20, ylab = "Sample Partial Autocorrelation",

main = NA, ylim = c(-0.3, 0.3), lwd = 2)

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



# 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)

# 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

}

}

aic

# 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

}

}

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)

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))

tsdiag(fit202)

tsdiag(fit4)

tsdiag(fitr4)

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

fit202 = arima(ret, 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(ret))

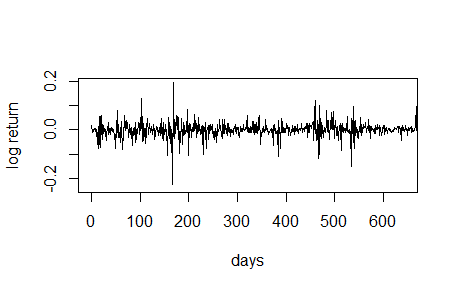
plot(ret, type = "l", xlim = c(0, 644), ylab = "log return", xlab = "days",

lwd = 1.5)

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)



Q3

rm(list = ls(all = TRUE))

graphics.off()

# install and load packages

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

lapply(libraries, function(x) if (!(x %in% installed.packages())) {

install.packages(x)

})

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

# plot of crix return

ret = diff(log(crx$Pr))

Dare = factor(date1[-1])

retts = data.frame(Dare, ret)

# comparison of different crix returns

par(mfrow = c(2, 2))

plot(crx$Da, crx$Pr, type = "o")

lines(crx$Pr)

plot(crx$Da, log(crx$Pr), type = "o")

lines(log(crx$Pr))

plot(retts$Dare, diff(crx$Pr), type = "o")

lines(diff(crx$Pr))

plot(retts$Dare, retts$ret, type = "o")

lines(retts$ret)

# ARIMAfit <- auto.arima(ret, approximation=FALSE,trace=FALSE)

# summary(ARIMAfit)

# arima202 predict

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

# 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)

par(mfrow = c(1, 2))

# plot(res2, ylab='Squared residuals', main=NA)

acfres2 = acf(res2, main = NA, lag.max = 20, ylab = "Sample Autocorrelation",

lwd = 2)

pacfres2 = pacf(res2, lag.max = 20, ylab = "Sample Partial Autocorrelation",

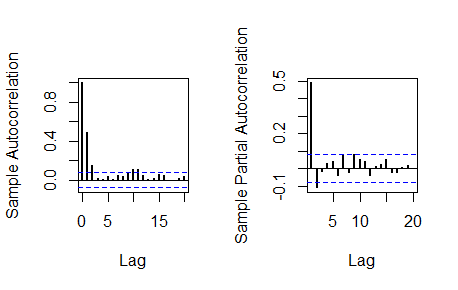
lwd = 2, main = NA)

# arch effect

res = fit202$residuals

ArchTest(res) #library FinTS

Box.test(res2, type = "Ljung-Box")



HW 5

Q1&Q2

rm(list = ls())

library(RCurl)

library(XML)

library(bitops)

library(stringr)

url=paste(c("http://publicliterature.org/pdf/2ws1610.pdf","http://publicliterature.org/pdf/2ws2410.pdf","http://publicliterature.org/pdf/2ws3310.pdf") )

abs=lapply(url, FUN = function(x) htmlParse(x, encoding = "Latin-1"))

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)

vec\_abs

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)

#Find the words that occur more than 5 times

findFreqTerms(abs\_dtm, 5)

#Remove sparse terms

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>100), 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(8, "Dark2")

wordcloud(names(freq), freq, max.words=200, rot.per=0.1, colors=dark2)

dev.off()





hist(freq, col = "grey", breaks = 20,ylim = c(0, 5000), xlab = "freq of words")

