

HW2

#Question 1

setwd("C:/Users/xiumei/Desktop/big data")

read.csv("hw unit2.csv")

plot(year,RAM,type ="o",col="red",main = "RAM of computer")

#Question 3

x = 6

n = 1000

lambda = 2

p = lambda / n

dbinom (x,2\*n,p) # binomial probability mass function

dpois (x, 2\*lambda ) # Poisson probability mass function

dpois (0, 5 )

#Question 2

f<-read.csv("hw unit2.csv")

year<-f$Year;RAM<-f$RAM

plot(year,RAM,type ="o",col="black",main = "RAM of computer")

# load necessary packages

require(datasets)

require(class)

require(grDevices)

require(lattice)

# define log-returns for the DAX and FTS

x= year

y = RAM

# estimated log-returns for the DAX index for different bandwidths

splines.reg.l1 = smooth.spline(x,y, spar = 0.2) # lambda = 0.2

splines.reg.l2 = smooth.spline(x,y, spar = 1) # lambda = 1

splines.reg.l3 = smooth.spline(x,y, spar = 2) # lambda = 2

# plot for the regression results

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

HW3

**What Is DSA (Digital Signature Algorithm)?**

Digital signatures are essential to **verify the sender of a document’s identity.** The signature is computer using a set of rules and algorithm such that the identity of the person can be verified.

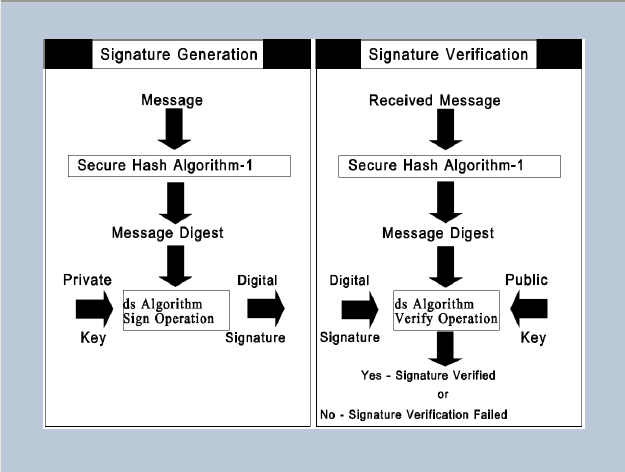
The signature is generated by the use of **a private key** that known only to **the user.** The signature is verified when a public key is corresponds to the private key. With every user having a public/private key pair, this is an example of public-key cryptography.

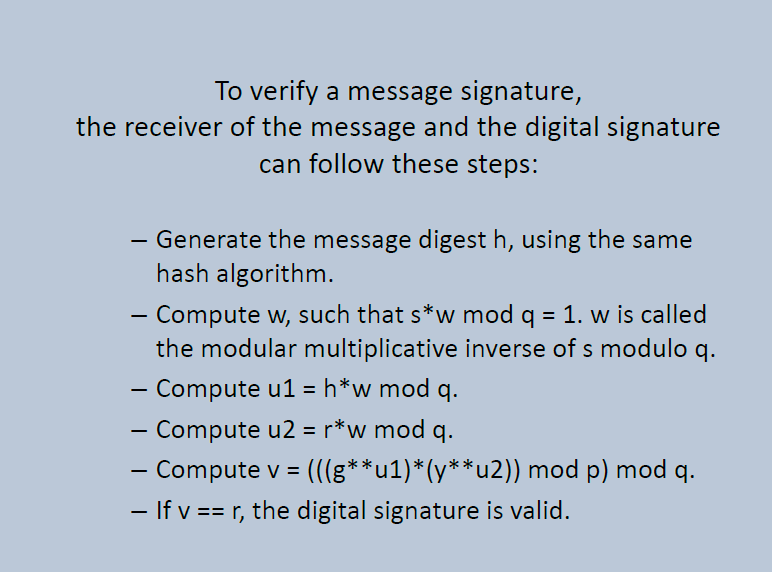
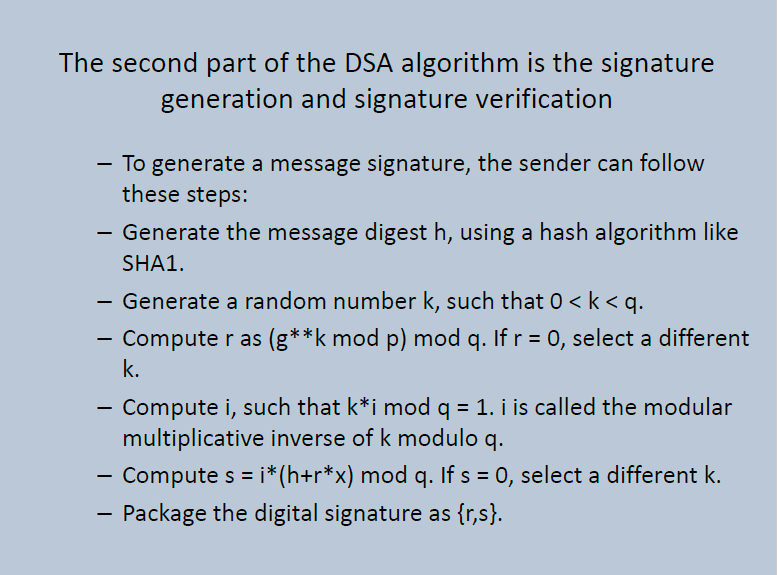
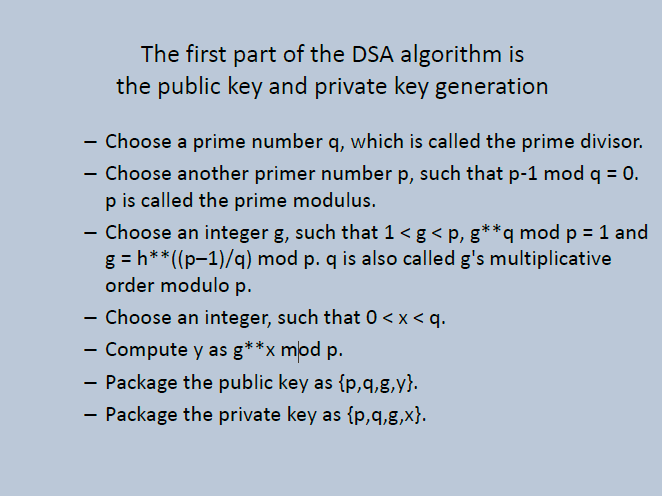
**Public keys,** which are known by everyone, can be used to verify the signature of a user**. The private key**, which is never shared, is used in signature generation, which can only be done by the user.

**What can DSA do?**

Digital signatures are used to detect unauthorized modifications to data. Also, the recipient of a digitally signed document in proving to a third party that the document was indeed signed by the person who it is claimed to be signed by. This is known as nonrepudiation, because the person who signed the document cannot repudiate the signature at a later time.

Digital signature algorithms can be used in e-mails, electronic funds transfer, electronic data interchange, software distribution, data storage, and just about any application that would need to assure the integrity and originality of data.





R-code:

>library(RJSONIO)

> letter<-LETTERS[1:10]

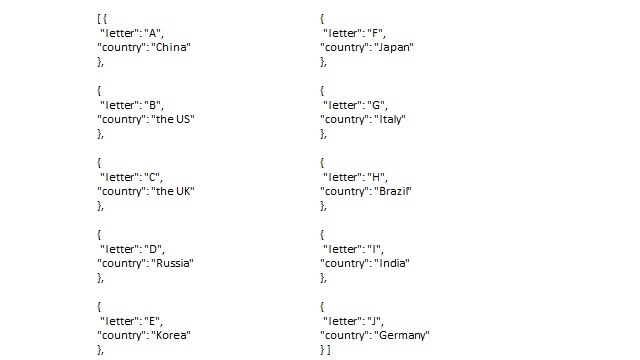
>country<-c("China","the US","the UK","Russia",

"Korea","Japan","Italy","Brazil","India","Germany")

> data<-data.frame(letter,country)

> da<-as.matrix(data)

>cat(toJSON(da))



HW3 Unit3

Answer 1

install.packages("digest")

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")

Answer 4

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)

# load dataset #

load(file = "C:/Users/xiumei/Desktop/big data/crix.RData")

ret = diff(log(crix))

# 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)

HW4

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)

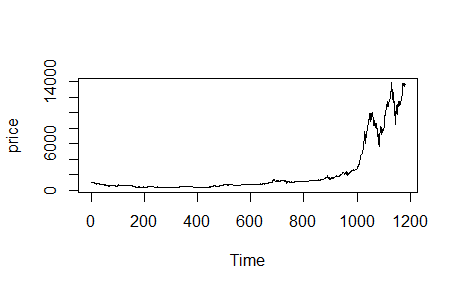
b = seq(2,n[2],2)

#figure 3 : The daily value of CRIX

date = t(x[1,a])

price = t(x[1,b])

plot(price)



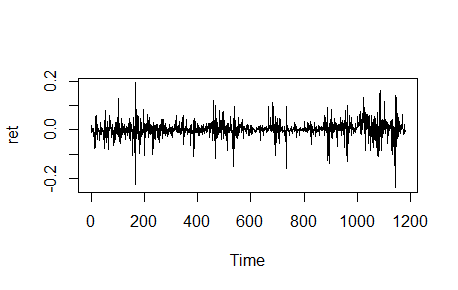
#figure 4 : The log returns of CRIX index

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)

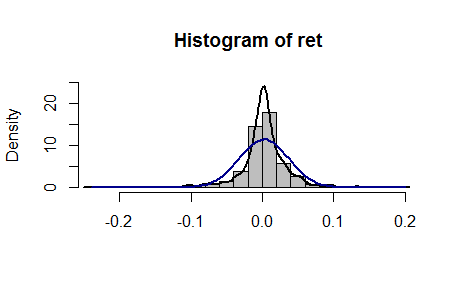
lines(density(ret), lwd = 2)

mu = mean(ret)l,

sigma = sd(ret)

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

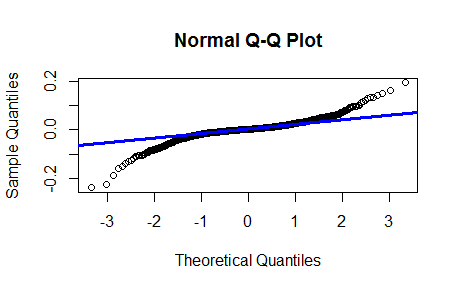
curve(dnorm(x, mean = mean(ret), sd = sd(ret)), add = TRUE, col = "darkblue", lwd = 2)



#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)

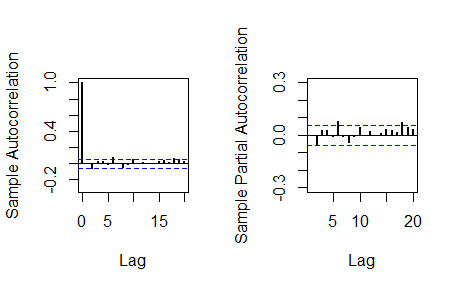
Box.test(autocorr$acf, type = "Ljung-Box")

# 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))

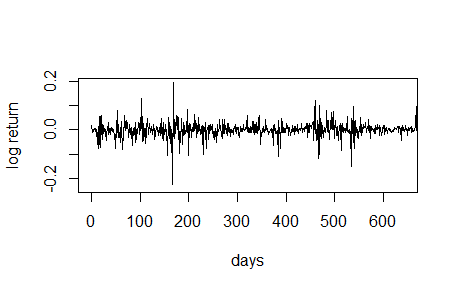
plot(ret, type = "l", xlim = c(0, 644), ylab = "log return", xlab = "days",

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",

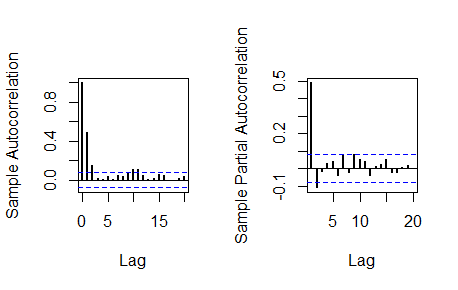
lwd = 2, main = NA)

# 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")

