**Final Exam Project**

**Xiumei Wang284**

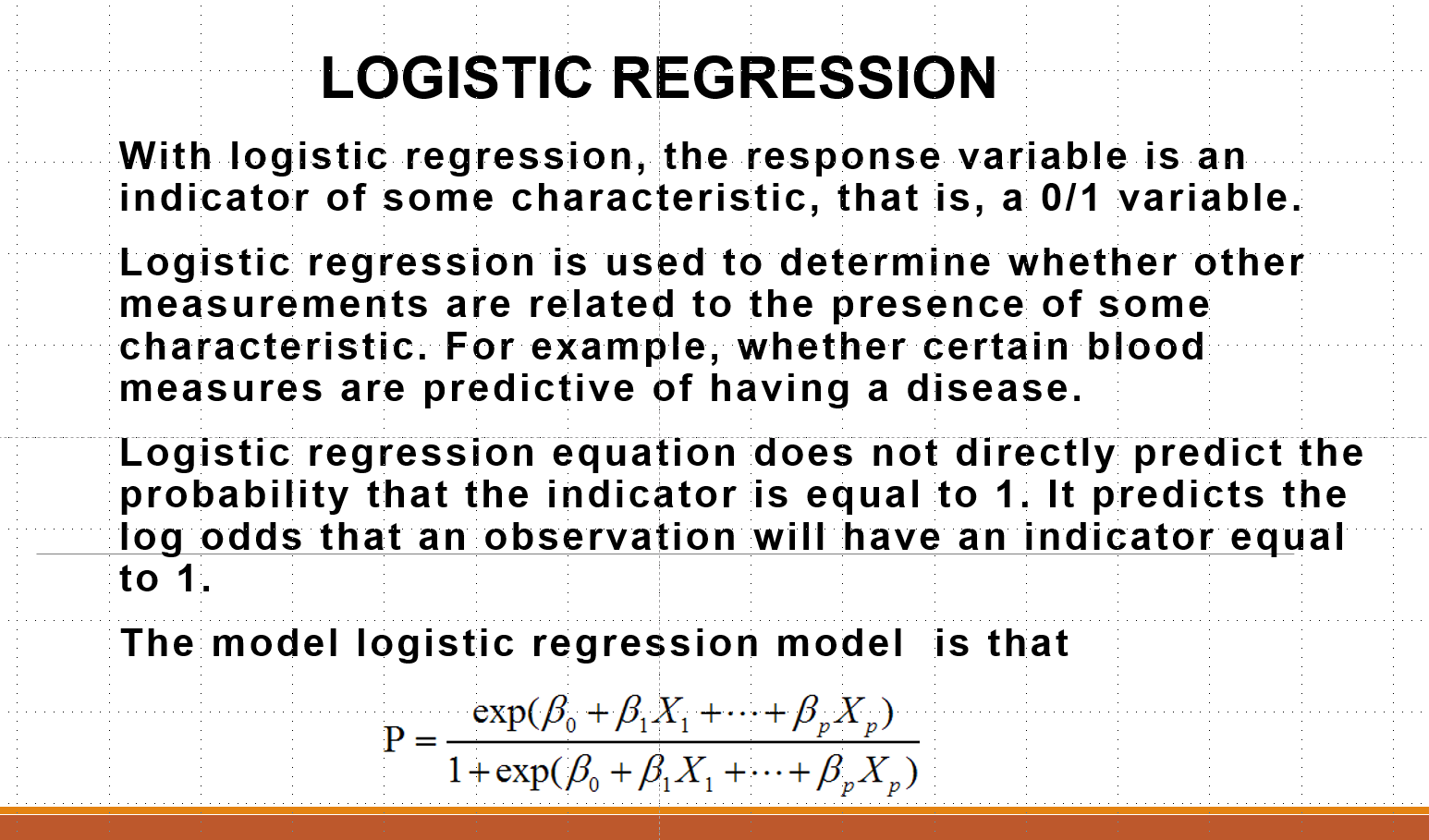
**HW Unit 1**

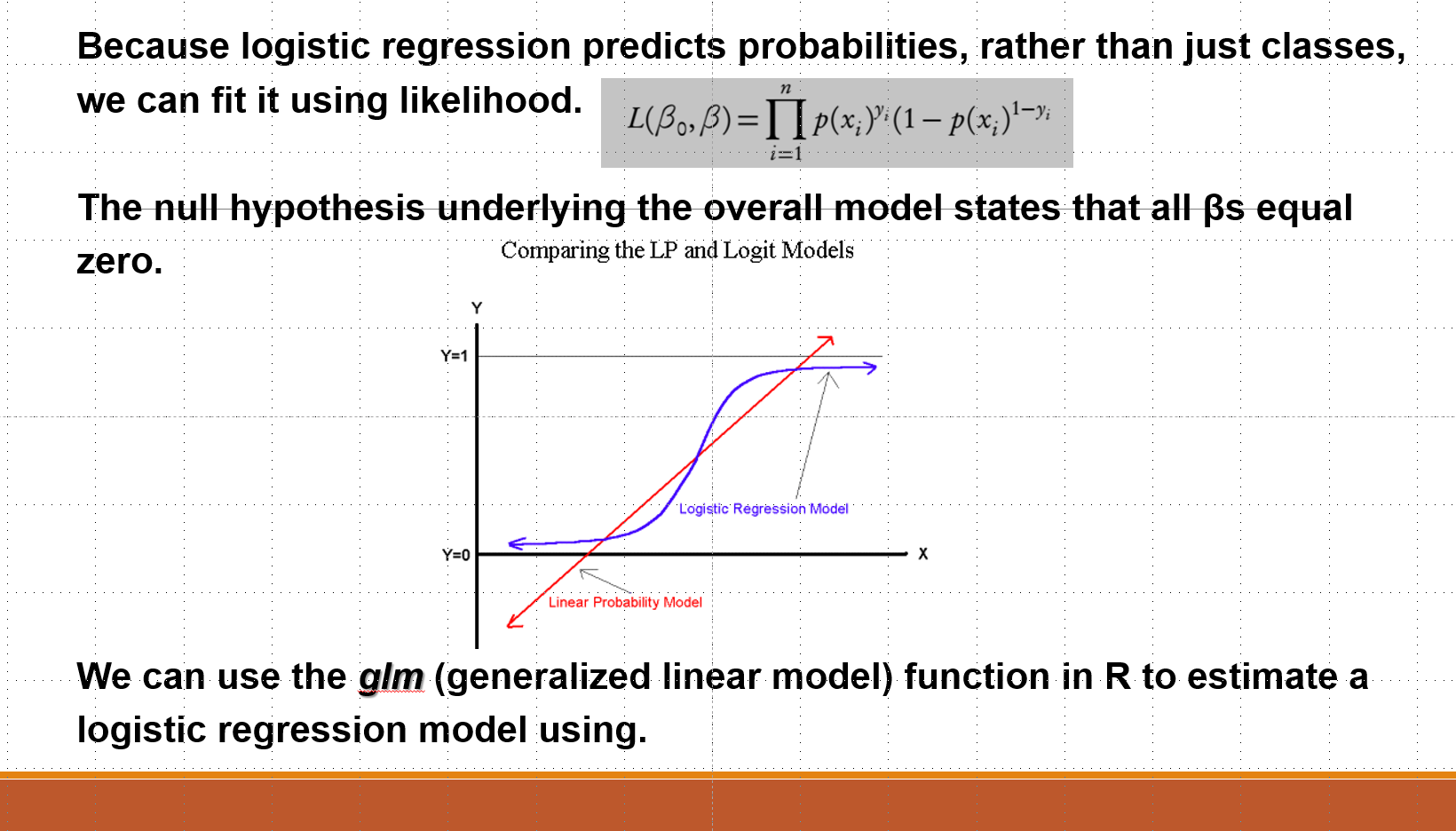
**1.1 Make an R Quantlet to solve HW #1 from unit 1 with R and show it on Github (GH). Hint: use the CMB Qs for this work.**

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

**Comments: There was the rather slow growth before 2000, later, there is dramatically great increase.**

**1.2 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.**





**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.**

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

plot(memory.df$Byte~memory.df$year,type="o",main="The development of internal memory")

**2.2 Use R with B-spline cod2.2 Use R with B-spline code to solve HW#1, any comments?**

splines.reg.l1 = smooth.spline(x = memory.df$year, y = memory.df$Byte, spar = 0.2)

splines.reg.l2 = smooth.spline(x = memory.df$year, y = memory.df$Byte, spar = 1)

splines.reg.l3= smooth.spline(x = memory.df$year, y = memory.df$Byte, spar = 2)

lines(splines.reg.l1, col = "green", lwd = 2)

lines(splines.reg.l2, col = "pink", lwd = 2)

lines(splines.reg.l3, col = "blue", lwd = 2)

**2.3 Suppose you observe that in n=1000 mails (in 1 week) you have about 2 scams. Use the LvB /Poisson cdf 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.**

**#×1**

lambda=2

x=3

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

probex1

**#×2**

lambda=5

x=0

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

probex2

**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.**

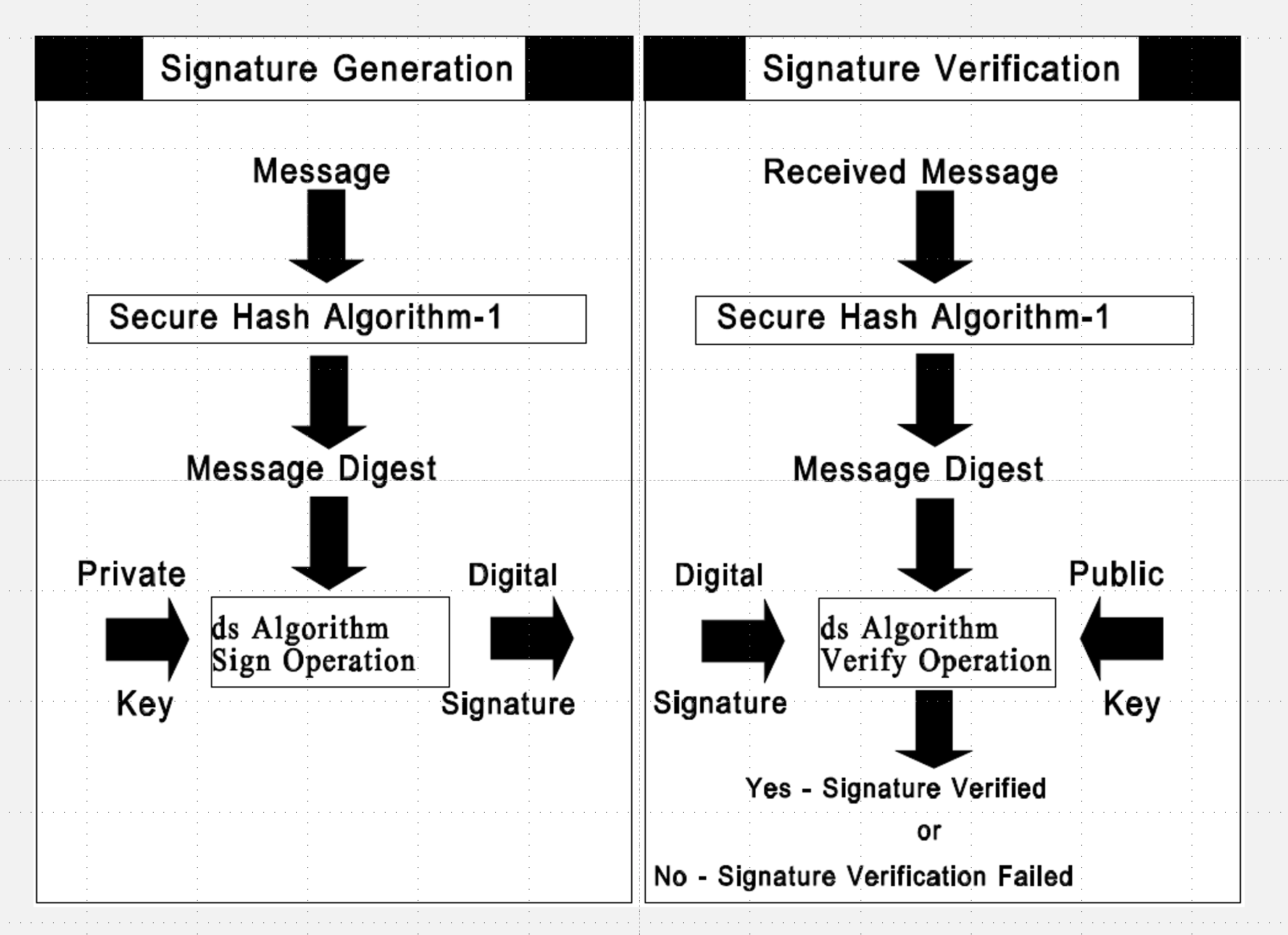
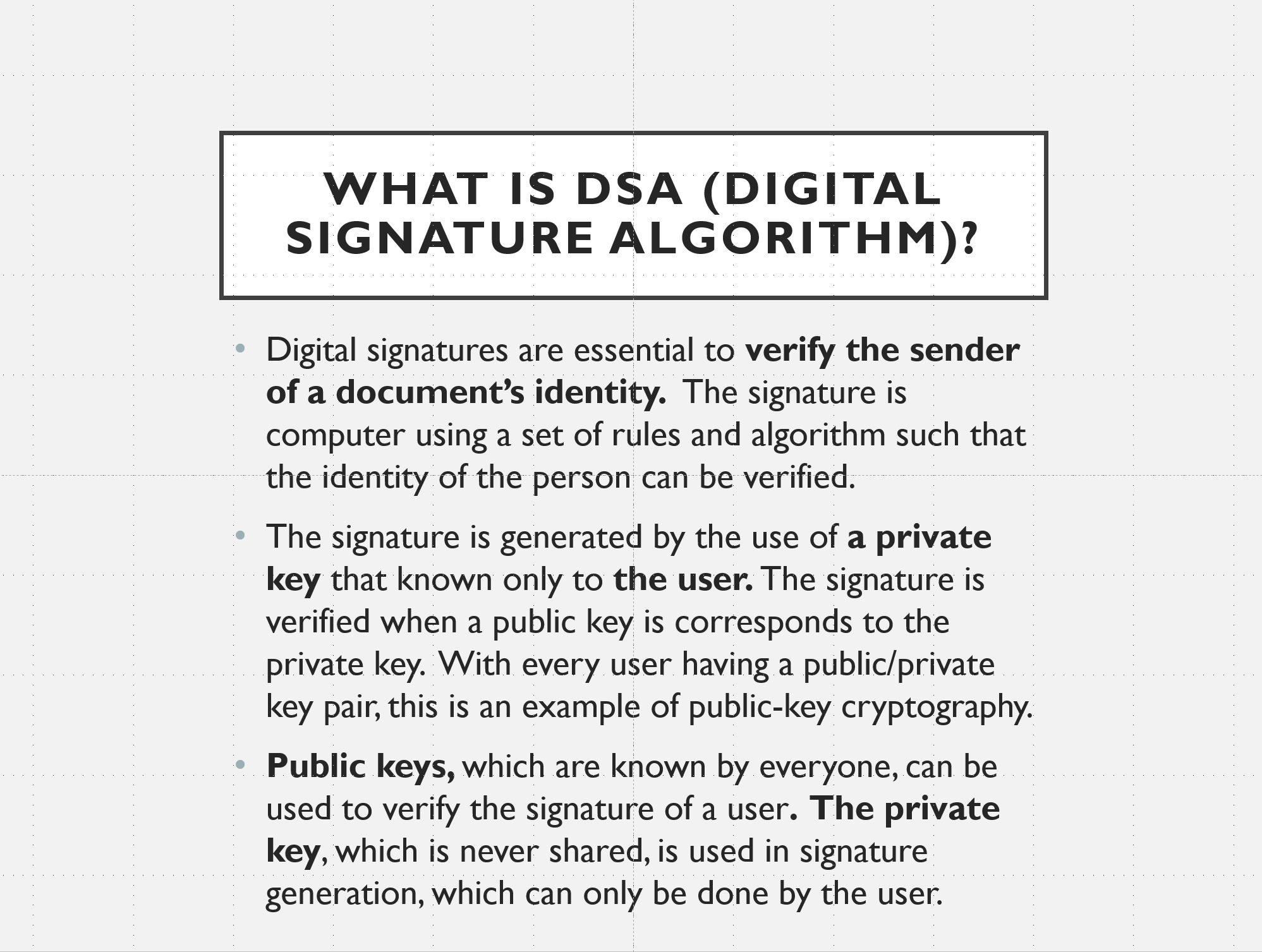
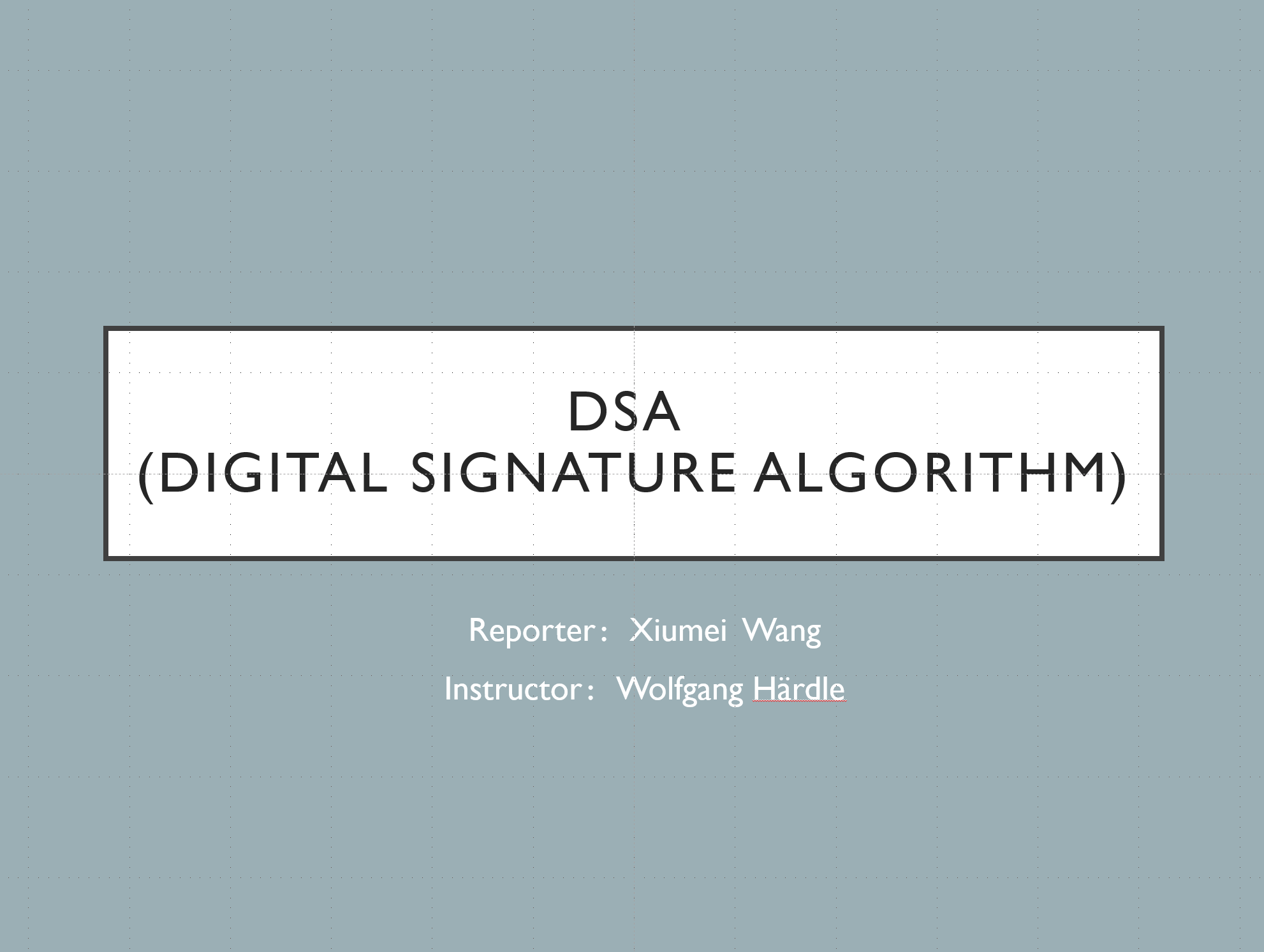
install.packages("digest")

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")

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



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

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))



**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?**

install.packages("caschrono ", repos="http://cran.us.r-project.org")

install.packages("TTR ", repos="http://cran.us.r-project.org")

install.packages("fGarch ", repos="http://cran.us.r-project.org")

install.packages("rugarch ", repos="http://cran.us.r-project.org")

install.packages("forecast ", repos="http://cran.us.r-project.org")

install.packages("TSA ", repos="http://cran.us.r-project.org")

library(caschrono)

library(TTR)

library(fGarch)

library(rugarch)

library(forecast)

library(TSA)

**#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ARIMA model\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

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")

xy.acfb(r,numer=FALSE)

**#\*\*\*\*\*2)Parameter Estimation**

**#estimation of p and q**

a.fin2=arima(r,order=c(2,0,2))

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)

**#\*\*\*\*3)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.me an=TRUE, na.action=na.pass)

summary(g2) g3=garchFit(~garch(2,1),data=residuals(a.fin2),trace=FALSE,include.me an=TRUE, na.action=na.pass)

summary(g3) g4=garchFit(~garch(2,2),data=residuals(a.fin2),trace=FALSE,include.me an=TRUE, na.action=na.pass)

summary(g4)

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

**Unit 4 HW**

**4.1 Improve the R quantlets on GH (from CRIX directory on quantlet.de) and make excellent graphics that follow Fig 3,4,5,6 of the „Econometrics of “CRIX” paper.**

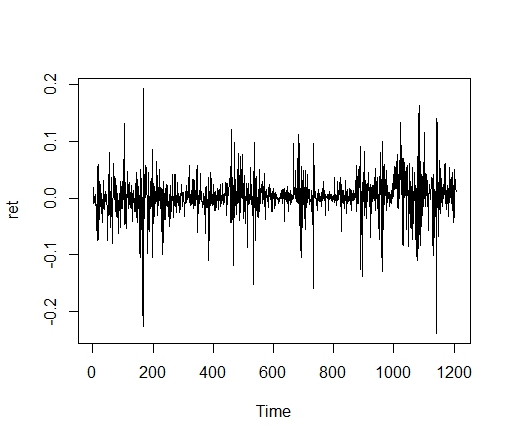
library(rjson)

json\_file = "http://crix.hu-berlin.de/data/crix.json"

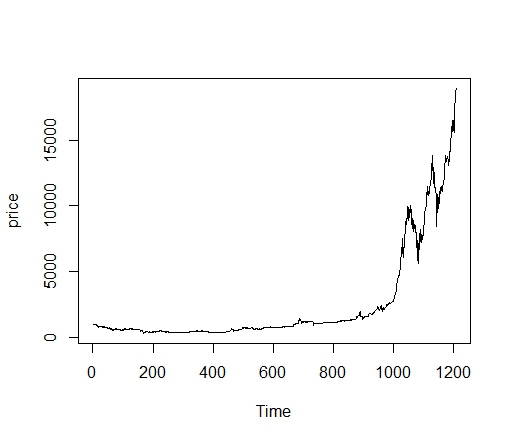
json\_data = fromJSON(file=json\_file)

crix\_data\_frame <- as.data.frame(json\_data)

w=crix\_data\_frame



**Figure 4**



**Figure 3**

dim(w)

n=dim(w)

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

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

data=t(w[1,a])

price=t(w[1,b])

**#figure3**

ts.plot(price)

**#figure4**

ret=diff(log(price))

ts.plot(ret)

**#figure5**

hist(ret, col = "grey", breaks = 40, freq = FALSE)

lines(density(ret), lwd = 2)

par(mfrow = c(1, 2))

**# histogram of returns**

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

lines(density(ret), lwd = 2)

mu = mean(ret)

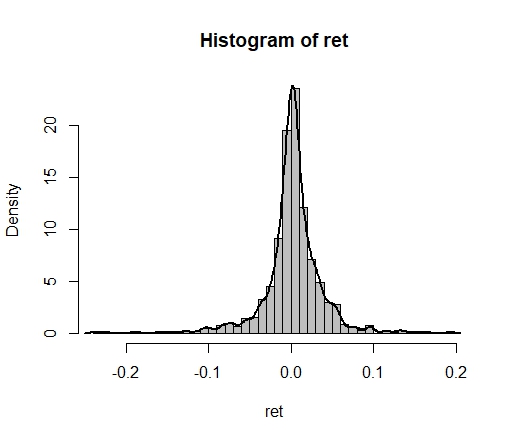
sigma = sd(ret)

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

curve(dnorm(x, mean = mean(ret), sd = sd(ret)), add = TRUE, col = "red",

lwd = 2)

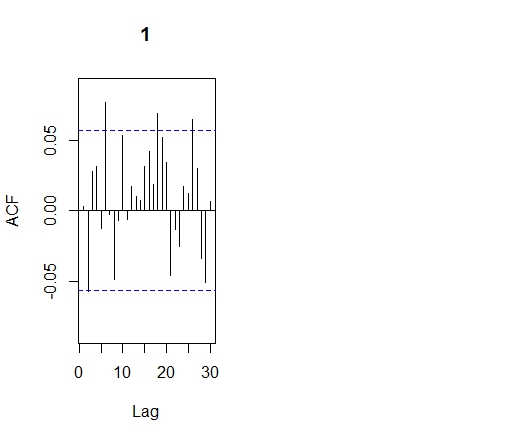
**# qq-plot**



**Figure 5**

qqnorm(ret)

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

**#figure6**

library(forecast)

library(tseries)

Acf(ret)

**4.2 Make your R code perfect as in the R examples on quantlet.de i.e. make sure that the code is “time independent” by using actual dimensions of the data that you are collecting from crix.hu-berlin.de Recreate Fig 7 from “Econometrics of CRIX”.**

**4.3 Redo as many ﬁgures as you can.**

**(all this to be done on perfect PPTX slides)**

**Answer 4.2 & 4.3**

#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)

# 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, 1200), ylab = "log return", xlab = "days",

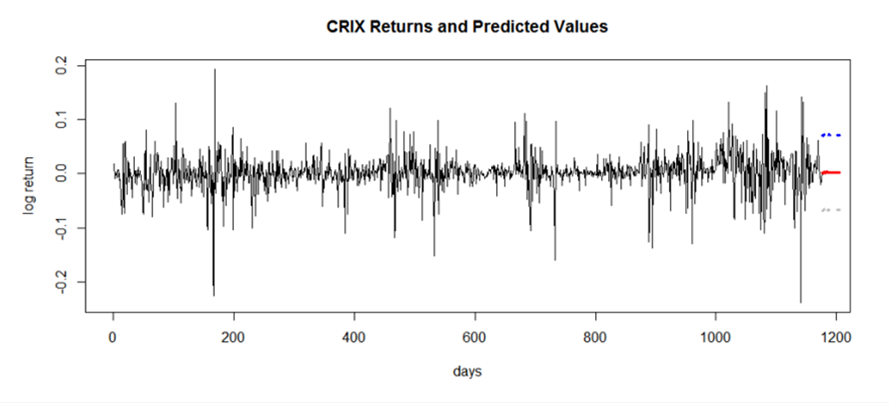
lwd = 1.5，mian=”CRIX Returns and Predicted Values”))

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)

**Figure 5**



**Unit 5 HW**

**5.1 Do a word cloud for Shakespeare’s dramas. Romeo and Julia, Julius Caesar, Hamlet.**

##Wordcloud##

rm(list = ls())

#install.packages("RCurl")

#install.packages("XML")

library(RCurl)

library(XML)

url1 = "http://shakespeare.mit.edu/romeo\_juliet/full.html"

url2 = "http://shakespeare.mit.edu/julius\_caesar/full.html"

url3 = "http://shakespeare.mit.edu/hamlet/full.html"

html1 = readLines(url1, encoding = "UTF-8")

html2 = readLines(url2, encoding = "UTF-8")

html3 = readLines(url3, encoding = "UTF-8")

html1 = htmlParse(html1, encoding = "UTF-8")

html2 = htmlParse(html2, encoding = "UTF-8")

html3 = htmlParse(html3, encoding = "UTF-8")

#install.packages("bitops")

#install.packages("stringr")

library(bitops)

library(stringr)

abs1 = lapply(url1, FUN = function(x) htmlParse(x, encoding = "Latin-1"))

abs2 = lapply(url2, FUN = function(x) htmlParse(x, encoding = "Latin-1"))

abs3 = lapply(url3, 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)

}

cleantxt1 = lapply(abs1,clean\_txt)

cleantxt2 = lapply(abs2,clean\_txt)

cleantxt3 = lapply(abs3,clean\_txt)

vec\_abs1 = unlist(cleantxt1)

vec\_abs2 = unlist(cleantxt2)

vec\_abs3 = unlist(cleantxt3)

###Text Mining

#install.packages("tm")

#install.packages("SnowballC")

library(tm)

library(SnowballC)

abs1 = Corpus(VectorSource(vec\_abs1))

abs2 = Corpus(VectorSource(vec\_abs2))

abs3 = Corpus(VectorSource(vec\_abs3))

abs\_dtm1 = DocumentTermMatrix(abs1, control = list(

stemming = TRUE, stopwords = TRUE, minWordLength = 3,

removeNumbers = TRUE, removePunctuation = TRUE))

abs\_dtm2 = DocumentTermMatrix(abs2, control = list(

stemming = TRUE, stopwords = TRUE, minWordLength = 3,

removeNumbers = TRUE, removePunctuation = TRUE))

abs\_dtm3 = DocumentTermMatrix(abs3, control = list(

stemming = TRUE, stopwords = TRUE, minWordLength = 3,

removeNumbers = TRUE, removePunctuation = TRUE))

##WordCloud

#instal.packages("ggplot2")

#install.packages("wordcloud2")

library(ggplot2)

library(wordcloud2)

freq1 = colSums(as.matrix(abs\_dtm1))

freq2 = colSums(as.matrix(abs\_dtm2))

freq3 = colSums(as.matrix(abs\_dtm3))

wf1 = data.frame(word=names(freq1[freq1>5]), freq=freq1[freq1>5])

wf2 = data.frame(word=names(freq2[freq2>5]), freq=freq2[freq2>5])

wf3 = data.frame(word=names(freq3[freq3>5]), freq=freq3[freq3>5])

#Romeo and Juliet

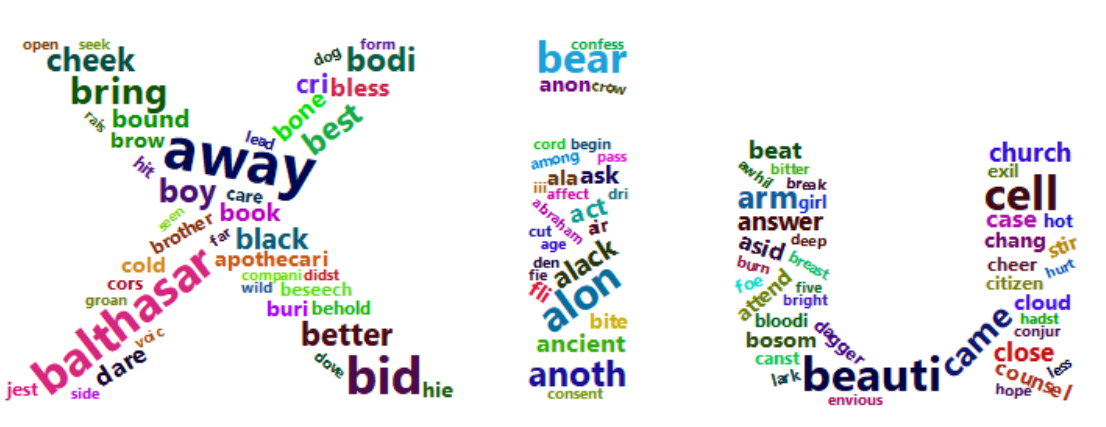
letterCloud(wf1,word="Xiu",size = 3)

#Julius Caeser

letterCloud(wf1,word="mei",size = 3)

#Hamlet

letterCloud(wf1,word="Wang",size = 3)





**5.2 Calculate the histogram of words.**

#Question 2

#Romeo and Juliet

wf1 <- wf1[order(-wf1$freq),]

wf1 <- wf1[c(1:20),]

p1 = ggplot(subset(wf1, freq > 5), aes(word, freq),color="red")

p1 = p1 + geom\_bar(stat = "identity",color="red")

p1 = p1 + theme(axis.text.x = element\_text(angle = 45, hjust = 1),color="red")

p1

#Julius Caeser

wf2 <- wf2[order(-wf2$freq),]

wf2 <- wf2[c(1:20),]

p2 = ggplot(subset(wf2, freq > 5), aes(word, freq),color="red")

p2 = p2 + geom\_bar(stat = "identity",color="red")

p2 = p2 + theme(axis.text.x = element\_text(angle = 45, hjust = 1),color="red")

p2

#Hamlet

wf3 <- wf3[order(-wf3$freq),]

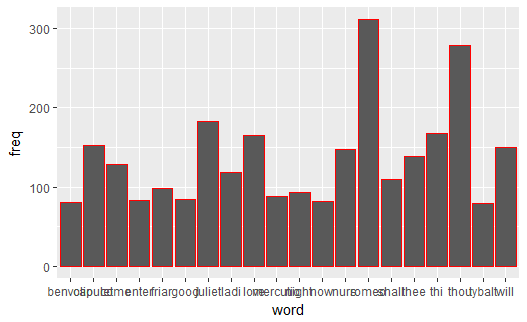
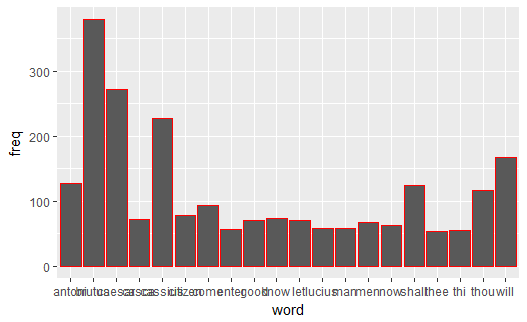
wf3 <- wf3[c(1:20),]

p3 = ggplot(subset(wf3, freq > 5), aes(word, freq),color="red")

p3 = p3 + geom\_bar(stat = "identity",color="red")

p3 = p3 + theme(axis.text.x = element\_text(angle = 45, hjust = 1),color="red")

p3

**5.3 Map the Shakespeare words into a dictionary to check its sentiment.**