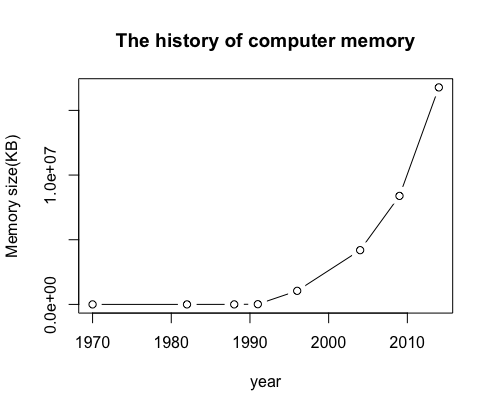
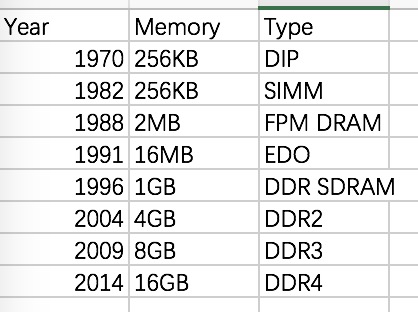
**Final Exam for Big Data and Internet Finance**

**Ran Cheng 27720161153019**

**HW Unit 1**

**Q1. Find the data of PC memory and plot it.**

****

**Q2. Learn and introduce logistic regression.**

Logistic regression is a regression model where the dependent variable  is categorical.

It was developed by statistician David Cox in 1958. It allows one to say that the presence of a risk factor increases the odds of a given outcome by a specific factor.

For example, if we want to calculate the probability of passing an exam versus hours of study, we can choose a group of 20 students spend between 0 and 6 hours studying for an exam. How does the number of hours spent studying affect the probability that the student will pass the exam?

In this regression, the dependent variable pass/fail represented by "1" and "0" are not cardinal numbers.

**Q3. Create your own github account.**

My github account name is RanCheng019.

**HW Unit 2**

**Q1. 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**

Codes:

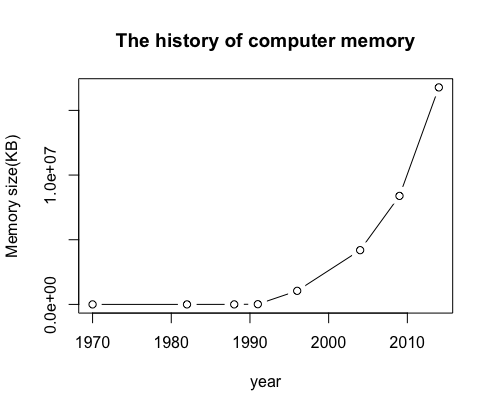
year<-c(1970,1982,1988,1991,1996,2004,2009,2014)

memory\_size<-c(256,256,2\*1024,16\*1024,1\*1024\*1024,4\*1024\*1024,8\*1024\*1024,16\*1024\*1024)

plot(year,memory\_size,type = "b", col="black",main = "The history of computer memory",

xlab = "year",ylab = "Memory size(KB)")

barplot(memory\_size, xlab = "year",ylab ="The number of memory ")

****

**Q2. Suppose you observe that in n=1000 mails (in 1 week) you have about 2 scams. Use the LvB /Possion pdf to calculate that you have 6 scam emails in 2 weeks. In Scammy land you have 5 scams on average, what is the probability to have no scam mail.**

Codes:

lambda=2

x=seq(0:6)

P<-data.frame(dpois(x,lambda))

lambda=5

x=0

dpois(x,lambda)

**HW Unit 3**

**Q1. 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.**

Codes:

#install.packages("digest")

library(digest)

digest("I learn a lot from this class when I am proper listening to the professor","sha1")

digest("I do not learn a lot from this class when I am absent and playing on my Iphone", "sha512")

**Q2. Make 3-5 slides (in PPTX) on the DSA (Digital Signature Algorithms).**

Digital Signature Algorithms is a Federal Information Processing Standard for digital signatures. It is a variant of the ElGamal signature scheme.

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. There are four revisions to the initial specification to DSA: FIPS 186-1 in 1996, FIPS 186-2 in 2000, FIPS 186-3 in 2009, FIPS 186-4 in 2013.

The key generation of DSA 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.

**Q3. Make slides with R code where you create a JSON data set that you save and read again.**

1. Create a JSON data set

Codes:

library(rjsonio)

Num <- [1:5]

Name <- c(“a”, ”b”, ”c”, “d”, “e”)

data <- as.matrix(data.frame(Num,Name))

cat(toJSON(data))

# Note: JSON data is a key-value pairs list.

1. Read the JSON data set

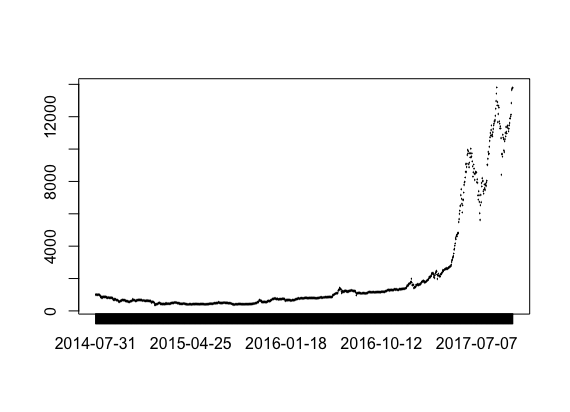
Codes:

library("rjson")

json\_data = fromJSON(file=data)

# We can use function ‘fromJSON’ to read the JSON data set

**Q4. 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?**



**Figure 1: Plot of CRIX Index**

Codes:

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)

crix\_data\_frame1 = as.vector(json\_data)

for(i in 1:1174){

date[i] = crix\_data\_frame1[[i]][1]

price[i] = crix\_data\_frame1[[i]][2]

}

date1 = as.vector(date)

price1 = as.numeric(price)

data1 = data.frame(date1,price1)

data2 = data.frame(t(as.vector(data1[1:1174])),t(as.vector(data1[-1:-1174])))

names(data2) <- c("date","price")

plot(x = data2$date, y = data2$price)

library(tseries)

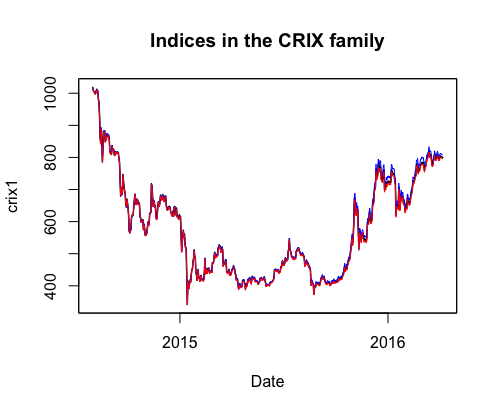
adf.test(time\_series)

# Since p-value greater than printed p-value,we can't reject the hypothesis#

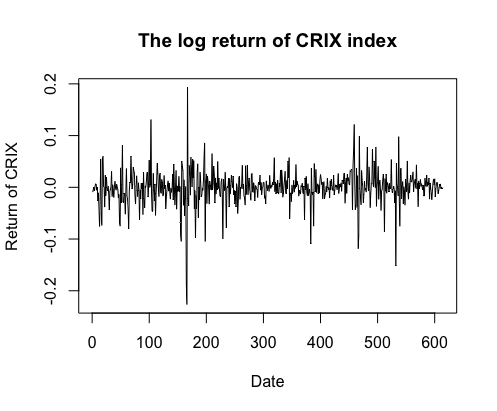
# the time series(time\_series) is not stationary#

**HW Unit 4**

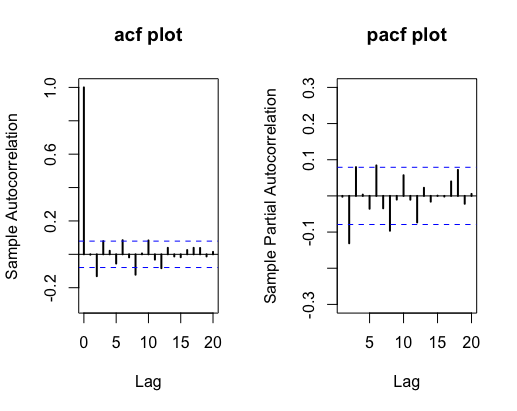
**Q1. 1.improve the R quantlets on GH (from CRIX directory on quantlet.de) and make excellent graphics that follow Fig 3,4,5,6 of the ”Econometrics of CRIX“ paper.**

****

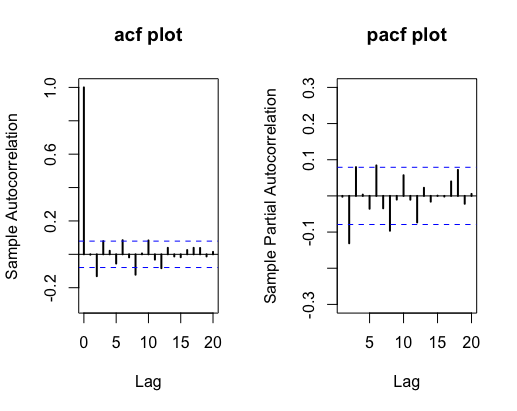
**Figure 3: The daily value of indices in the CRIX family**



**Figure 4: The log returns of CRIX index**

****

**Figure 5: Histogram and QQ plot of CRIX returns**



**Figure 6: The sample ACF and PACF of CRIX returns**

Codes:

rm(list = ls(all = TRUE))

graphics.off()

# install and load packages

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

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

install.packages(x)

})

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

# load dataset

load(file.choose())

load(file.choose())

load(file.choose())

# three indices return

ecrix1 = zoo(ecrix, order.by = index(crix1))

efcrix1 = zoo(efcrix, order.by = index(crix1))

# plot with different x-axis scales with zoo

my.panel <- function(x, ...) {

lines(x, ...)

lines(ecrix1, col = "blue")

lines(efcrix1, col = "red")

}

plot.zoo(crix1, plot.type = "multiple", type = "l", lwd = 1.5, panel = my.panel,

main = "Indices in the CRIX family", xlab = "Date")

# plot of crix

# plot(as.xts(crix), type="l", auto.grid=FALSE, main = NA)

plot(crix1, ylab = "Price of CRIX", xlab = "Date")

# plot of crix return

ret = diff(log(crix1))

# plot(as.xts(ret), type="l", auto.grid=FALSE, main = NA)

plot(ret, ylab = "Return of CRIX", xlab = "Date")

# stationary test

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

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

par(mfrow = c(1, 2))

# histogram of returns

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

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

qqnorm(ret)

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

# acf plot

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

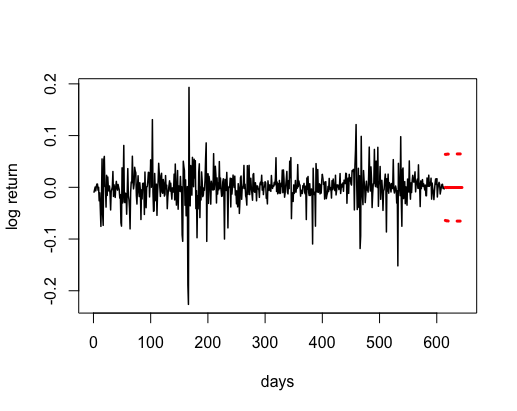
lwd = 2, ylim = c(-0.3, 1))

# pacf plot

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

main = "pacf plot", ylim = c(-0.3, 0.3), lwd = 2)

**Q2. 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“.**

****

**Figure 7: CRIX returns and predicted values.**

Codes:

# arima model

par(mfrow = c(1, 1))

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))

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

predict\_num = 30

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

crpre = predict(fit202, n.ahead = predict\_num)

dates = seq(as.Date("02/08/2014", format = "%d/%m/%Y"), by = "days", length = length(ret))

plot(ret, type = "l", xlim = c(0, length(ret)+predict\_num), ylab = "log return", xlab = "days",

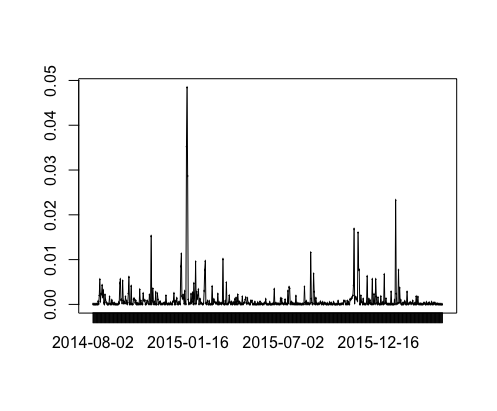
lwd = 1.5, col = "black")

lines(crpre$pred, col = "red", lwd = 3)

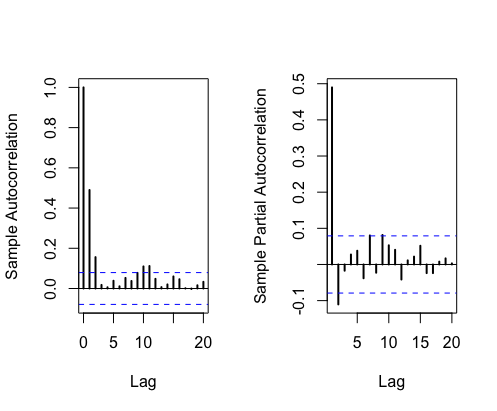
lines(crpre$pred + 2 \* crpre$se, col = "red", lty = 3, lwd = 3)

lines(crpre$pred - 2 \* crpre$se, col = "red", lty = 3, lwd = 3)

**Q3. Redo as many ﬁgures as you can.**



**Figure 8: The squared ARIMA(2,0,2) residuals of CRIX returns.**

****

**Figure 9: The ACF and PACF of squared ARIMA(2,0,2) residuals**

Code:

rm(list = ls(all = TRUE))

graphics.off()

# install and load packages

libraries = c("tseries")

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

install.packages(x)

})

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

# please change your working directory

setwd()

load(file.choose())

Pr = as.numeric(crix)

Da = factor(date1)

crx = data.frame(Da, Pr)

# plot of crix return

ret = diff(log(crx$Pr))

Dare = factor(date1[-1])

retts = data.frame(Dare, ret)

# 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)

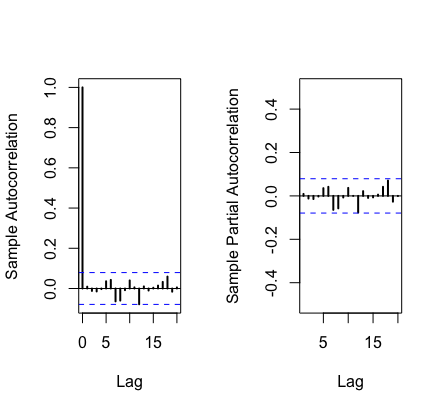
lines(tsres202$res2)

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

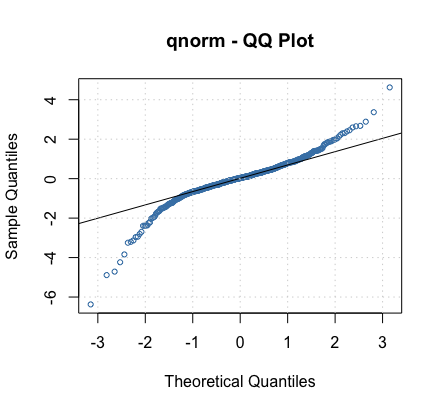
par(mfrow = c(1, 2))

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)



**Figure 10: The ACF and PACF of squared ARIMA(2,0,2) residuals**

****

**Figure 11: The QQ plots of model residuals of ARIMA-GARCH process.**

Codes:

rm(list = ls(all = TRUE))

graphics.off()

# install and load packages

libraries = c("forecast", "fGarch")

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

install.packages(x)

})

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

# load dataset

load(file.choose())

ret = diff(log(crix1))

# vol cluster

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

par(mfrow = c(1, 1))

res = fit202$residuals

res2 = fit202$residuals^2

# different garch model

fg11 = garchFit(data = res, data ~ garch(1, 1))

summary(fg11)

fg12 = garchFit(data = res, data ~ garch(1, 2))

summary(fg12)

fg21 = garchFit(data = res, data ~ garch(2, 1))

summary(fg21)

fg22 = garchFit(data = res, data ~ garch(2, 2))

summary(fg22)

# residual plot

reszo = zoo(fg11@residuals, order.by = index(crix1))

plot(reszo, ylab = NA, lwd = 2)

par(mfrow = c(1, 2))

fg11res2 = fg11@residuals

acfres2 = acf(fg11res2, lag.max = 20, ylab = "Sample Autocorrelation",

main = NA, lwd = 2)

pacfres2 = pacf(fg11res2, lag.max = 20, ylab = "Sample Partial Autocorrelation",

main = NA, lwd = 2, ylim = c(-0.5, 0.5))

fg12res2 = fg12@residuals

acfres2 = acf(fg12res2, lag.max = 20, ylab = "Sample Autocorrelation",

main = NA, lwd = 2)

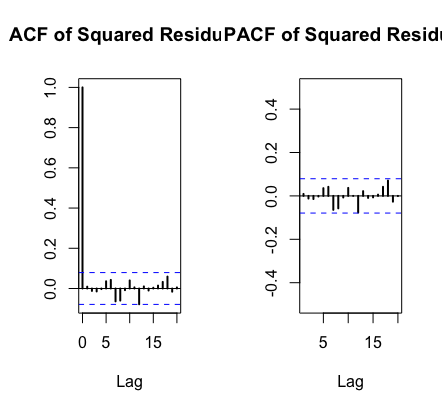
pacfres2 = pacf(fg12res2, lag.max = 20, ylab = "Sample Partial Autocorrelation",

main = NA, lwd = 2, ylim = c(-0.5, 0.5))

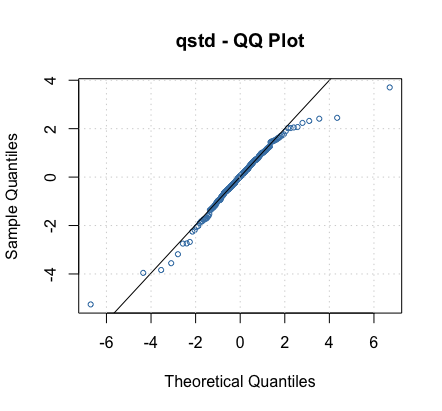
# qq plot

par(mfrow = c(1, 1))

plot(fg11, which = 13) #9,10,11,13



**Figure 12: The ACF and PACF plots for model residuals of ARIMA(2,0,2)- t-GARCH(1,1) process.**

****

**Figure 13: The QQ plots of model residuals of ARIMA-t-GARCH process.**

Codes:

fg11stu = garchFit(data = res, data ~ garch(1, 1), cond.dist = "std")

# different forecast with t-garch

# fg11stufore = predict(fg11stu, n.ahead = 30, plot=TRUE, mse='uncond', auto.grid=FALSE)

fg11stufore = predict(fg11stu, n.ahead = 30, plot = TRUE, cond.dist = "QMLE",

auto.grid = FALSE)

par(mfrow = c(1, 2))

stu.fg11res2 = fg11stu@residuals

# acf and pacf for t-garch

stu.acfres2 = acf(stu.fg11res2, ylab = NA, lag.max = 20, main = "ACF of Squared Residuals",

lwd = 2)

stu.pacfres2 = pacf(stu.fg11res2, lag.max = 20, main = "PACF of Squared Residuals",

lwd = 2, ylab = NA, ylim = c(-0.5, 0.5))

# ARIMA-t-GARCH qq plot

par(mfrow = c(1, 1))

plot(fg11stu, which = 13)

**HW Unit 5**

**Q1. do a word cloud for Shakesspeare’s dramas. Romeo and Julia, Julius Caesar, Hamlet.**

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)

#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("wordcloud")

library(ggplot2)

library(wordcloud)

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

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

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

wf1 = data.frame(word=names(freq1), freq=freq1)

wf2 = data.frame(word=names(freq2), freq=freq2)

wf3 = data.frame(word=names(freq3), freq=freq3)

#Romeo and Juliet

plot1 = ggplot(subset(wf1, freq>15), aes(word, freq1))

plot1 = plot1 + geom\_bar(stat="identity")

plot1 = plot1 + theme(axis.text.x=element\_text(angle=45, hjust=1))

plot1

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

dark2\_1 = brewer.pal(6, "Dark2")

wordcloud(names(freq1), freq1, max.words=100, rot.per=0.2, colors=dark2\_1)

#Julius Caeser

plot2 = ggplot(subset(wf2, freq>15), aes(word, freq2))

plot2 = plot2 + geom\_bar(stat="identity")

plot2 = plot2 + theme(axis.text.x=element\_text(angle=45, hjust=1))

plot2

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

dark2\_2 = brewer.pal(6, "Dark2")

wordcloud(names(freq2), freq2, max.words=100, rot.per=0.2, colors=dark2\_2)

#Hamlet

plot3 = ggplot(subset(wf3, freq>15), aes(word, freq3))

plot3 = plot3 + geom\_bar(stat="identity")

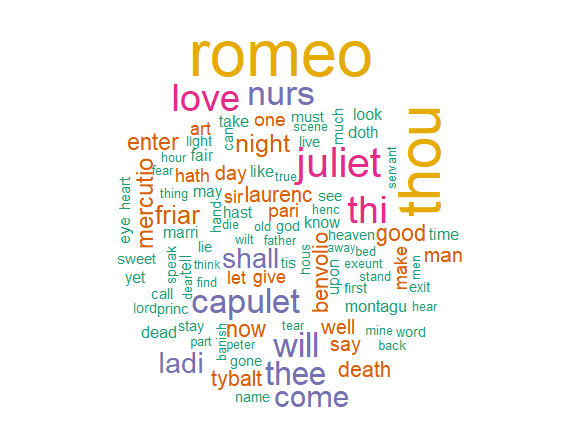
plot3 = plot3 + theme(axis.text.x=element\_text(angle=45, hjust=1))

plot3

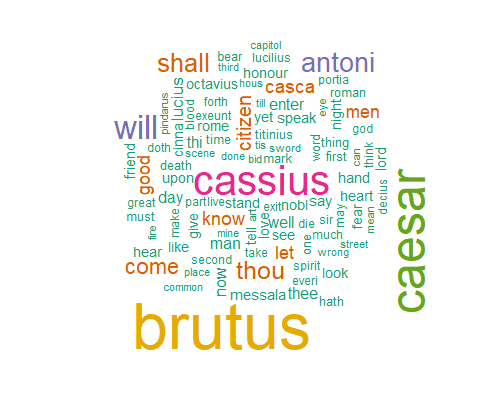
freq3 = colSums(as.matrix(abs\_dtm3))

dark2\_3 = brewer.pal(6, "Dark2")

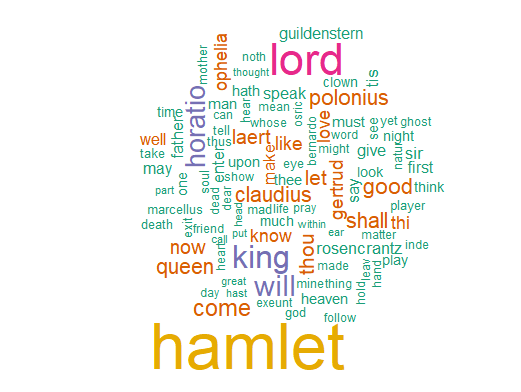
wordcloud(names(freq3), freq3, max.words=100, rot.per=0.2, colors=dark2\_3)



**Figure 14: Romeo and Julia**

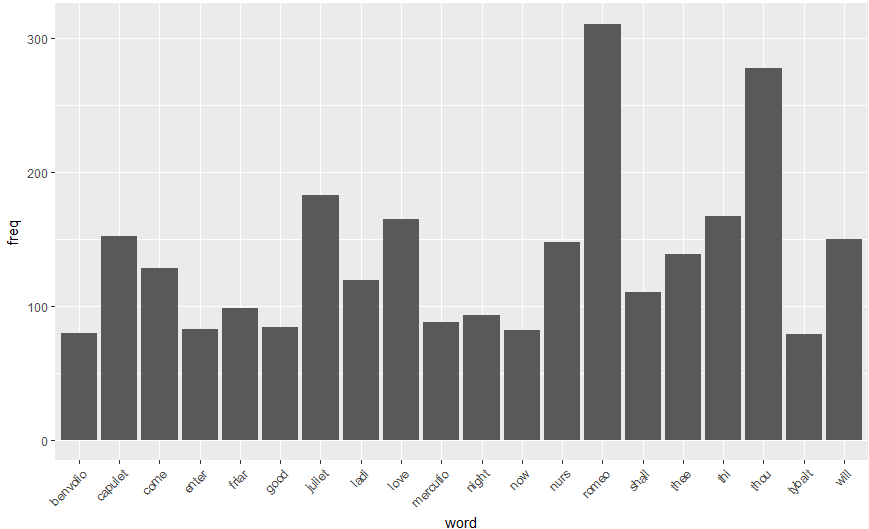


**Figure 15: Julius Caesar**

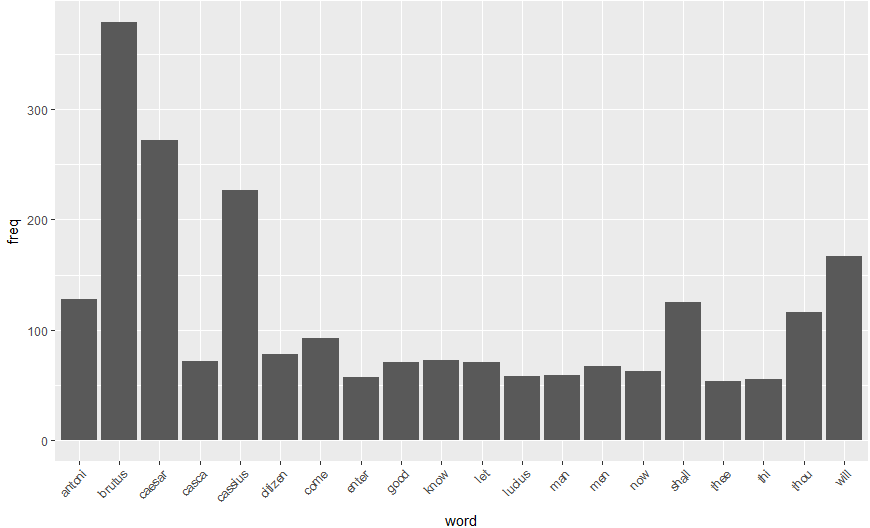


**Figure 16: Hamlet**

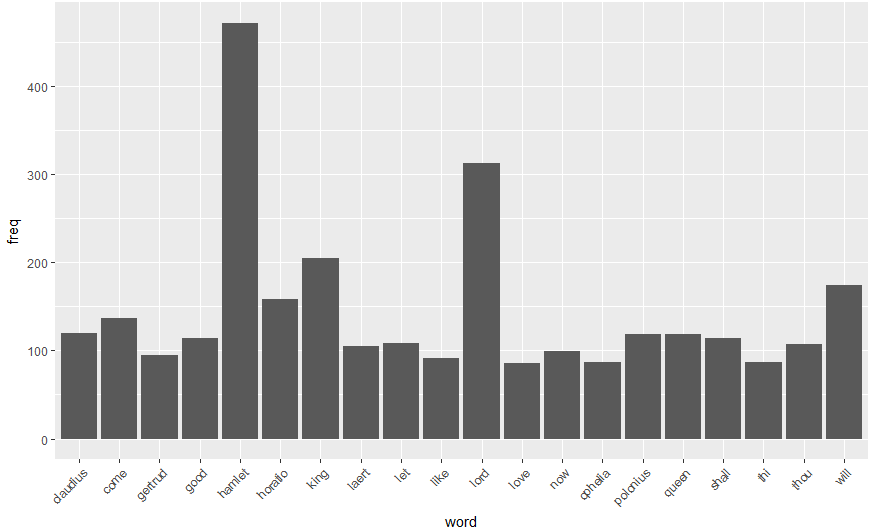
**Q2. calculate the histogram of words**



**Figure 17: Romeo and Julia**



**Figure 18: Julius Caesar**



**Figure 19: Hamlet**