Final

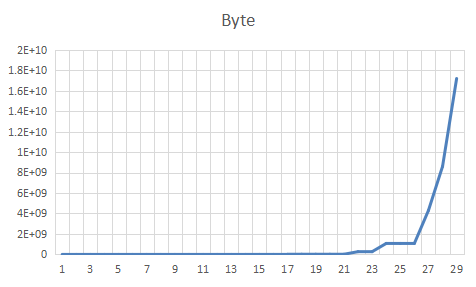
Lu Yao

15620161152294

**HW Unit 1**

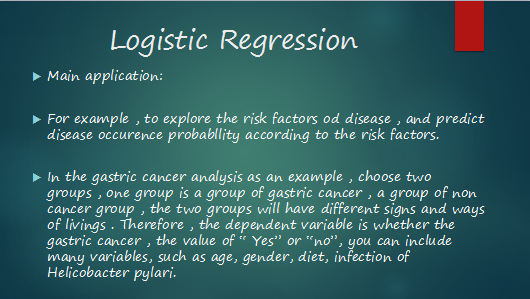
**Q1 Calculate the increase of memory of PCs over the last 30 years and check whether the FMRI analysis could have been done 20 years .**

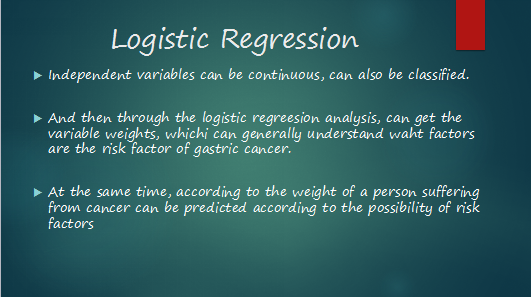
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **The Trend of Storage** | | | | | |
| Year | Byte | Year | Byte | Year | Byte |
| 1970 | 262144 | 1980 | 262144 | 1995 | 16777216 |
| 1971 | 262144 | 1981 | 262144 | 1996 | 268435456 |
| 1972 | 262144 | 1982 | 262144 | 1997 | 268435456 |
| 1973 | 262144 | 1988 | 2097152 | 1998 | 1073741824 |
| 1974 | 262144 | 1989 | 2097152 | 1999 | 1073741825 |
| 1975 | 262144 | 1990 | 2097152 | 2000 | 1073741826 |
| 1976 | 262144 | 1991 | 16777216 | 2004 | 4294967296 |
| 1977 | 262144 | 1992 | 16777216 | 2009 | 8589934592 |
| 1978 | 262144 | 1993 | 16777216 | 2014 | 17179869184 |
| 1979 | 262144 | 1994 | 16777216 |  |  |



**Q2 prepare 2-5 slides explaining logistic regression.**







Q3. I have done it and my account is LuYao294.

**HW Unit 2**

**Q1**

R code:

setwd("C:/Users/Administrator/Desktop")

read.csv("data.csv")

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

**Q2**

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

**Q3**

> x=3

> lambda=2

> dpois(x,lambda)

[1] 0.180447

> x=0

> lambda=5

> dpois(x,lambda)

[1] 0.006737947

**HW Unit 3**

**Q1**

R code：

install.packages("digest")

library("digest")

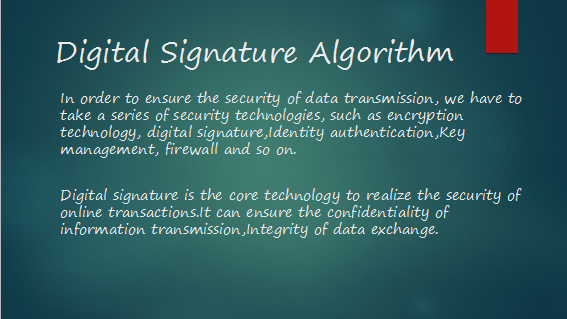
digest("I learn a lot from this class when I am proper listening to the professor", "sha256")

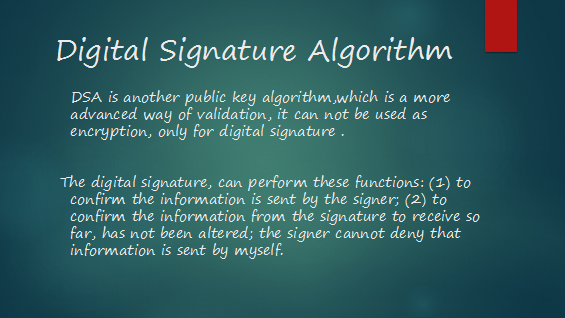
[1] "c16700de5a5c1961e279135f2be7dcf9c187cb6b21ac8032308c715e1ce9964c"

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

"2533d529768409d1c09d50451d9125fdbaa6e5fd4efdeb45c04e3c68bcb3a63e

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







Q3

R code:  
>install.packages("RJSONIO)  
>library(rjson)   
> bodylength<-c(188,152,201,165)  
>bodyfeature<-c("tall","short","very tall","normal" )  
> individualfeature<-data.frame(bodylength,bodyfeature)  
> data<-as.matrix( individualfeature)  
>cat(toJSON(data))

Q4

R code：

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\Administrator\Desktop/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)

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)

HW Unit 4

Q1

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

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

qqnorm(ret)

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

#figure6

library(forecast)

library(tseries)

Acf(ret)

Results output:

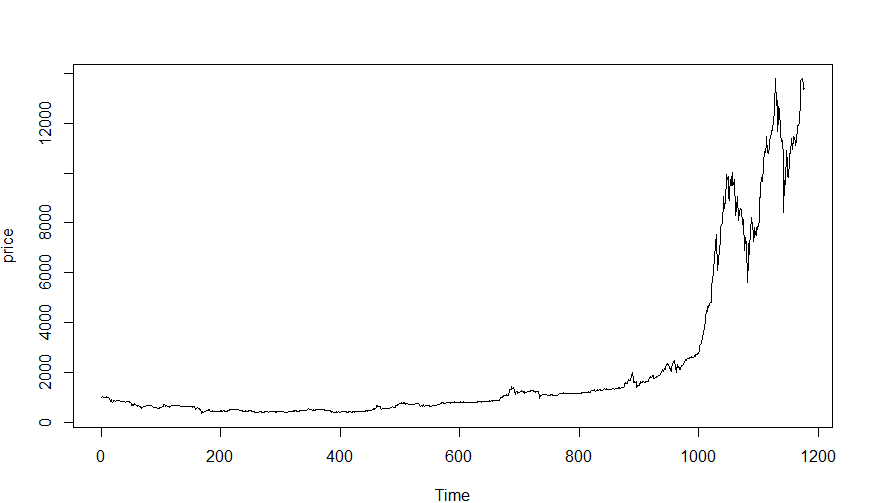


Figure 3

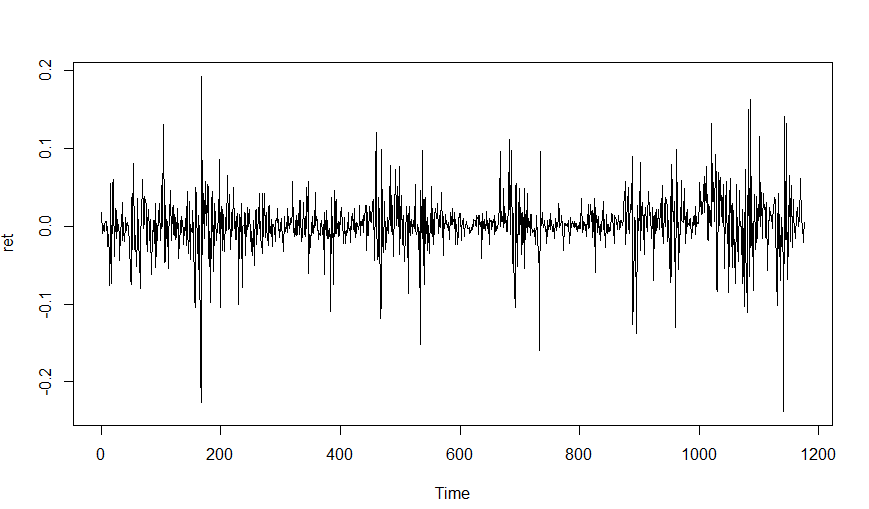
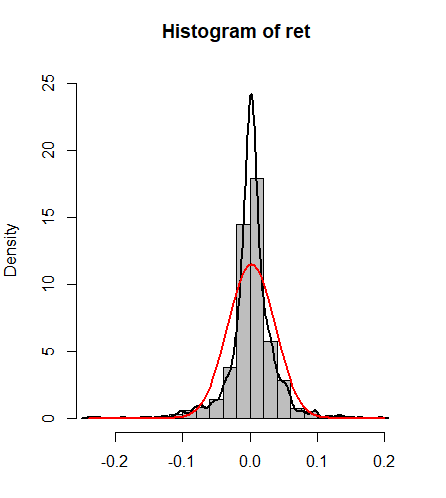


Figure 4



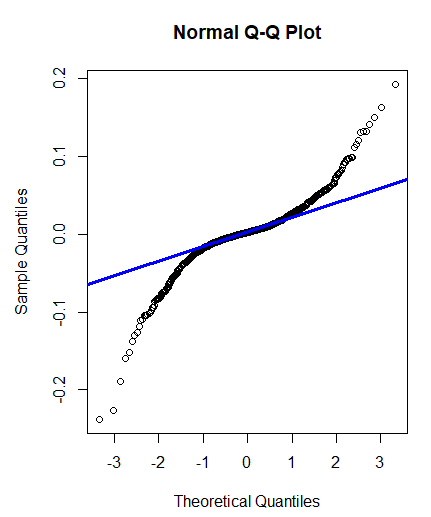


Figure 5

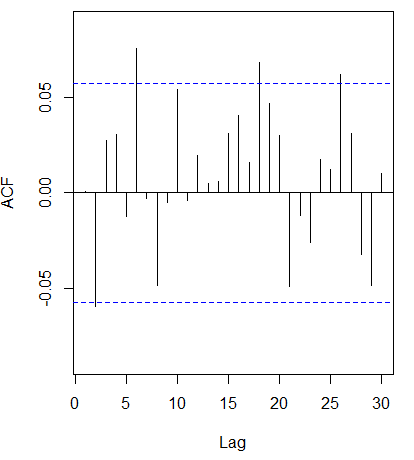


Figure 6

**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)

# 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)

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)

**Results output:**

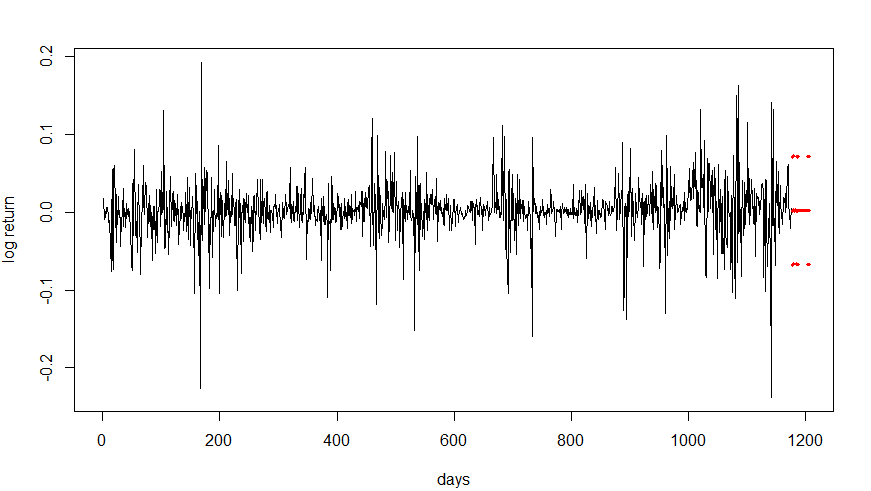


Figure 7

**HW Unit 5**

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

R Code:

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))

install.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)

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

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

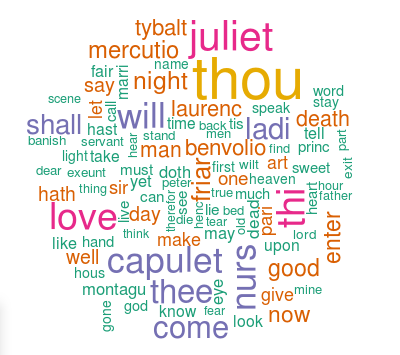
plot1 = plot1 + theme(axis.text.x=element\_text(angle=45, hjust=1))

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)



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

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

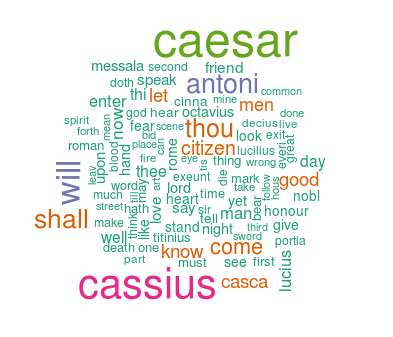
plot2 = plot2 + theme(axis.text.x=element\_text(angle=45, hjust=1))

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)



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

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

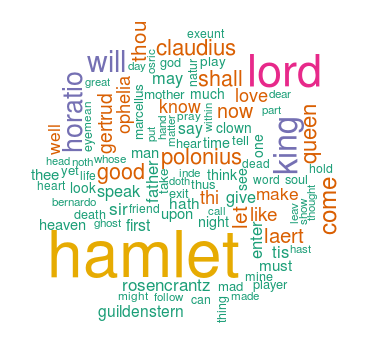
plot3 = plot3 + theme(axis.text.x=element\_text(angle=45, hjust=1))

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)



**Q2.calculate the histogram of words**

#Romeo and Juliet

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

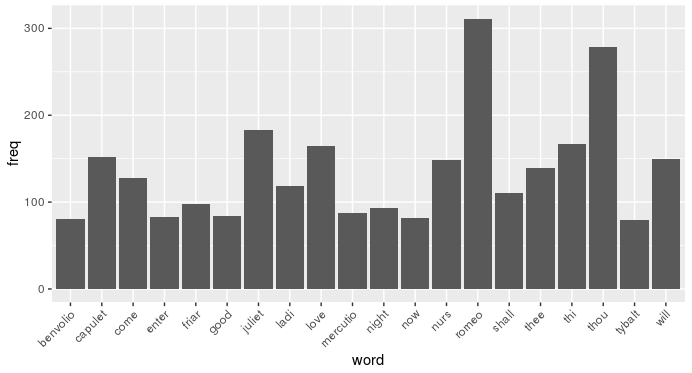
wf1 <- wf1[c(1:20),]

p1 = ggplot(subset(wf1, freq > 15), aes(word, freq))

p1 = p1 + geom\_bar(stat = "identity")

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

p1



#Julius Caeser

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

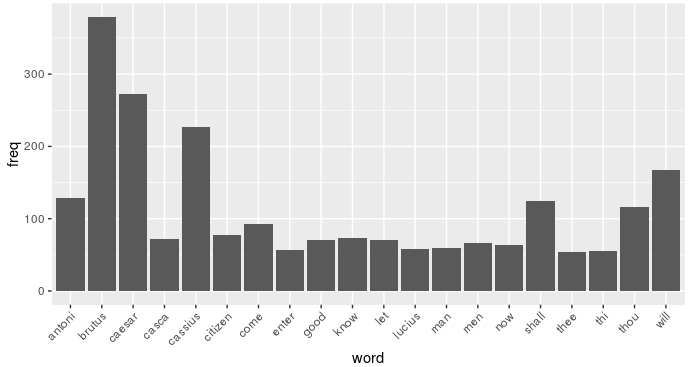
wf2 <- wf2[c(1:20),]

p2 = ggplot(subset(wf2, freq > 15), aes(word, freq))

p2 = p2 + geom\_bar(stat = "identity")

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

p2



#Hamlet

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

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

p3 = ggplot(subset(wf3, freq > 15), aes(word, freq))

p3 = p3 + geom\_bar(stat = "identity")

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

p3

