

Final EXAM

Aiqing Jiang 27720151153581

Homework 1

Question 1

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

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

For 20 years ago, it's impossible for us to perform FMRI analysis as the memory of computer is not large enough for us to handle the huge amount of data.

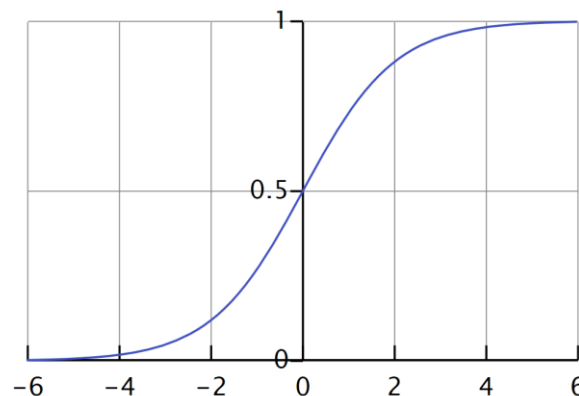
Question 2 Describe Logistic Regression

In statistics, logistic regression, or logit regression, or logit model is a regression model where the dependent variable (DV) is categorical.

Which means the dependent variable can be binary—where the output can take only two values, "0" and "1", represents outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysis in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression.

The logistic function is useful because it can take any real input t whereas the output always takes values between zero and one and hence is interpretable as a probability.

The logistic function $\sigma(t)$ is defined as follows: $\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$



Let us assume that t is a linear function of a single explanatory variable x , which means $t = \beta_0 + \beta_1 x$

And the logistic function can now be written as:

$$F(x) = 1 / (1 + e^{-(\beta_0 + \beta_1 x)})$$

Note that $F(x)$ is interpreted as the probability of the dependent variable equaling a "success" or "case" rather than a failure or non-case.

Logistic regression is used in various fields, including machine learning, most medical fields, and social sciences.

Logistic regression may be used to predict whether a patient has a given disease (e.g. diabetes; coronary heart disease), based on observed characteristics of the patient (age, sex, body mass index, results of various blood tests, etc.).

Another example might be to predict whether an American voter will vote Democratic or Republican, based on age, income, sex, race, state of residence, votes in previous elections, etc.

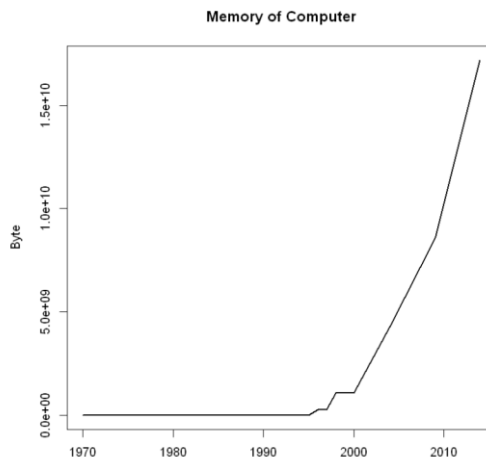
It is also used in marketing applications such as prediction of a customer's propensity to purchase a product or halt a subscription, etc. In economics it can be used to predict the likelihood of a person's choosing to be in the labor force, and a business application would be to predict the likelihood of a homeowner defaulting on a mortgage.

Homework 2

Question 1: Make an R quantlet to solve HW #1 from unit 1 with R and show it on Github (GH)

Code:

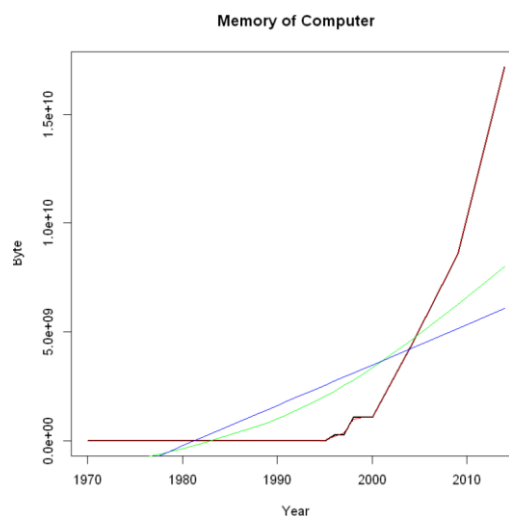
```
library(readr)
cpum <- read_csv("cpum.csv", col_names = TRUE)
par(mfrow = c(1, 1))
plot(cpum, type = "l", xlab = "Year", ylab = "Byte", lwd = 2, main = "Memory of Computer")
```



Question 2: Use R with B-spline code to solve HW#1, any comments?

Code:

```
plot(cpum, type = "l", xlab = "Year", ylab = "Byte", lwd = 2, main = "Memory of Computer")
splines.reg.l1 = smooth.spline(x = cpum$Year, y = cpum$Byte, spar = 0.2) # lambda = 0.2
splines.reg.l2 = smooth.spline(x = cpum$Year, y = cpum$Byte, spar = 1) # lambda = 1
splines.reg.l3 = smooth.spline(x = cpum$Year, y = cpum$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
```



Question 3: Suppose you observe that in $n=1000$ mails (in 1 week) you have about 2

scams. Use the 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

Code:

$\text{Lambda} = 4$

$x = 6$

$P1 = \exp(-\text{lambda}) * \text{lambda}^x / \text{factorial}(x)$

$P1 = 0.104195634567021$

$\text{lambda} = 5$

$x = 0$

$P2 = \exp(-\text{lambda}) * \text{lambda}^x / \text{factorial}(x)$

$P2 = 0.00673794699908547$

Homework 3

Question 1:

Code:

```
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")
```

For the first sentence, the hash number is

```
"c16700de5a5c1961e279135f2be7dcf9c187cb6b21ac8032308c715e1ce9964c"
```

For the second sentence, the hash number is

```
"2533d529768409d1c09d50451d9125fdbaa6e5fd4efdeb45c04e3c68bcb3a63e"
```

Question 2: Describe Digital Signature Algorithm

1. What is digital signature?

A digital signature is a mathematical scheme for demonstrating the authenticity of digital messages or documents.

A valid digital signature gives a recipient reason to believe that the message was created by a known sender, that the sender cannot deny having sent the message, and that the message was not altered in transit (integrity). Applications: software distribution, financial transactions, data storage etc.

2. Digital Signature Algorithm

The Digital Signature Algorithm (DSA) is a Federal Information Processing Standard for digital signatures. 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.

3. Steps of Digital Signature Algorithm

● Key generation

Key generation has two phases. The first phase is a choice of *algorithm parameters* which may be shared between different users of the system, while the second phase computes public and private keys for a single user.

● Signing

● Verifying

Question 3: json data write and save

Code:

```
library(jsonlite)
```

```
json <-
```

```
'[
  {"Name" : "Mario", "Age" : 32, "Occupation" : "Plumber"},
  {"Name" : "Peach", "Age" : 21, "Occupation" : "Princess"},
  {},
  {"Name" : "Bowser", "Occupation" : "Koopa"}
]'
```

```
data_frame <- fromJSON(json)
```

```
data_frame
```

Name	Age	Occupation
Mario	32	Plumber
Peach	21	Princess
NA	NA	NA
Bowser	NA	Koopa

```
data_frame$Ranking <- c(3, 1, 2, 4)
```

```
data_frame
```

Name	Age	Occupation	Ranking
Mario	32	Plumber	3
Peach	21	Princess	1
NA	NA	NA	2
Bowser	NA	Koopa	4

```
toJSON(data_frame, pretty=TRUE)
```

```
[
  {
    "Name": "Mario",
    "Age": 32,
    "Occupation": "Plumber",
    "Ranking": 3
  },
  {
    "Name": "Peach",
    "Age": 21,
    "Occupation": "Princess",
    "Ranking": 1
  },
  {
    "Ranking": 2
  },
  {
    "Name": "Bowser",
    "Occupation": "Koopa",
    "Ranking": 4
  }
]
```

```
write_json(json,path="C:/Users/Aiqing-Jiang/1.json")
```

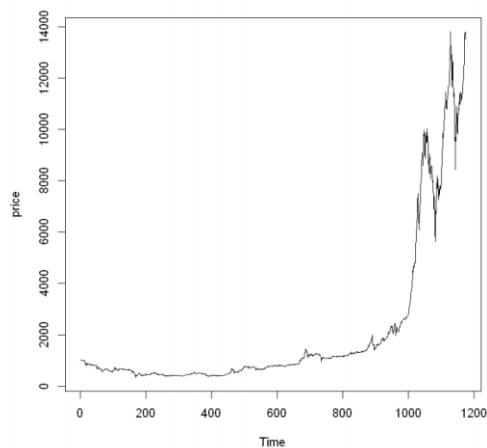
```
read_json(path="C:/Users/Aiqing-Jiang/1.json",simplifyVector = FALSE)
```

```
1. '[
  {"Name" : "Mario", "Age" : 32, "Occupation" : "Plumber"},
  {"Name" : "Peach", "Age" : 21, "Occupation" : "Princess"},
  {},
  {"Name" : "Bowser", "Occupation" : "Koopa"}
]'
```

Question 4: json data write and save

Code:

```
library(rjson)
json_file = "http://crux.hu-berlin.de/data/crux.json"
json_data = fromJSON(file=json_file)
crux_data_frame = as.data.frame(json_data)
a <- 1:1175
n <- 2*a
m <- n-1
a <- 1:1175
n <- 2*a
m <- n-1
ts.plot(price)
```



```
library(tseries)
```

```
adf.test(price)
```

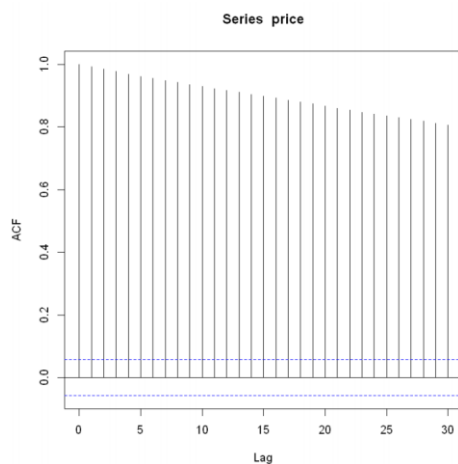
Augmented Dickey-Fuller Test

```
data: price
```

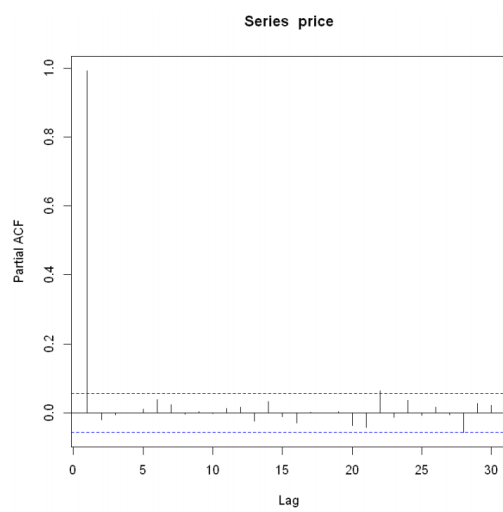
```
Dickey-Fuller = 0.47023, Lag order = 10, p-value = 0.99
```

```
alternative hypothesis: stationary
```

```
acf(price)
```



```
pacf(price)
```



```
library(forecast)
```

```
auto.arima(price)# ARIMA(5,2,0)
```

```
Series: price  
ARIMA(5,2,0)
```

```
Coefficients:
```

	ar1	ar2	ar3	ar4	ar5
	-0.8808	-0.7101	-0.5786	-0.4783	-0.2543
s.e.	0.0284	0.0359	0.0380	0.0362	0.0286

```
sigma^2 estimated as 32821: log likelihood=-7761.48  
AIC=15534.95 AICc=15535.02 BIC=15565.36
```


Homework 4

Code:

```
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
dim(x)
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])
crix=data.frame(date,price)
load("ecrix.RData")
load("efcrix.RData")
plot(ecrix, type = "l", col = "blue", xaxt = "n", main = " Daily value of indices in the CRIX
family", xlab = "Date", ylab = "Daily Value of Indices")
lines(efcrix, col = "black")
lines(price, col = "red")
lab=seq(1,n[2],365)
axis(1, at = lab, label = names(ecrix)[lab])
```

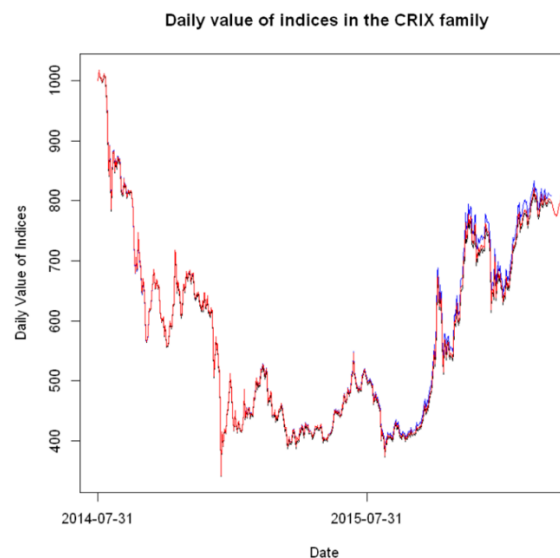


Figure3: Daily value of indices in the CRIX family

```
ret=diff(log(price))
plot(ret~as.Date(date[-1]), type="l", col="black", xlab="Date", ylab="Log return",
main="Log returns of crix index")
```

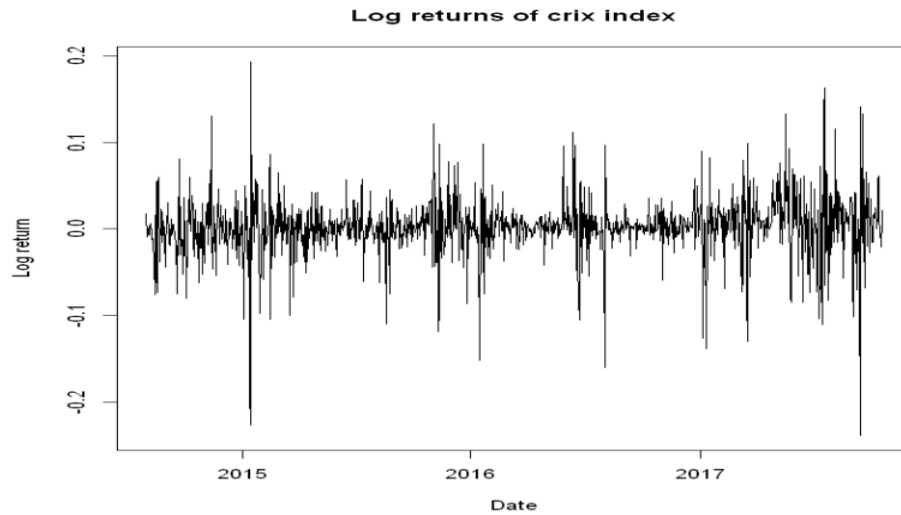


Figure 4: The log returns of CRIX index

```
hist(ret, col = "grey", breaks = 20, freq = FALSE, ylim = c(0, 25), xlab = "Return")
lines(density(ret), lwd = 2)
x = seq(-4, 4, length = 100)
curve(dnorm(x, mean = mean(ret), sd = sd(ret)), add=TRUE, col = "darkblue", lwd = 2)
```

Histogram of ret

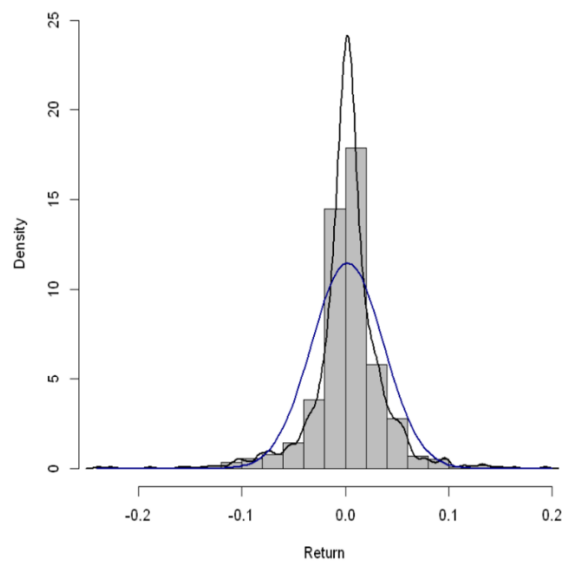


Figure 5: Histogram and QQ plot of CRIX returns

```
qqnorm(ret)
qqline(ret, col = "blue", lwd = 4)
```

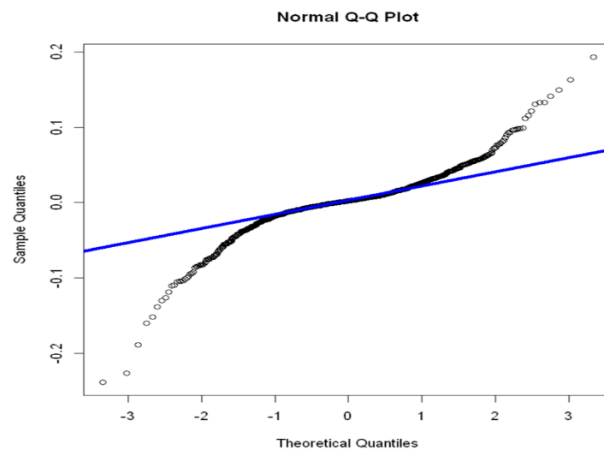


Figure 5: Histogram and QQ plot of CRIX returns

```
par(mfrow = c(2, 1))
libraries = c("zoo", "tseries")
autocorr = acf(ret, lag.max = 20, ylab = "Sample Autocorrelation", main = "ACF", lwd =
2, ylim = c(-0.3, 1))
autopcorr = pacf(ret, lag.max = 20, ylab = "Sample Partial Autocorrelation", main =
"PACF", ylim = c(-0.3, 0.3), lwd = 2)
```

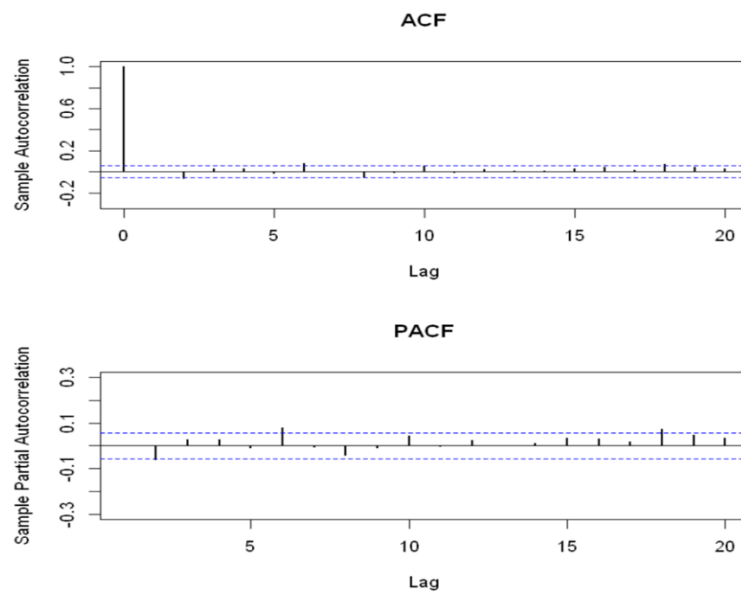


Figure 6: The sample ACF and PACF of CRIX returns

```

library(TTR)
library(TSA)
library(caschnono)
Library(forecast)
fit = arima(ret, order = c(2, 0, 2))
tsdiag(fit)

```

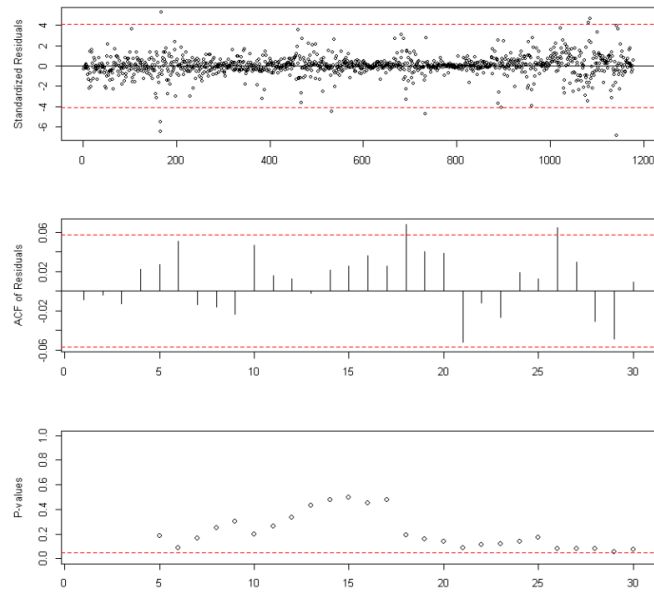


Figure 7:Diagnostic Checking

```

crix_pre = predict(fit, n.ahead = 30)
plot(ret, type = "l", ylab = "Log return", xlab = "Date",
lwd = 1, main = "CRIX returns and predicted values")
lines(crix_pre$pred, col = "red", lwd = 1)
lines(crix_pre$pred + 2 * crix_pre$se, col = "red", lty = 3, lwd = 1)
lines(crix_pre$pred - 2 * crix_pre$se, col = "red", lty = 3, lwd = 1)

```

CRIX returns and predicted values

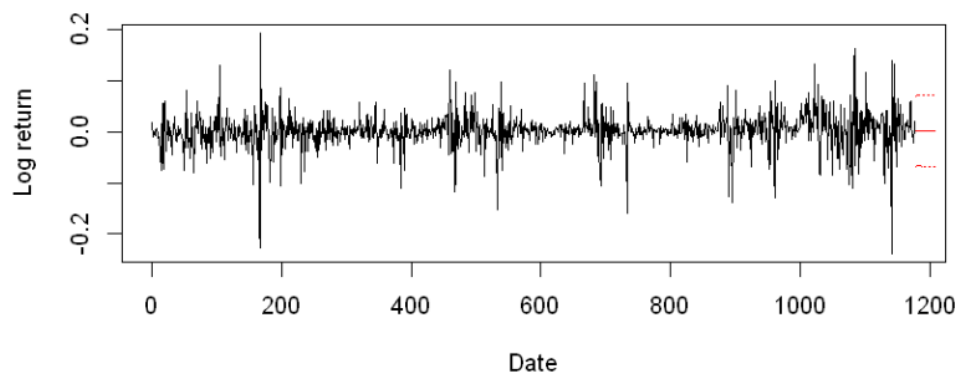


Figure 7:Diagnostic Checking

Homework 5

Code:

```
library(RCurl)
library(XML)
library(stringr)

url      = paste("http://shakespeare.mit.edu/romeo_juliet/full.html")
#url     = paste("http://shakespeare.mit.edu/julius_caesar/full.html")
#url     = paste("http://shakespeare.mit.edu/hamlet/full.html")
abs      = lapply(url, FUN = function(x) htmlParse(x, encoding = "ANSI"))

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)

library(NLP)
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)
findFreqTerms(abs_dtm, 3)
removeSparseTerms(abs_dtm, 0.5)
inspect(removeSparseTerms(abs_dtm, 0.5))

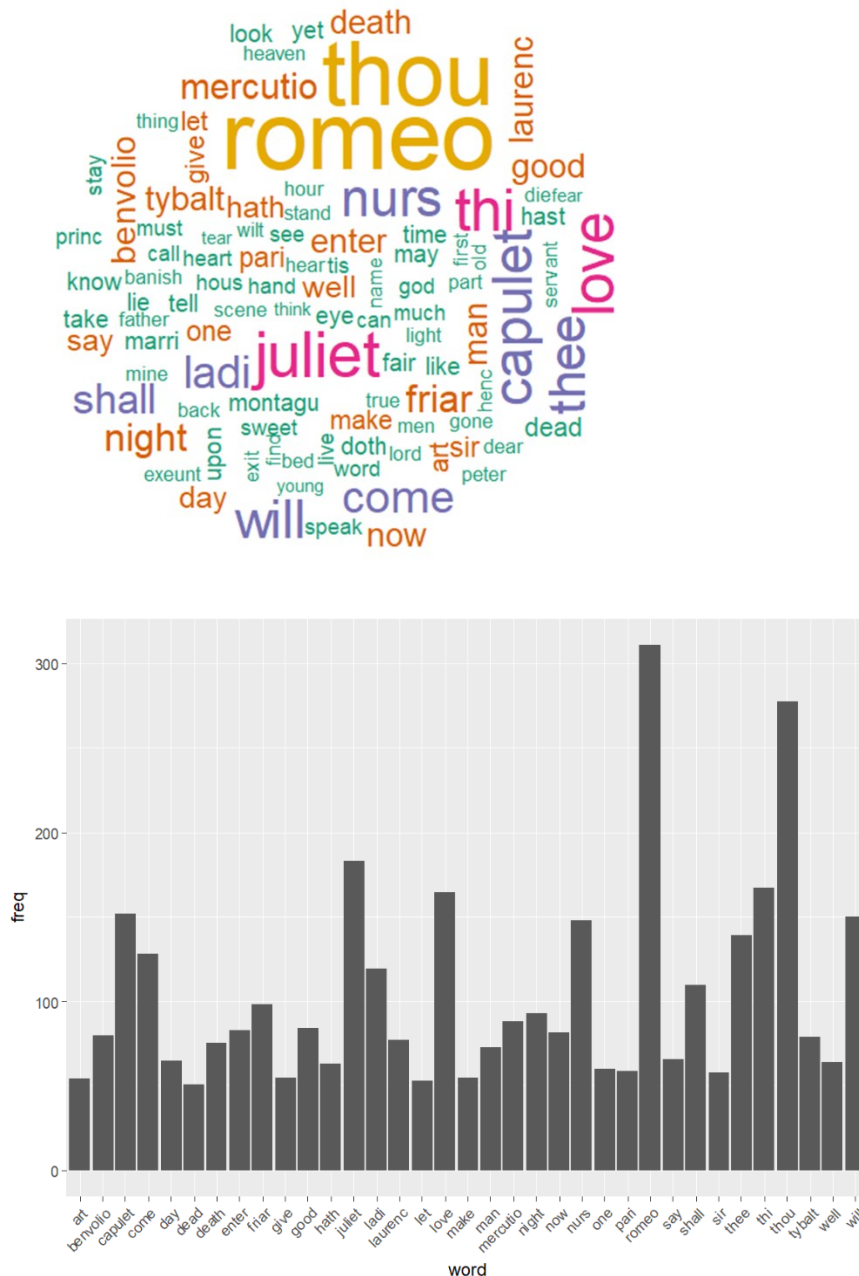
library(ggplot2)
library(wordcloud)
freq = colSums(as.matrix(abs_dtm))
wf    = data.frame(word=names(freq), freq=freq)
plot = ggplot(subset(wf, freq>50), aes(word, freq))
plot = plot + geom_bar(stat="identity")
```

```

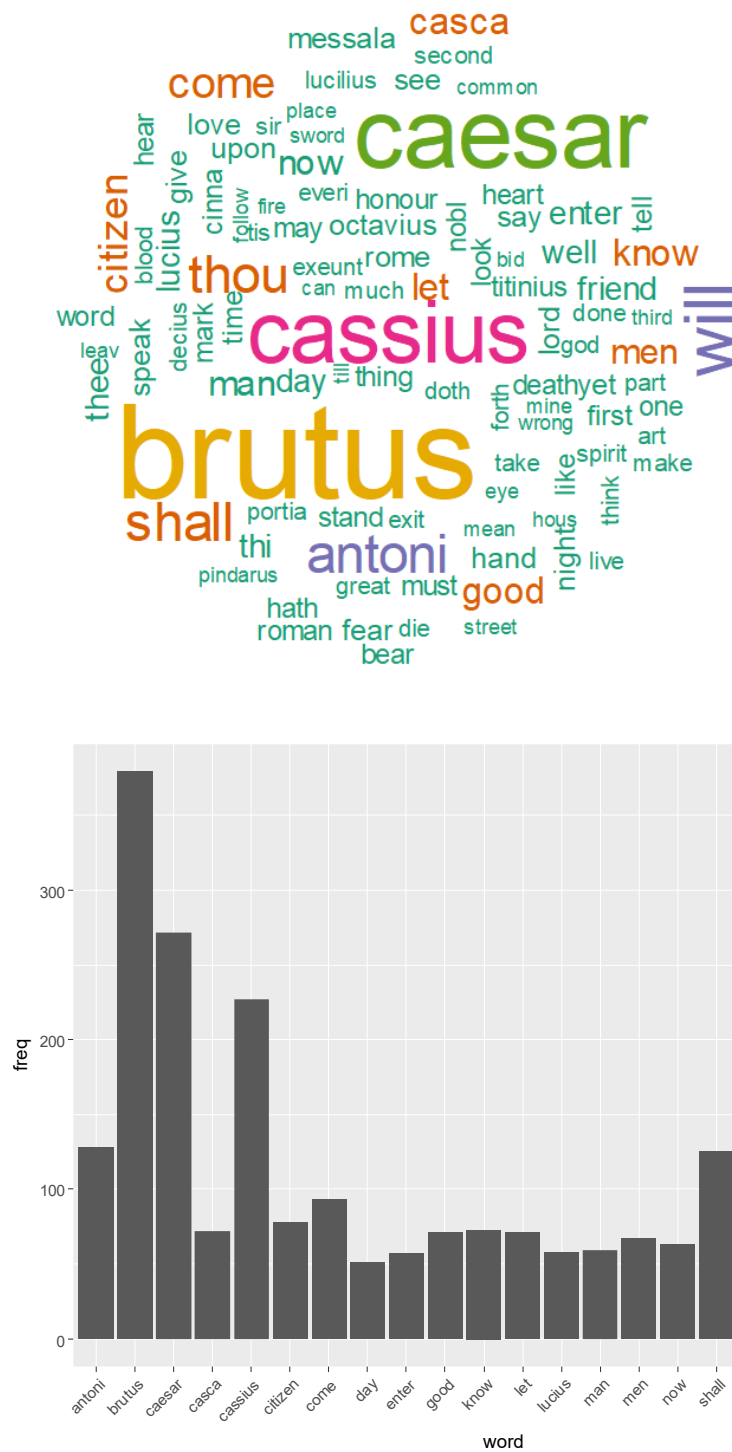
plot = plot + theme(axis.text.x=element_text(angle=45, hjust=1))
plot
freq  = colSums(as.matrix(abs_dtm))
dark2 = brewer.pal(6, "Dark2")
wordcloud(names(freq), freq, max.words=100, rot.per=0.2, colors=dark2)

```

1.Romeo and Juliet



2. Julius Caesar



3. Hamlet

