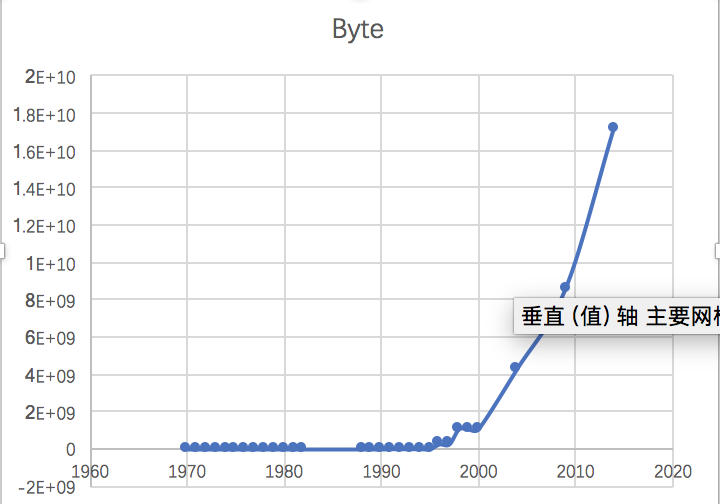
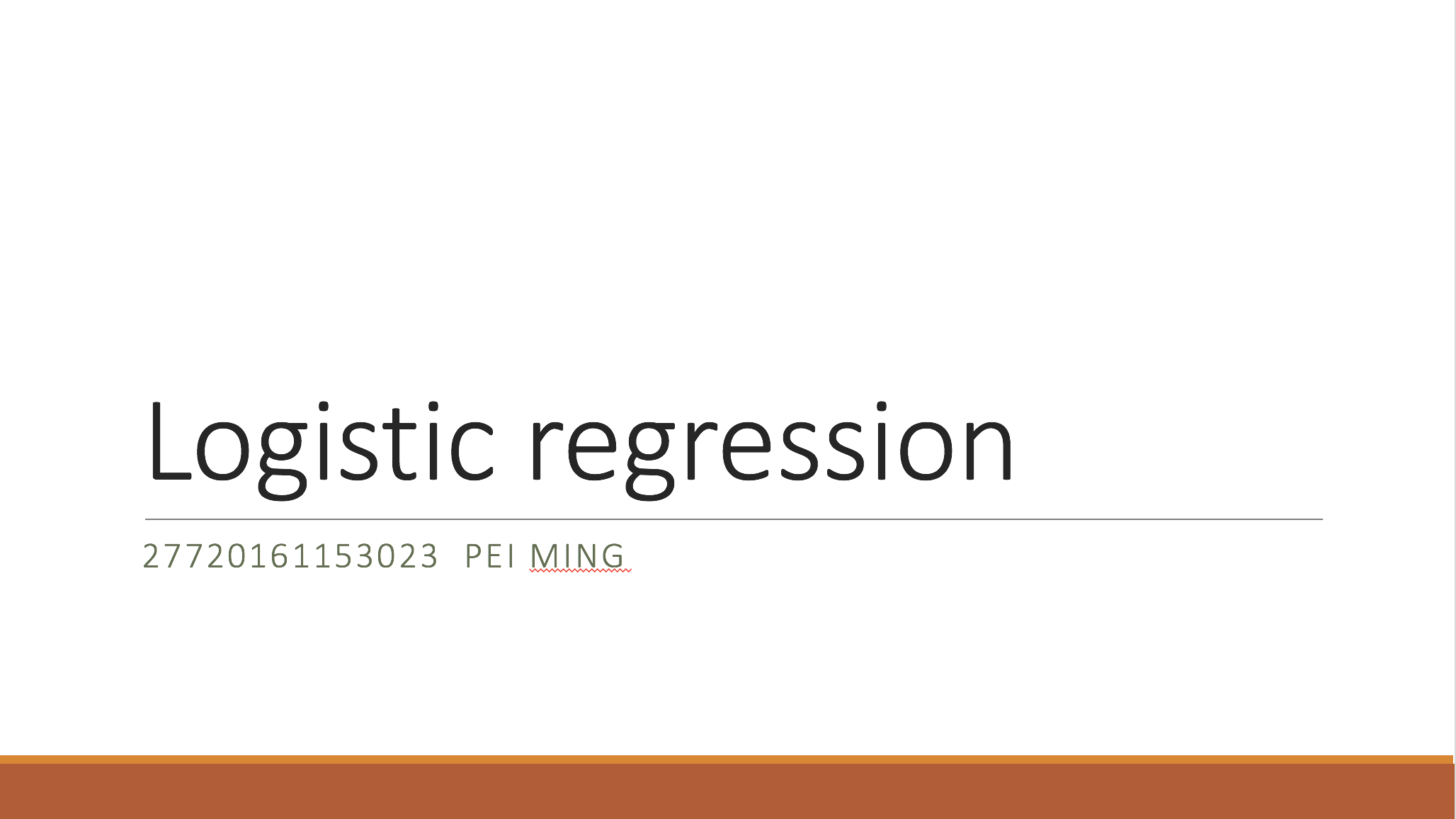
**Homework 1**

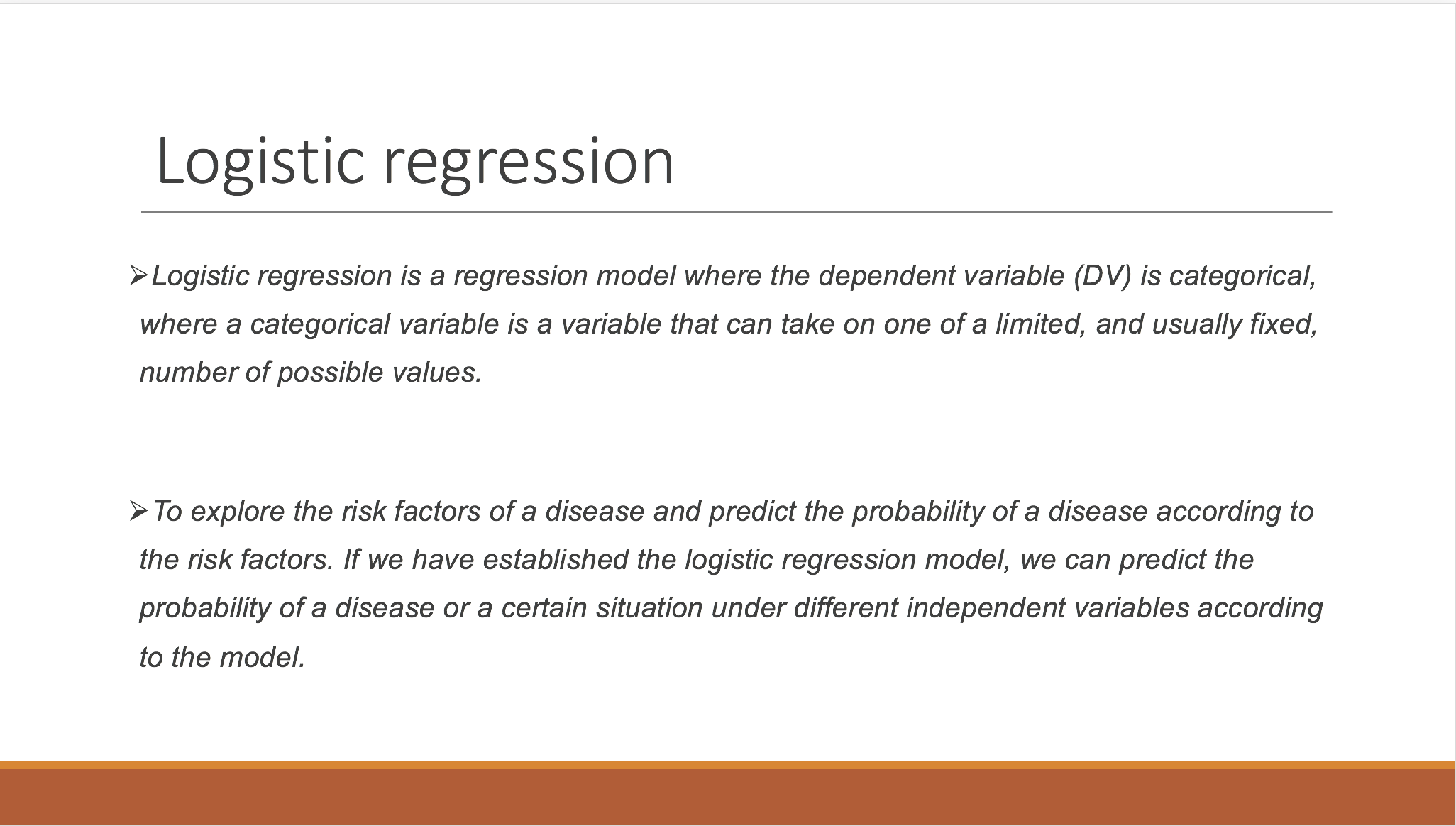
1. **memory of PCs**

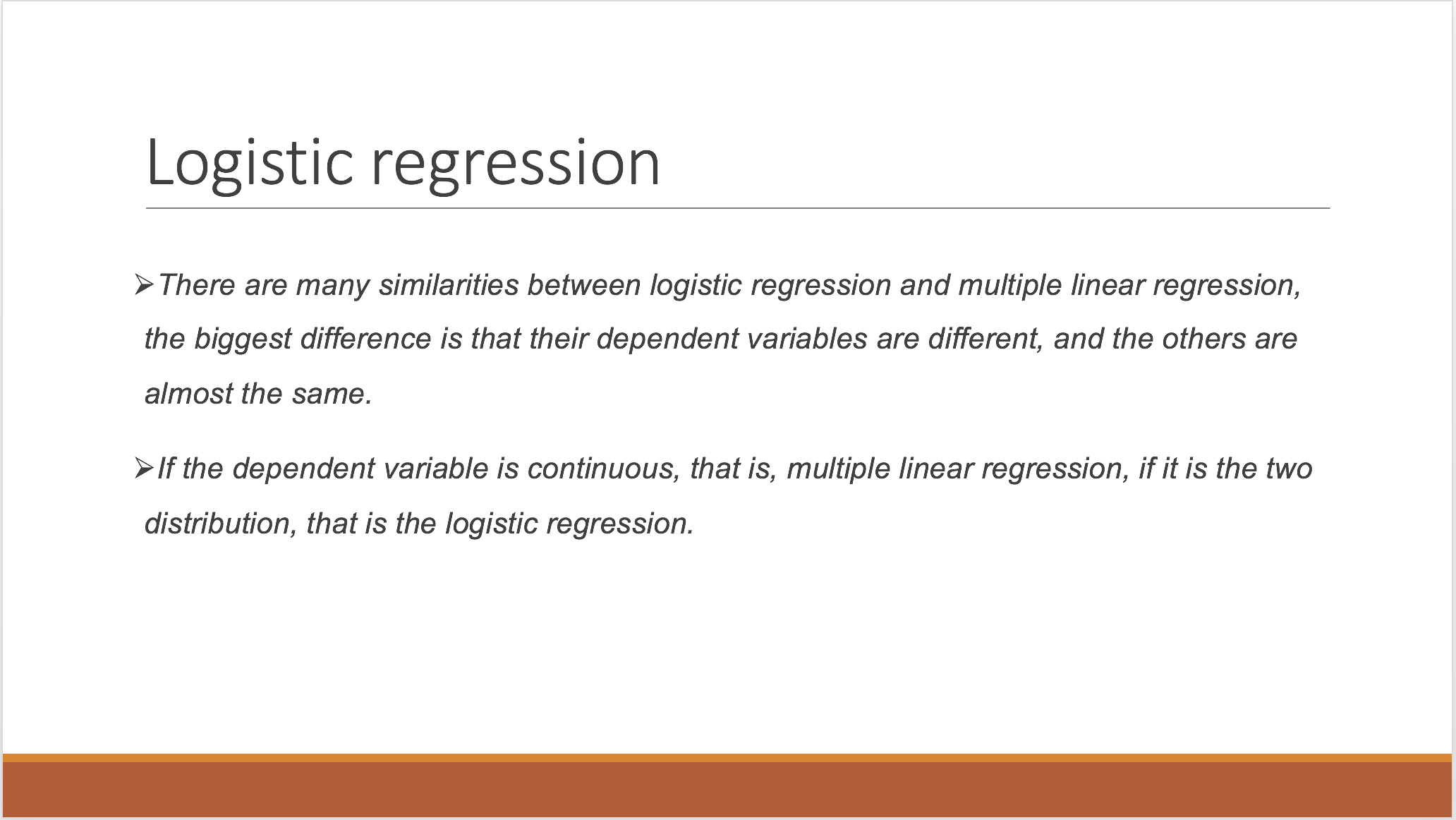
|  |  |  |  |
| --- | --- | --- | --- |
| ***year*** | ***Byte*** | ***year*** | ***Byte*** |
| 1970 | 262144 | 1990 | 2097152 |
| 1971 | 262144 | 1991 | 16777216 |
| 1972 | 262144 | 1992 | 16777216 |
| 1973 | 262144 | 1993 | 16777216 |
| 1974 | 262144 | 1994 | 16777216 |
| 1975 | 262144 | 1995 | 16777216 |
| 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 |  |  |



1. **logistic regression**







1. **Github Account:**  <https://github.com/mpmp2013/Home-Work-for-BDIF>

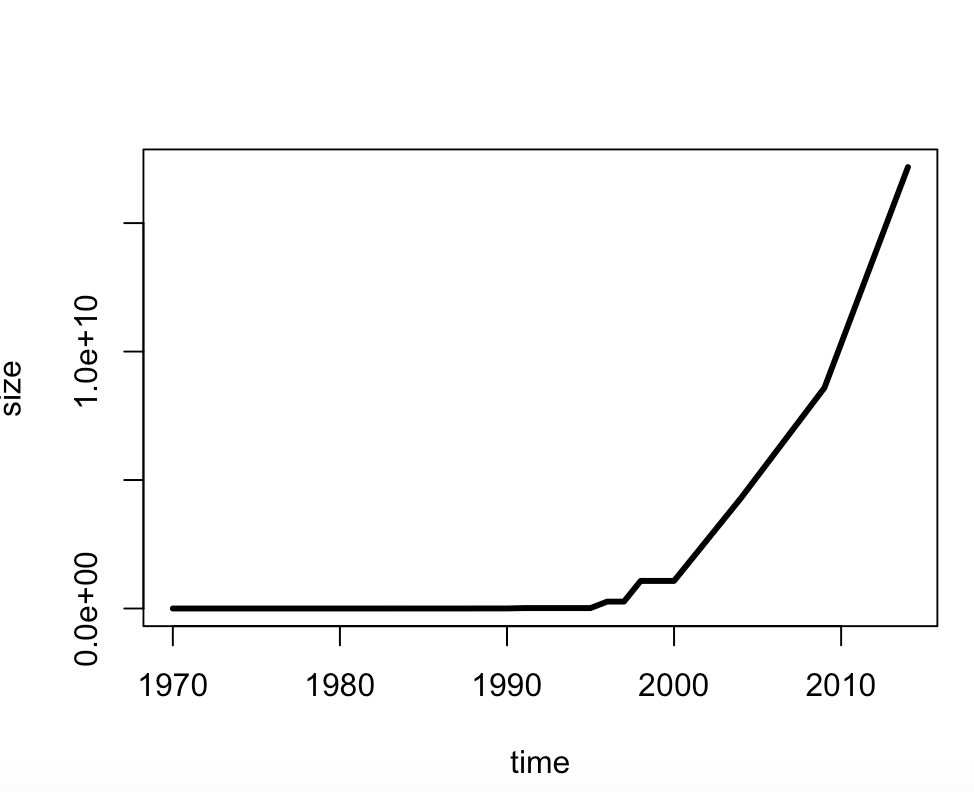
**Homework 2**

1. **Use R to solve HW #1**

*library(readr)*

*RAM\_size <- read\_csv("~/R data/Home-Work-for-BDIF/RAM\_size.csv")*

*plot(RAM\_size,type="l",xlab = "time",ylab = "size",lwd=3)*

**

1. **use R with B-spline code to solve HW#1**

*splines.reg.l1 = smooth.spline(x = RAM\_size$year, y = RAM\_size$Byte, spar = 0.2) # lambda = 0.2*

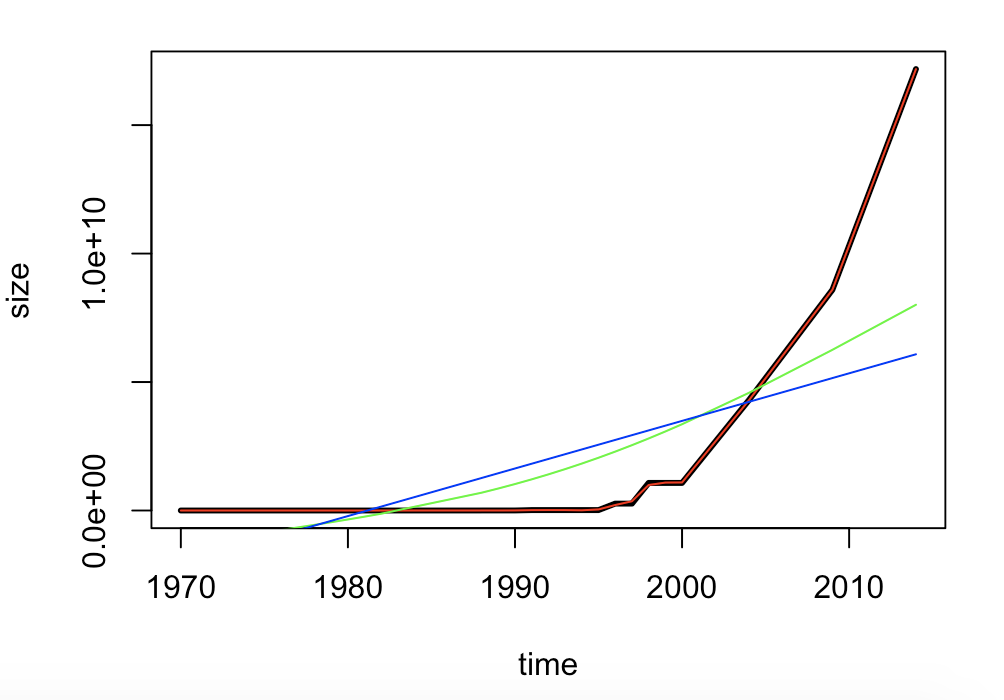
*splines.reg.l2 = smooth.spline(x = RAM\_size$year, y = RAM\_size$Byte, spar = 1) # lambda = 1*

*splines.reg.l3 = smooth.spline(x = RAM\_size$year, y = RAM\_size$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*

**

Comments: The larger the spar is, more smooth the line is.

1. **Poisson Distribution**

*lambda=4*

*x=6*

*dpois(x,lambda)*

*lambda=5*

*x=0*

*dpois(x,lambda)*

**Homework 3**

1. **hash code**

*#install.packages("digest",repos='http://cran.us.r-projest.org')*

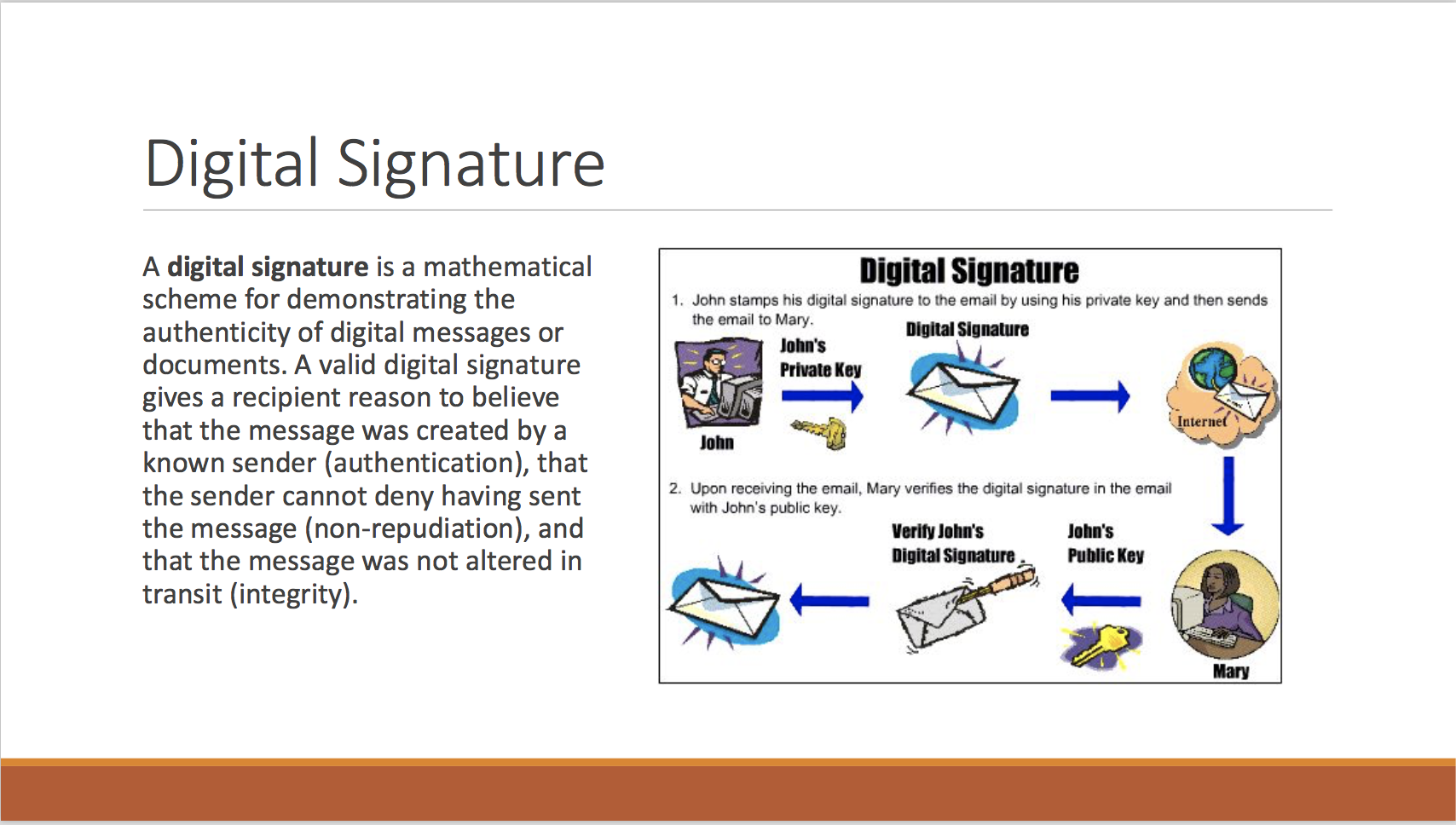
*library(digest)*

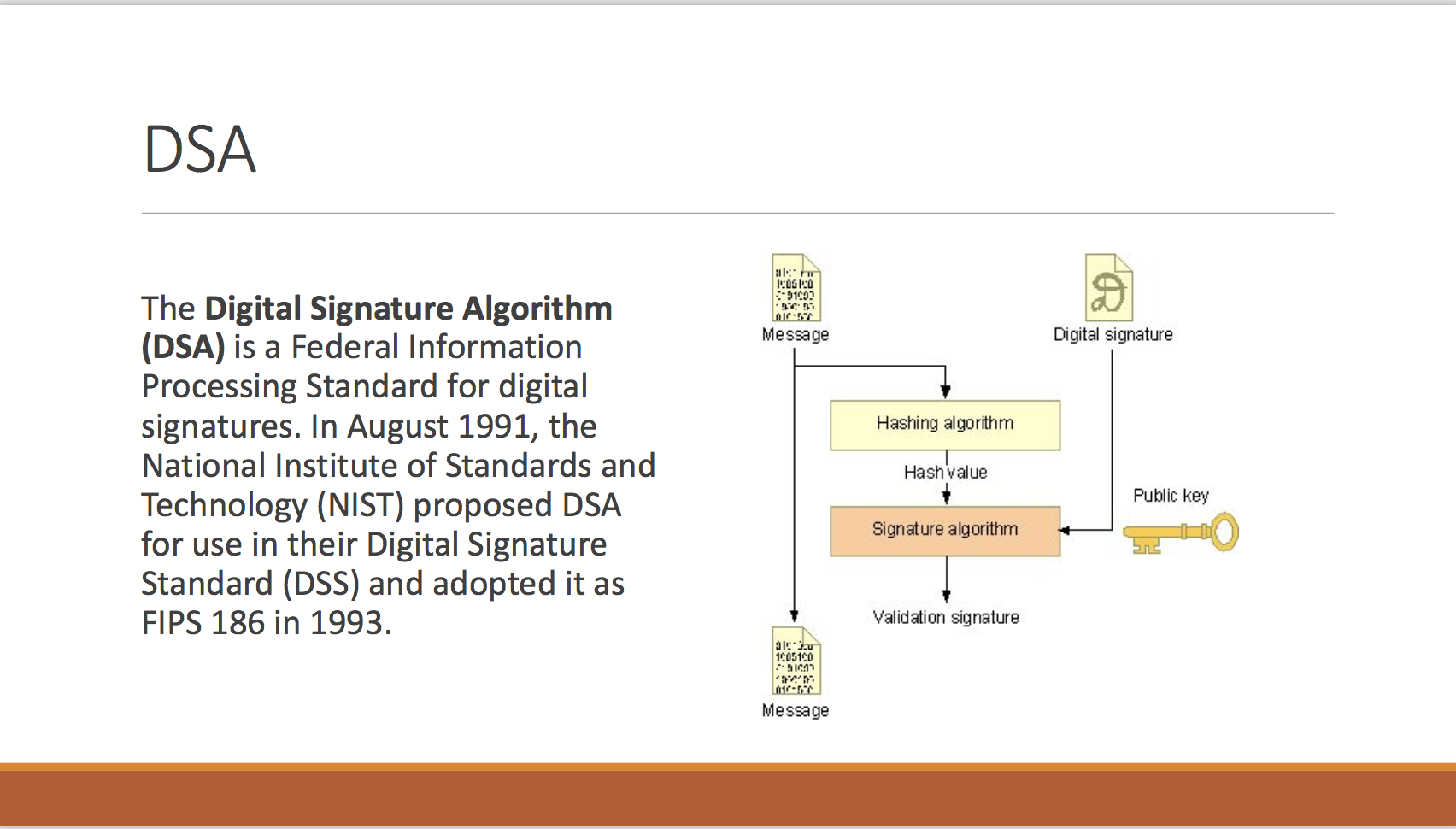
*digest("I learn a lot from this class when I am proper listening to the professor","sha256")*

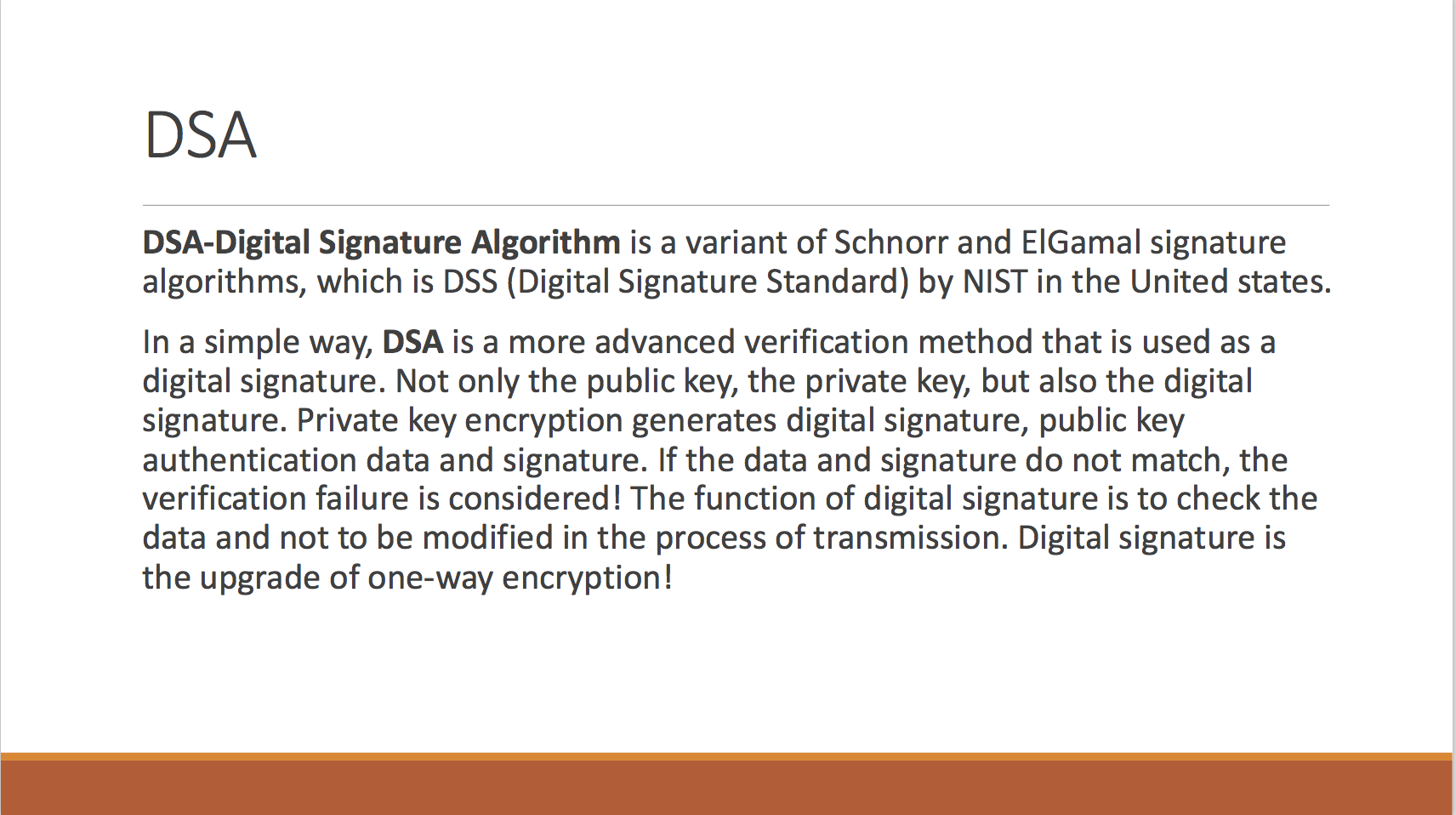
*digest("I do not learn a lot from this class when I am absent and playing on my Iphone","sha256")*

1. **Digital Signature Algorithms**

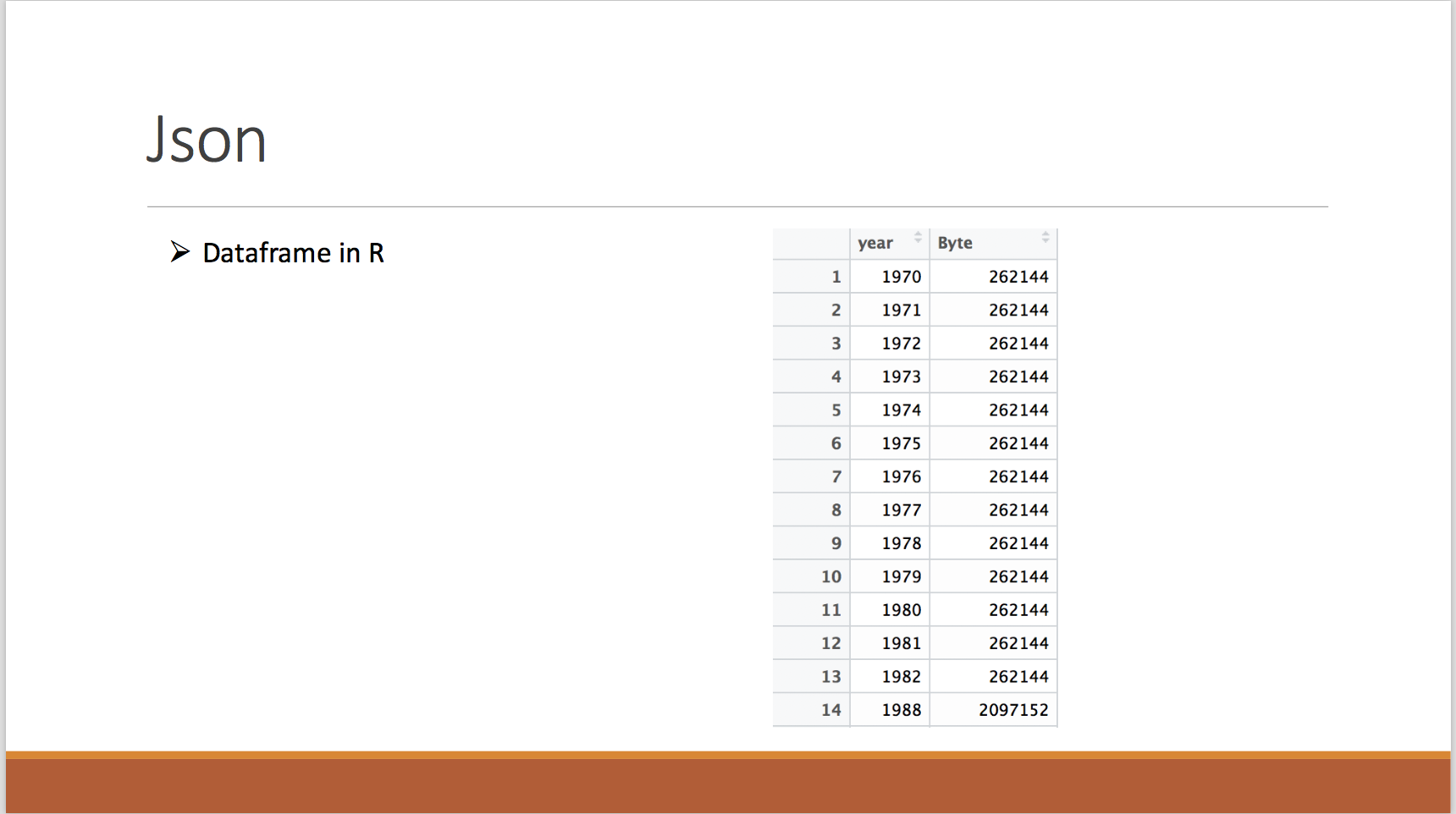








1. **Json data**





1. **CRIX data**

*#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)*

*lst <- lapply(json\_data,function(x){*

*df<-data.frame(date=x$date,price=x$price)*

*return(df)*

*})*

*crix\_data\_frame <- Reduce(rbind,lst)*

*plot(crix\_data\_frame$date,crix\_data\_frame$price)*

*#install.packages("forecast")*

*#install.packages("tseries")*

*library(forecast)*

*library(tseries)*

*ts.plot(crix\_data\_frame$price)*

*Acf(crix\_data\_frame$price)*

*for(i in 1:length(crix\_data\_frame$price)){*

*crixreturn[i] <- log(crix\_data\_frame$price[i+1]/crix\_data\_frame$price[i])*

*}*

*ts.plot(crixreturn)*

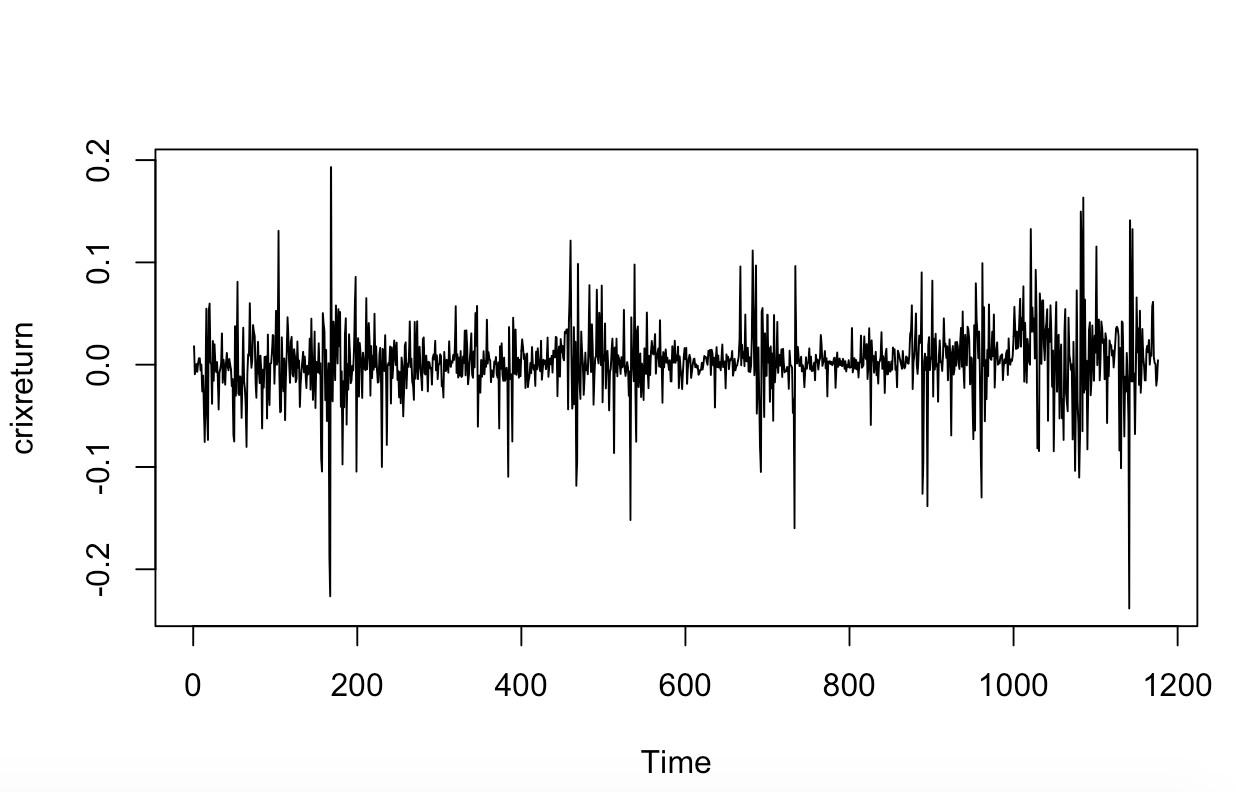
*Box.test(crixreturn, type = "Ljung-Box", lag = 20)*

*autocorr = acf(crixreturn, lag.max = 20, ylab = "Sample Autocorrelation", main = NA, lwd = 2, ylim = c(-0.3, 1))*

*Acf(crixreturn)*

*Pacf(crixreturn)*

*arima(crixreturn,order = c(2,0,2))*

**

**Homework 4**

1. **Figure3,4,5,6**

*#HW4.1*

*library(rjson)*

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

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

*lst <- lapply(json\_data,function(x){*

*df<-data.frame(date=x$date,price=x$price)*

*return(df)*

*})*

*crix\_data\_frame <- Reduce(rbind,lst)*

*crix\_data\_frame <- crix\_data\_frame[-1,]*

*load(file = "ecrix.RData")*

*load(file = "efcrix.RData")*

*length(ecrix)=length(crix\_data\_frame$price)*

*length(efcrix)=length(crix\_data\_frame$price)*

*ecrix\_data\_frame <- as.data.frame(ecrix)*

*efcrix\_data\_frame <- as.data.frame(efcrix)*

*#install.packages("dplyr")*

*library(dplyr)*

*sum\_crix <- cbind(crix\_data\_frame,ecrix\_data\_frame,efcrix\_data\_frame)*

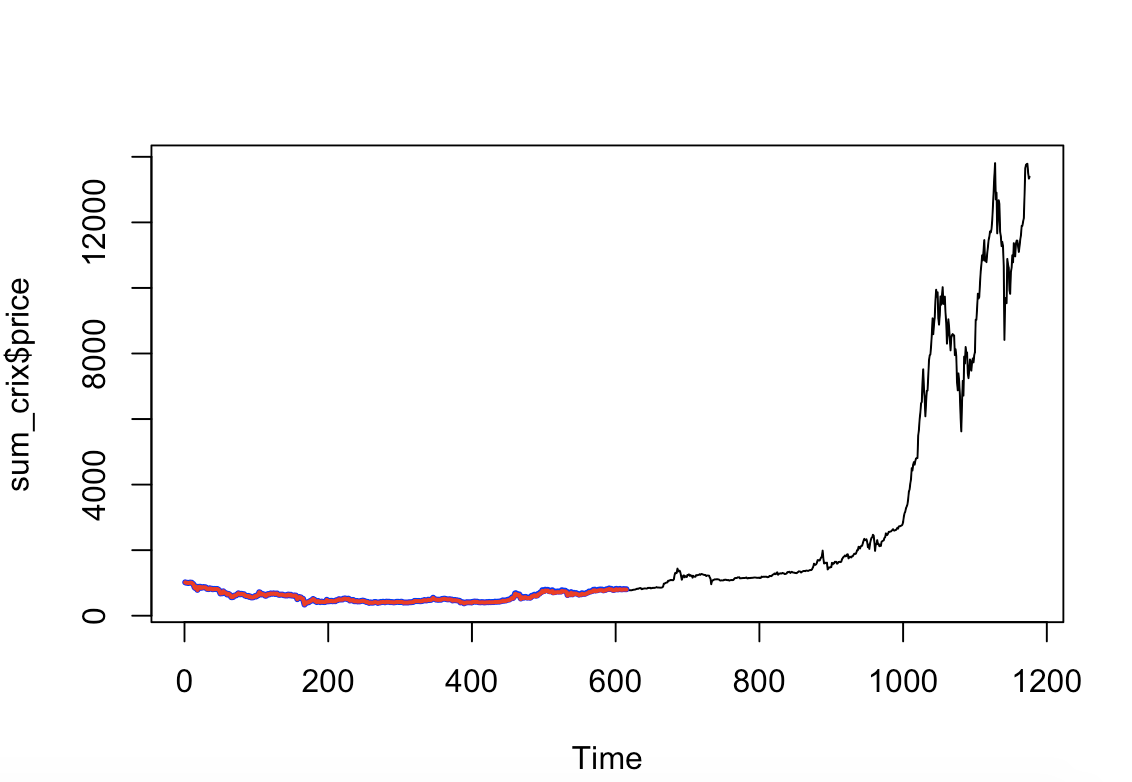
*#figure3*

*ts.plot(sum\_crix$price)*

*lines(sum\_crix$price,col="black",lwd=0.5)*

*lines(sum\_crix$ecrix,col="blue",lwd=1)*

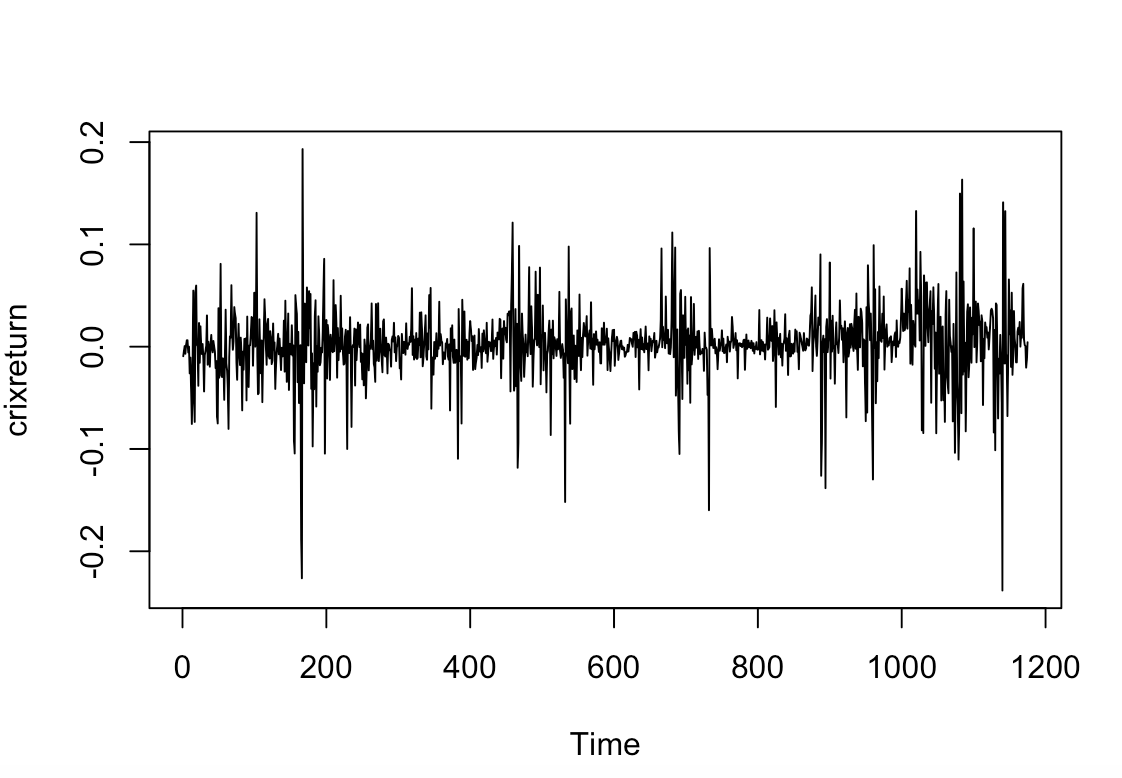
*lines(sum\_crix$efcrix,col="red",lwd=1)*

**

*#figure4*

*crixreturn <- diff(log(crix\_data\_frame$price))*

*ts.plot(crixreturn)*

**

*#figure5*

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

*lines(density(crixreturn),lwd=1)*

*mu = mean(crixreturn)*

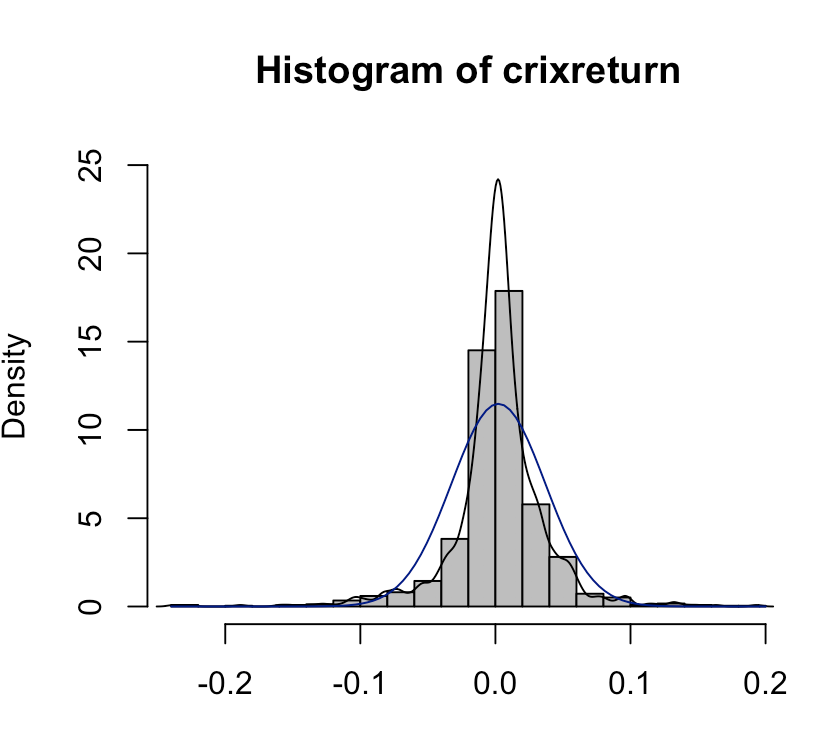
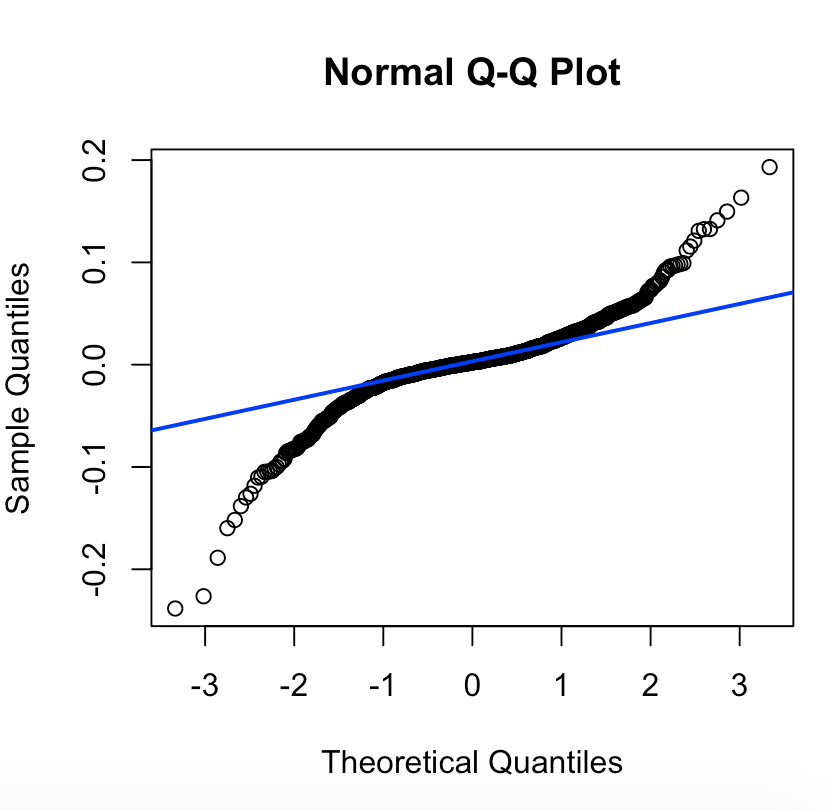
*sigma = sd(crixreturn)*

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

*curve(dnorm(x, mean = mean(crixreturn), sd = sd(crixreturn)), add = TRUE, col = "darkblue", lwd = 1)*

*qqnorm(crixreturn)*

*qqline(crixreturn, col = "blue", lwd = 2)*

** **

*#figure6*

*Box.test(crixreturn, type = "Ljung-Box", lag = 20)*

*adf.test(crixreturn, alternative = "stationary")*

*kpss.test(crixreturn, null = "Trend")*

*par(mfrow = c(1, 2))*

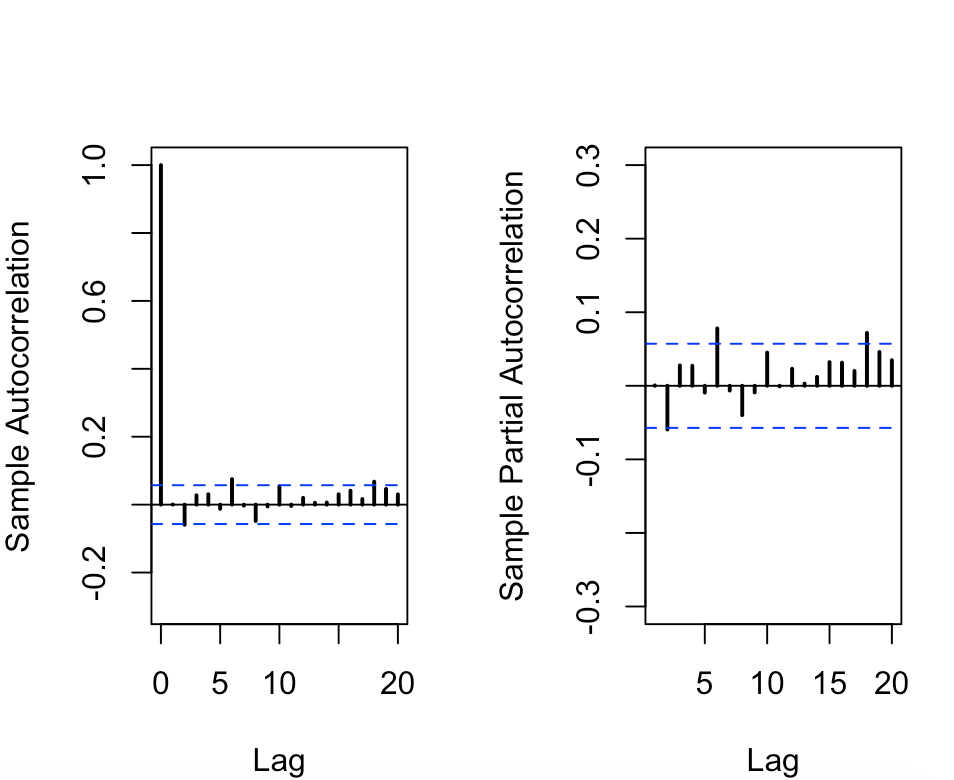
*autocorr = acf(crixreturn, lag.max = 20, ylab = "Sample Autocorrelation", main = NA, lwd = 2, ylim = c(-0.3, 1))*

*print(cbind(autocorr$lag, autocorr$acf))*

*Box.test(crixreturn, type = "Ljung-Box", lag = 1, fitdf = 0)*

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

*autopcorr = pacf(crixreturn, lag.max = 20, ylab = "Sample Partial Autocorrelation",main = NA, ylim = c(-0.3, 0.3), lwd = 2)*

**

1. **Figure 7**

*par(mfrow = c(1, 1))*

*auto.arima(crixreturn)*

*fit1 = arima(crixreturn, order = c(1, 0, 1))*

*tsdiag(fit1)*

*Box.test(fit1$residuals, lag = 1)*

*aic = matrix(NA, 6, 6)*

*for (p in 0:4) {*

*for (q in 0:3) {*

*a.p.q = arima(crixreturn, order = c(p, 0, q))*

*aic.p.q = a.p.q$aic*

*aic[p + 1, q + 1] = aic.p.q*

*}*

*}*

*aic*

*bic = matrix(NA, 6, 6)*

*for (p in 0:4) {*

*for (q in 0:3) {*

*b.p.q = arima(crixreturn, order = c(p, 0, q))*

*bic.p.q = AIC(b.p.q, k = log(length(crixreturn)))*

*bic[p + 1, q + 1] = bic.p.q*

*}*

*}*

*bic*

*fit4 = arima(crixreturn, order = c(2, 0, 3))*

*tsdiag(fit4)*

*Box.test(fit4$residuals, lag = 1)*

*fitr4 = arima(crixreturn, order = c(2, 1, 3))*

*tsdiag(fitr4)*

*Box.test(fitr4$residuals, lag = 1)*

*fit202 = arima(crixreturn, order = c(2, 0, 2))*

*tsdiag(fit202)*

*tsdiag(fit4)*

*tsdiag(fitr4)*

*AIC(fit202, k = log(length(crixreturn)))*

*AIC(fit4, k = log(length(crixreturn)))*

*AIC(fitr4, k = log(length(crixreturn)))*

*fit202$aic*

*fit4$aic*

*fitr4$aic*

*fit202 = arima(crixreturn, order = c(2, 0, 2))*

*crpre = predict(fit202, n.ahead = 30)*

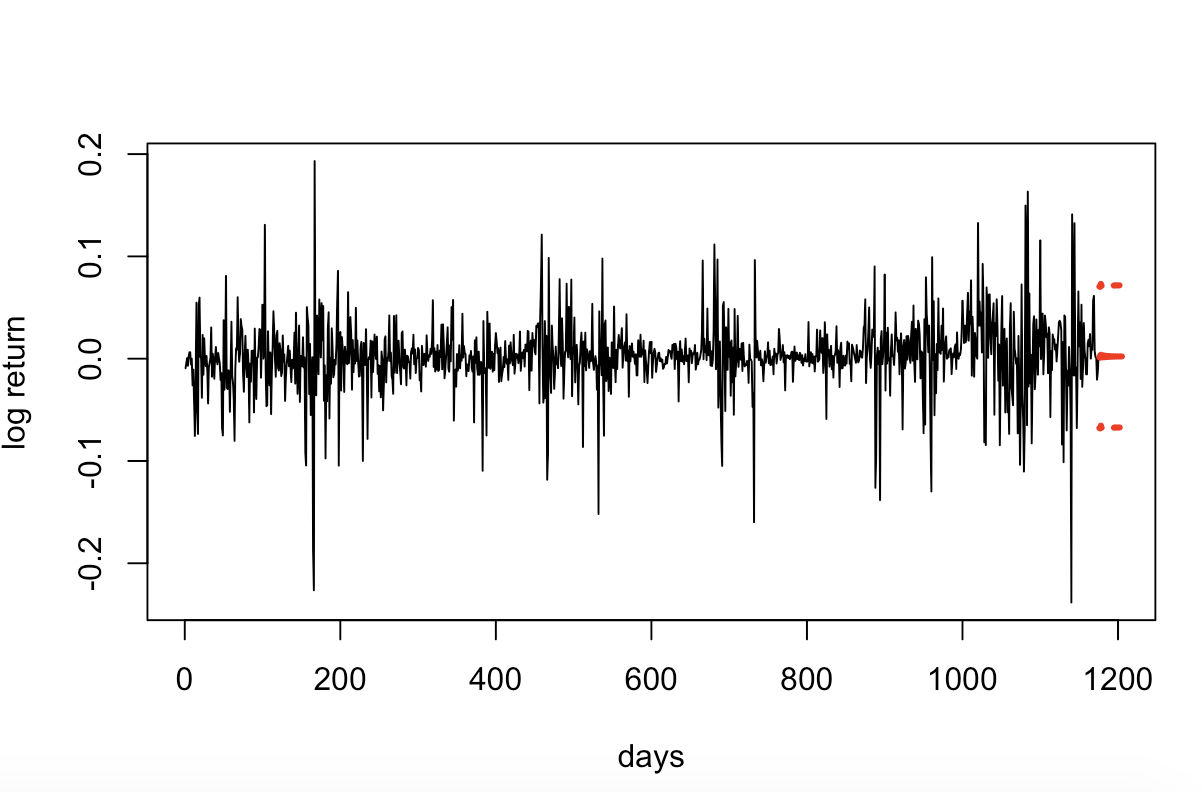
*dates = seq(as.Date("02/08/2014", format = "%d/%m/%Y"), by = "days", length = length(crixreturn))*

*plot(crixreturn, type = "l", xlim = c(0, 1200), ylab = "log return", xlab = "days", lwd = 1)*

*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)*

**

**Homework 5**

*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")*

*#HW5.1  Wordcloud*

*#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("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")*

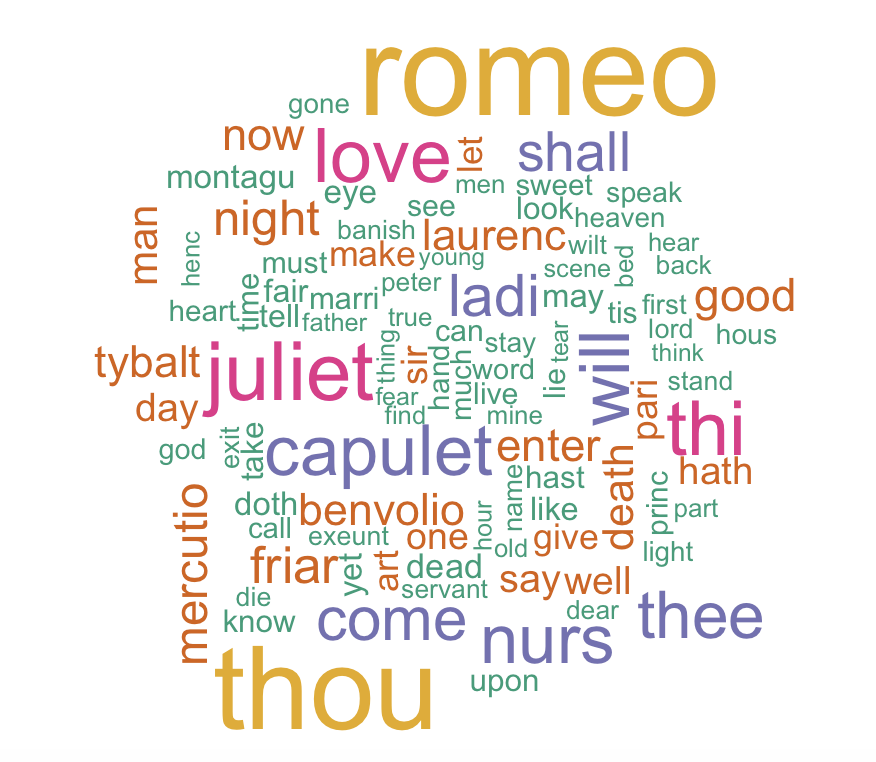
*plot1 = plot1 + theme(axis.text.x=element\_text(angle=45, hjust=1))*

*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 Caesar*

*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")*

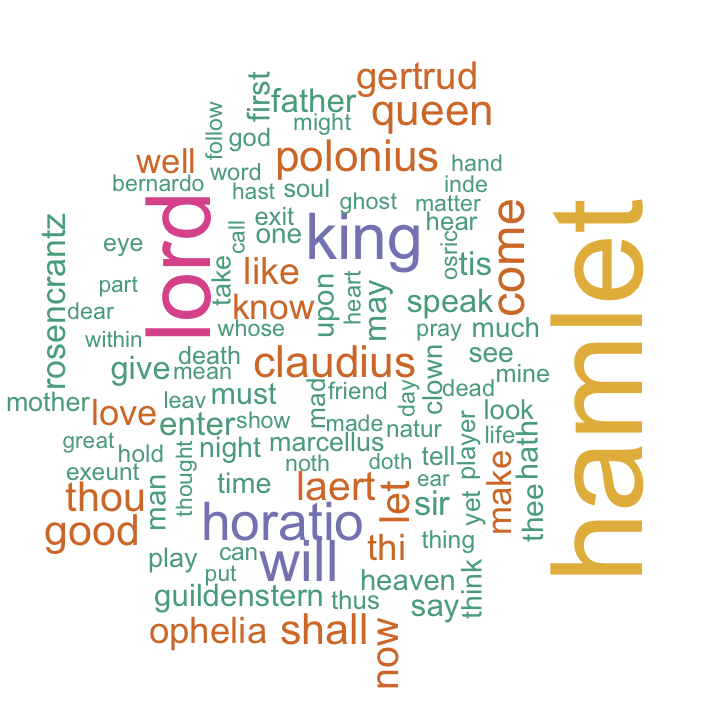
*plot3 = plot3 + theme(axis.text.x=element\_text(angle=45, hjust=1))*

*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)*

**

*#HW5.2 histogram of top20 word*

*#Romeo and Juliet*

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

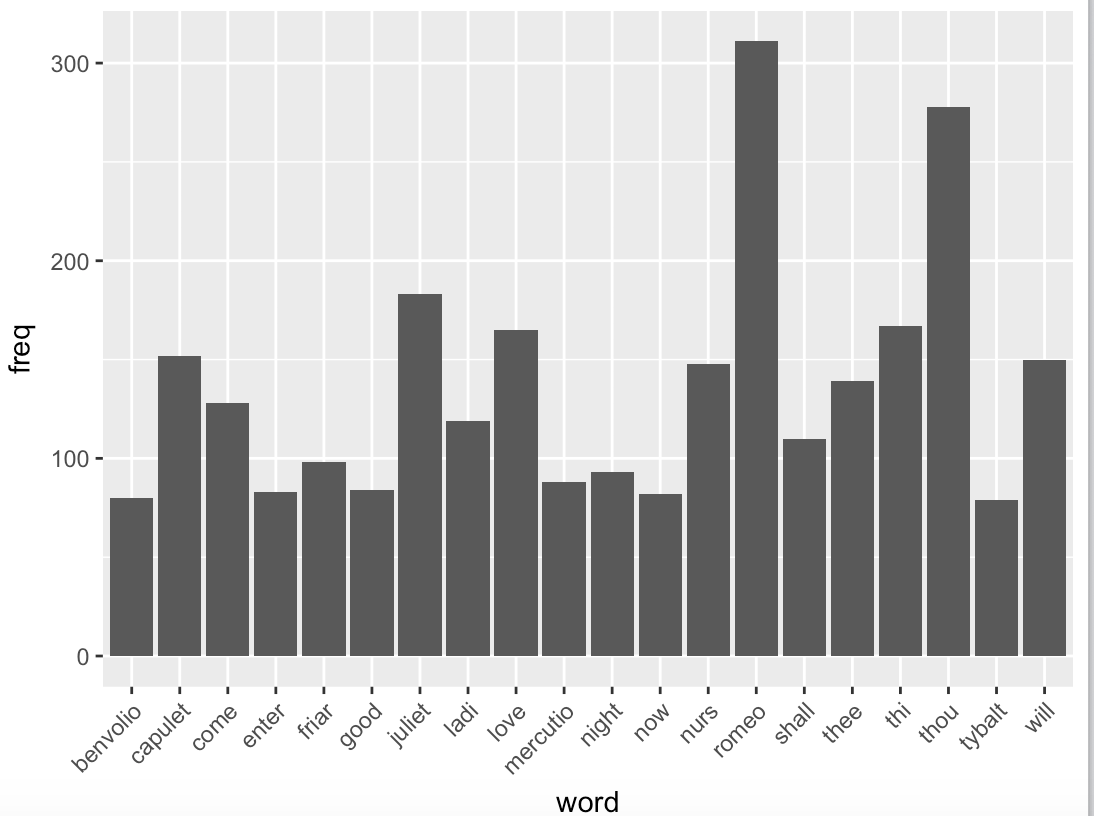
*wf1 <- wf1[c(1:20),]*

*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),]*

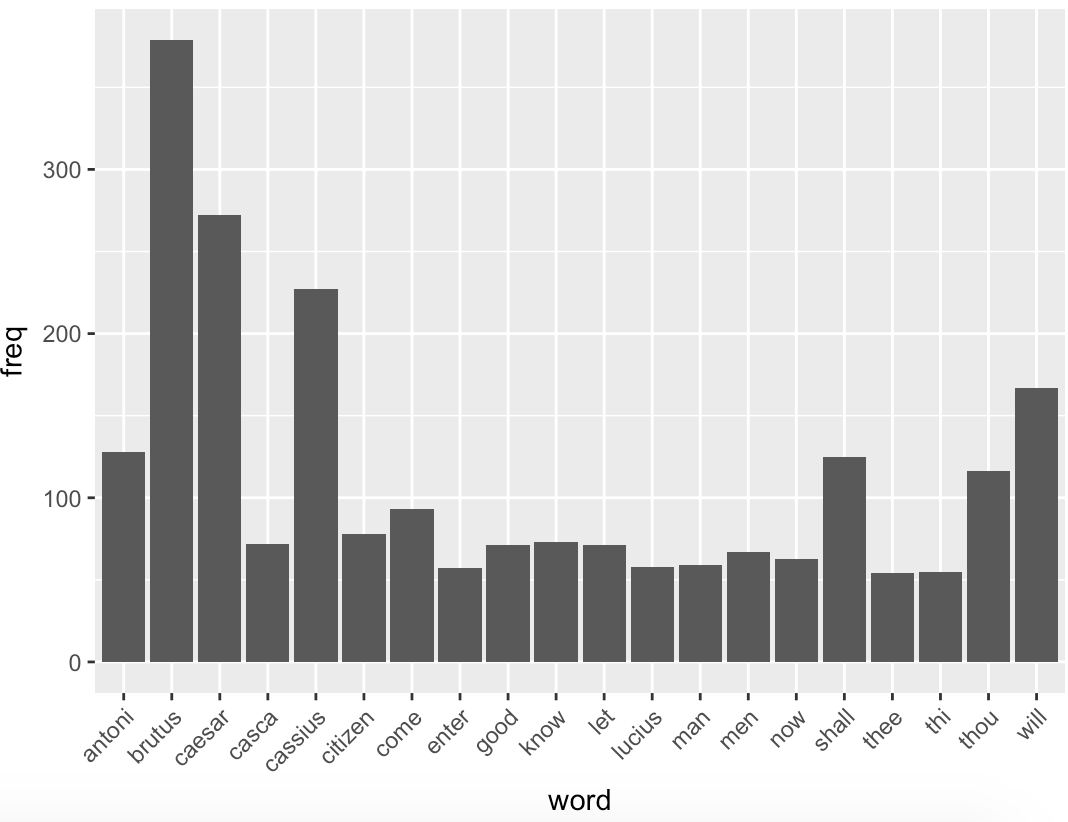
*wf2 <- wf2[c(1:20),]*

*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),]*

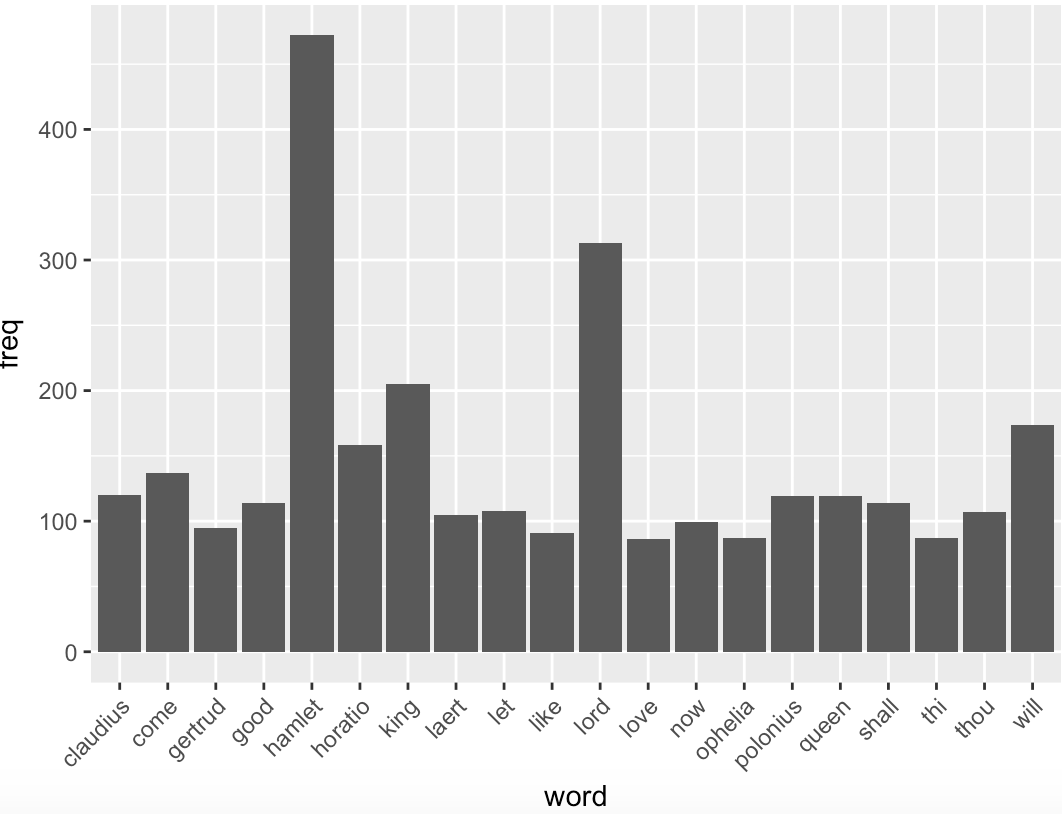
*wf3 <- wf3[c(1:20),]*

*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*

**