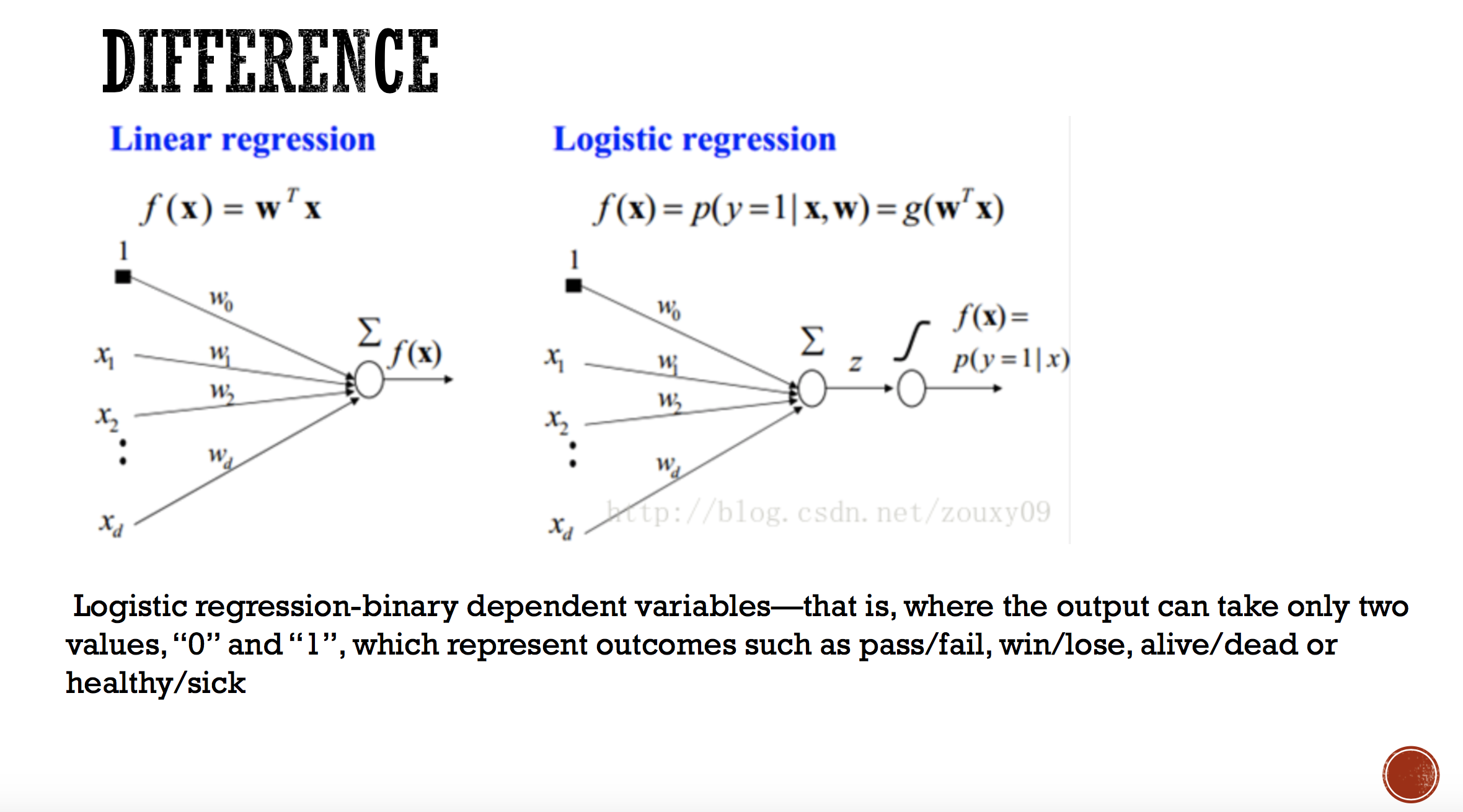
HW1

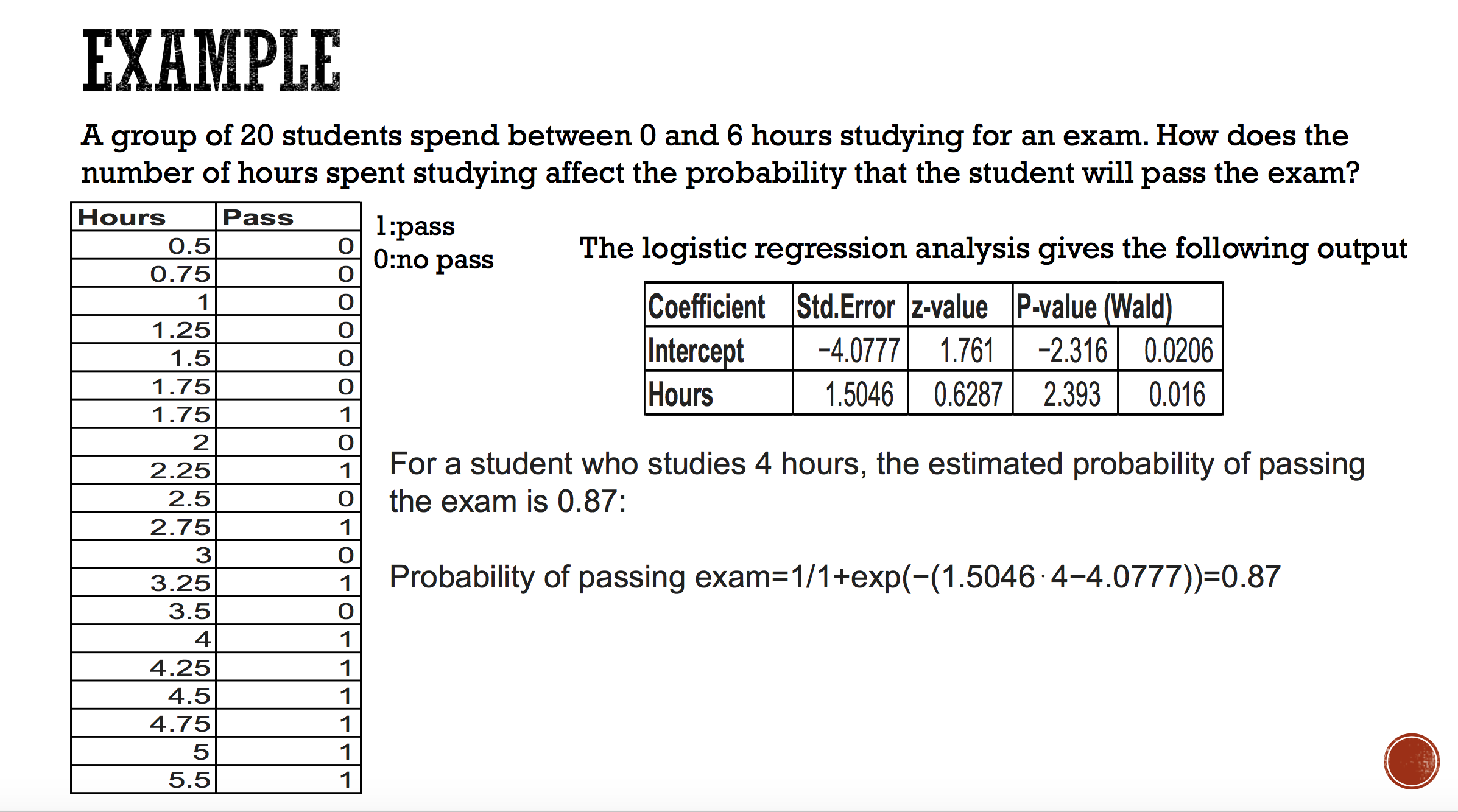
#Memory of PCs over the last 30 years



#Using PPT to explain logistic regression







HW2

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

Library(readxl)

data<-read.excel(“Desktop/ram.xls")

x<-data$Year

y<-data$RAM

plot(x,y)

lines(spline(x,y))

lines(spline(x, y, n = 201), col = 2)

#3

x = 6

n = 1000

lambda = 2

p = lambda / n

dbinom (x,2\*n,p) # binomial probability mass function

dpois (x, 2\*lambda ) # Poisson probability mass function

dpois (0, 5 )

HW3

#1

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")

library("digest")

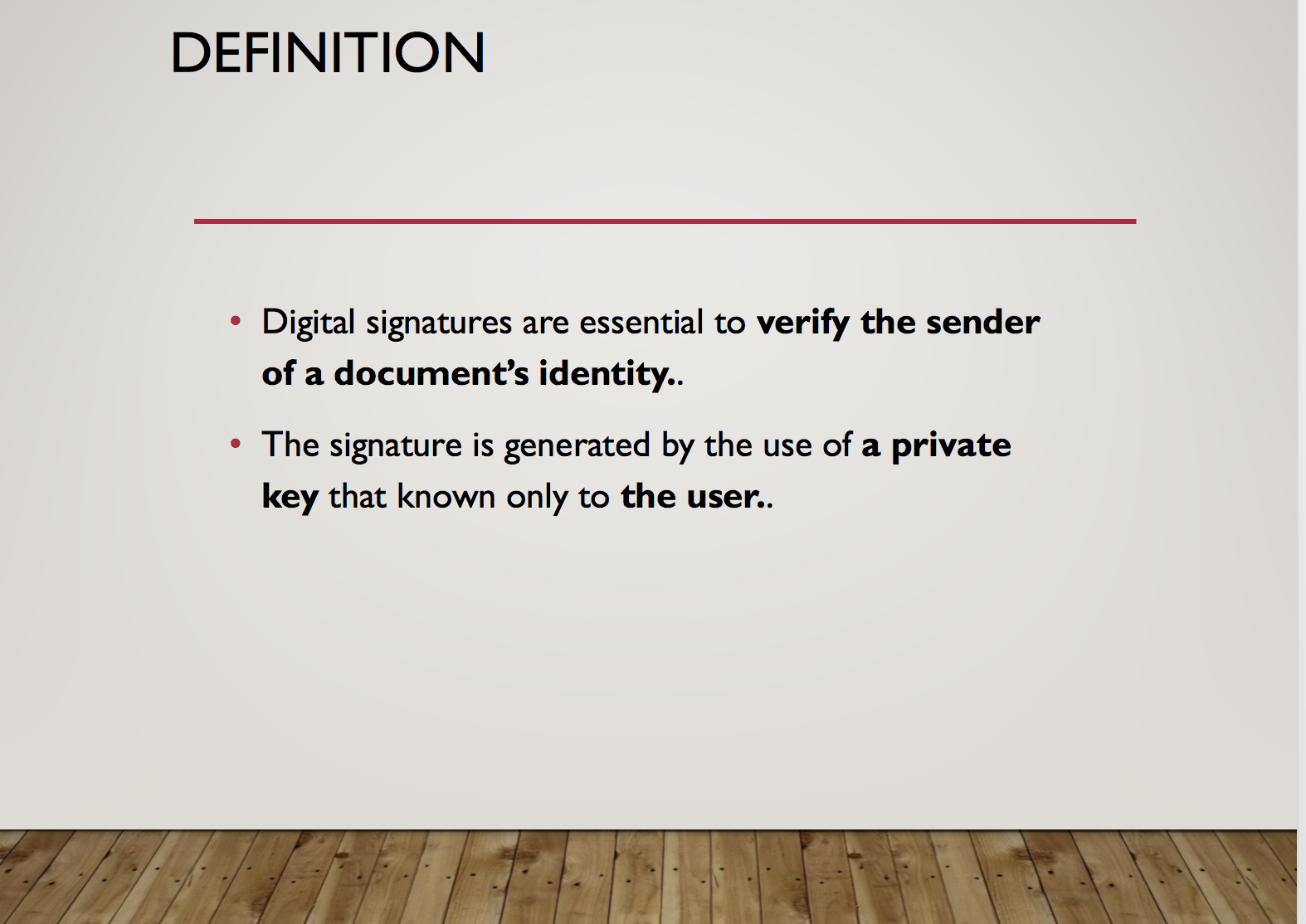
> digest("I learn a lot from this class when I am proper listening to the professor", "sha256")

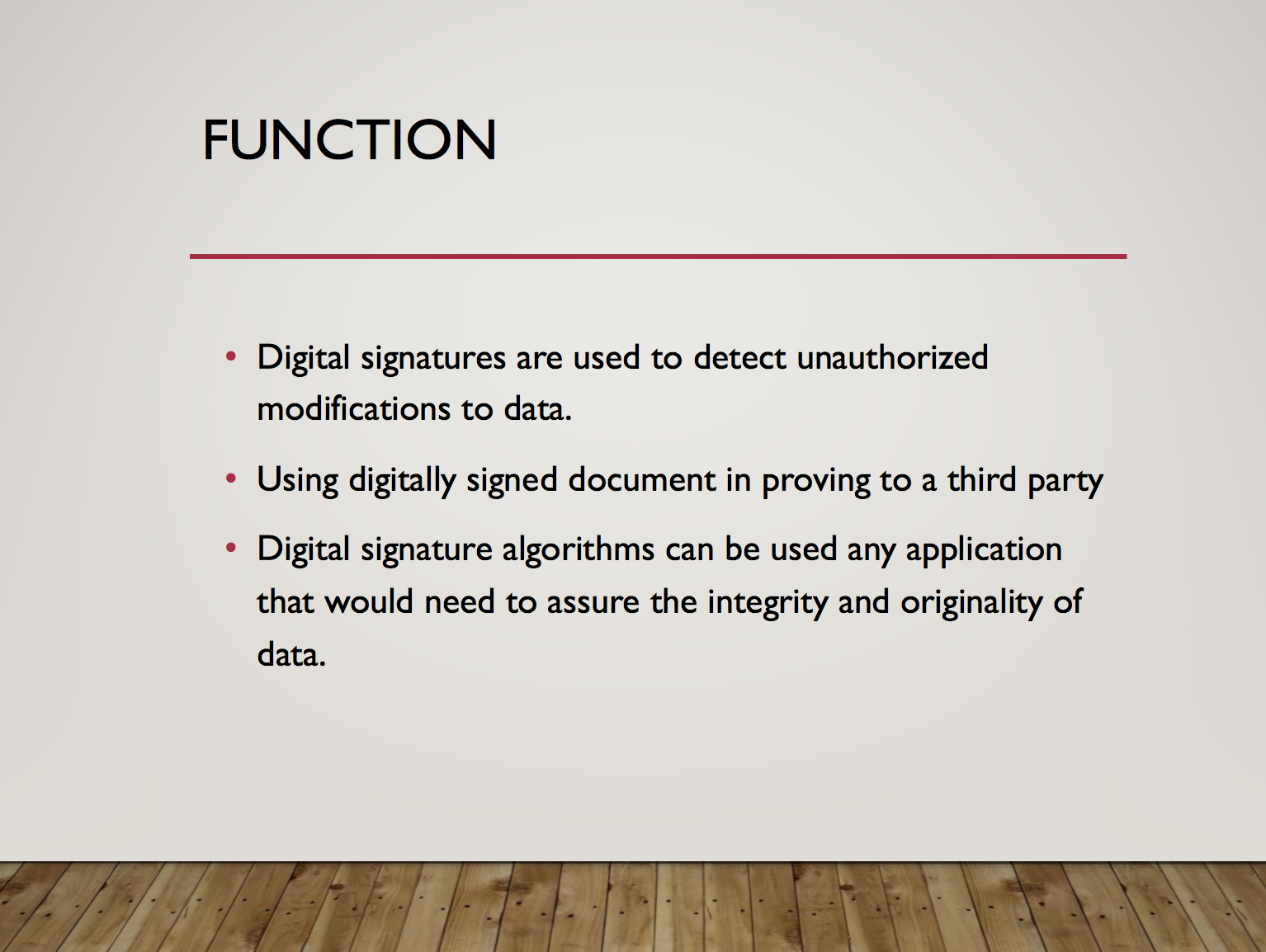
[1] "c16700de5a5c1961e279135f2be7dcf9c187cb6b21ac8032308c715e1ce9964c"

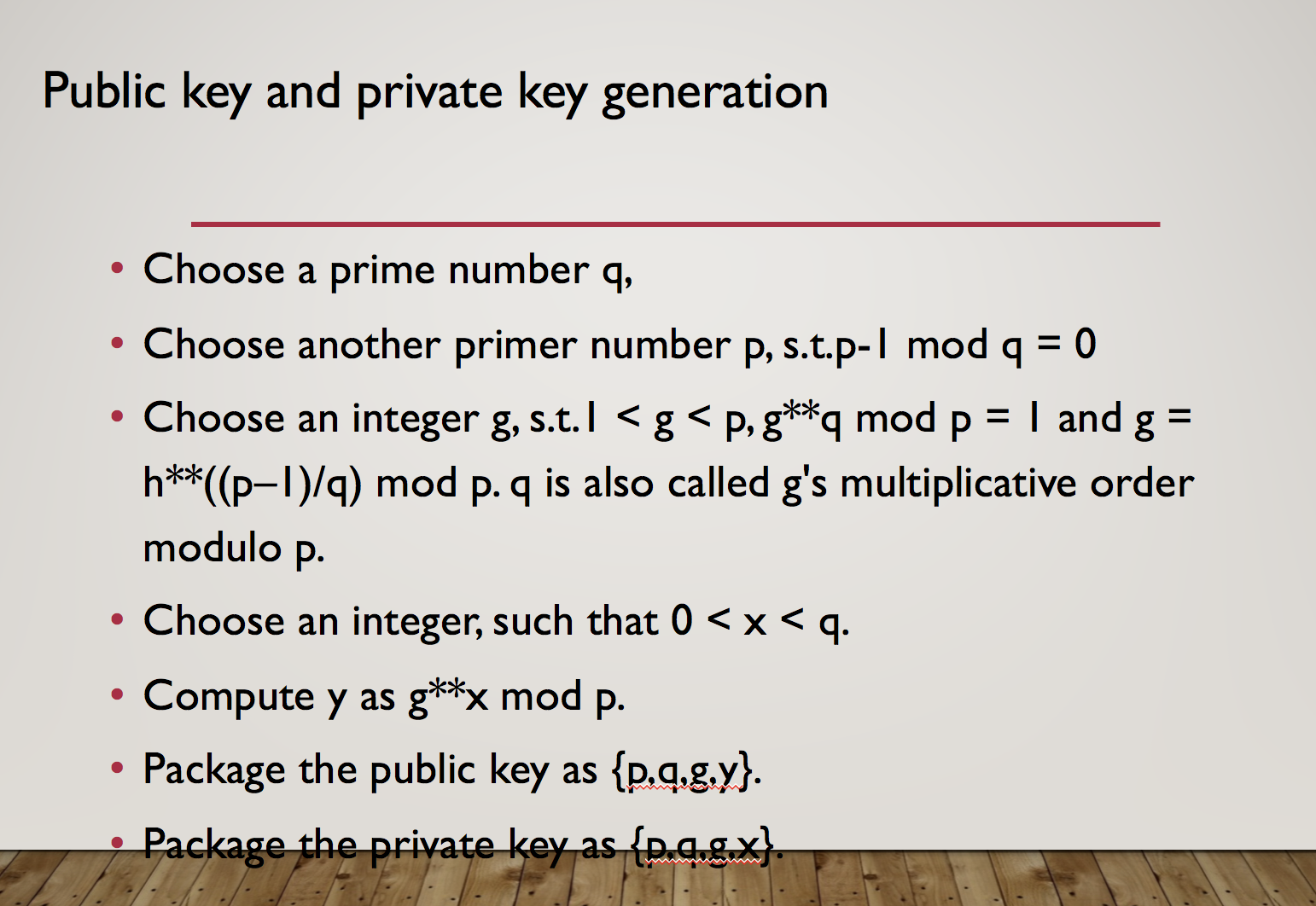
> digest("I do not learn a lot from this class when I am absent and playing on my Iphone", "sha256")

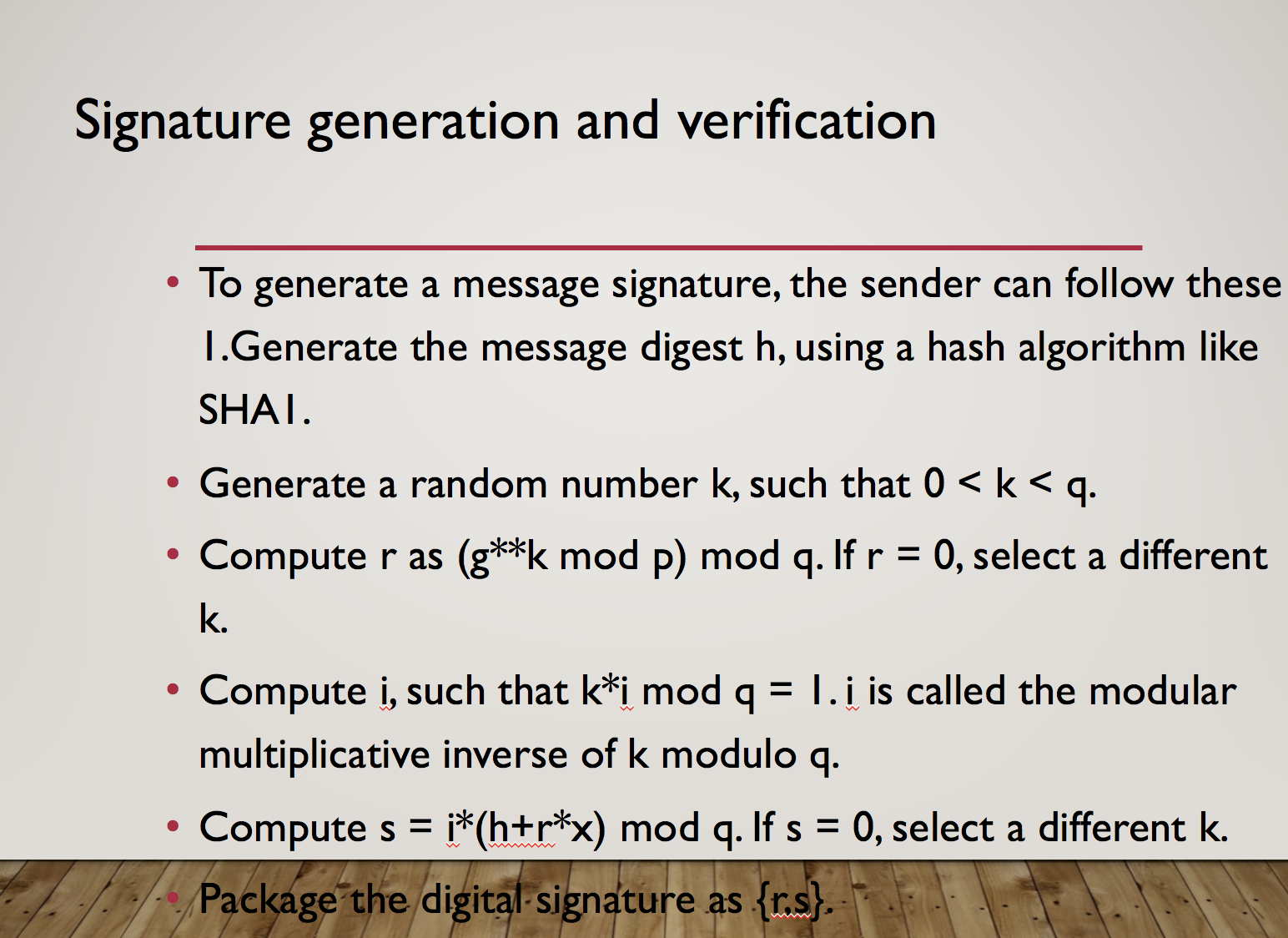
[1] "2533d529768409d1c09d50451d9125fdbaa6e5fd4efdeb45c04e3c68bcb3a63e"

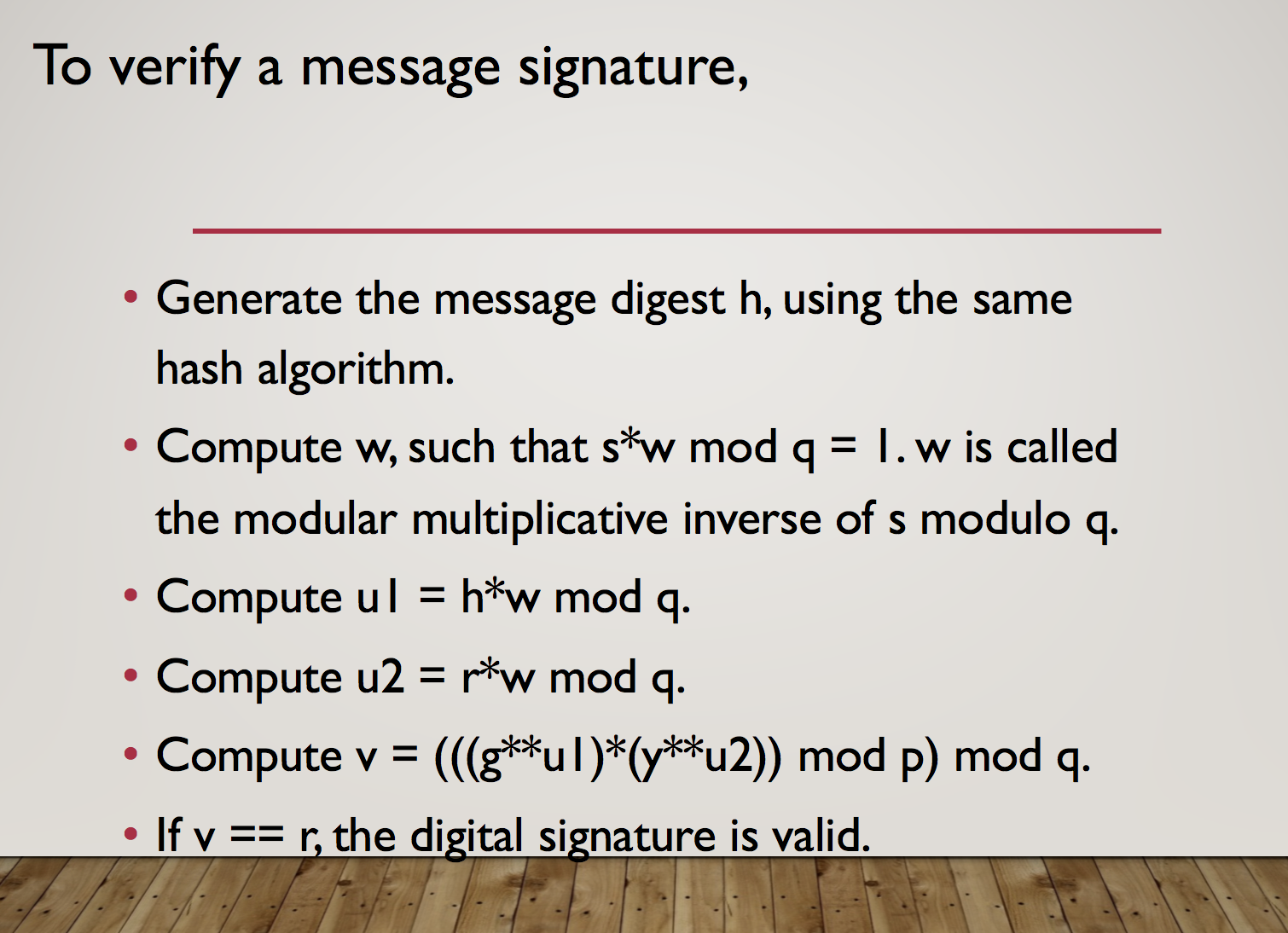
#2 Make 3-5 slides (in PPTX) on the DSA



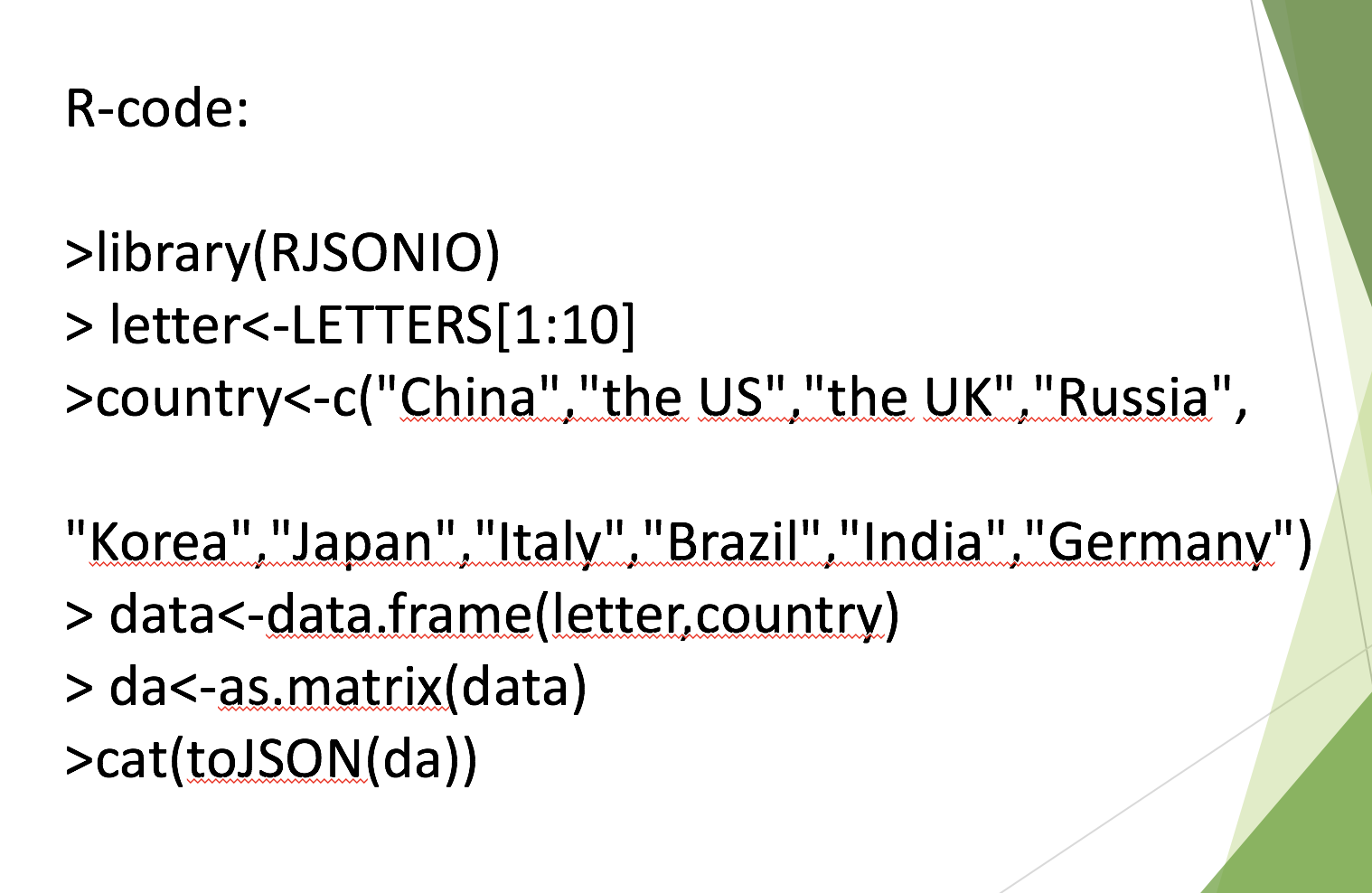








#3Make slides with R code where you create a JSON data se





#4

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/xiumei/Desktop/big data/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)

# Produces GARCH estimation results using ARIMA model residuals

rm(list = ls(all = TRUE))

graphics.off()

# install and load packages

libraries = c("FinTS", "tseries", "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 = "C:/Users/xiumei/Desktop/big data/crix.RData")

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

# kp test

set.seed(100)

x = rnorm(200)

# Do x and y come from the same distribution?

ks.test(x, fg11@residuals)

HW 4

#install.packages(“rjson”)

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)

n<-dim(crix\_data\_frame)

a<-seq(1,n[2],2)

b<-seq(2,n[2],2)

date<-t(crix\_data\_frame[1,a])

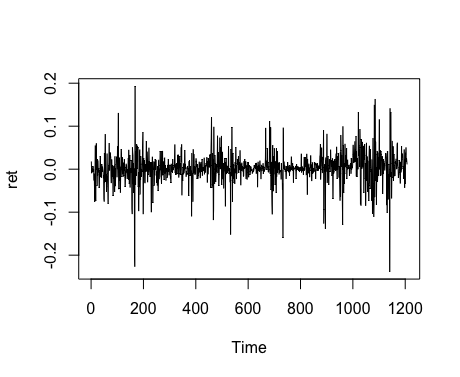
price<-t(crix\_data\_frame[1,b])

ts.plot(price)

ret<-diff(log(price))

plot(ret)

ts.plot(ret)



# histogram of returns

hist(ret, freq=equidist,col = "blue", breaks = 20, main=”The histogram of return”, ylim = c(0, 25), xlab = NA)

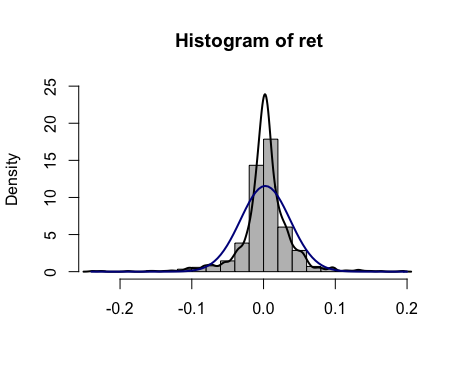
lines(density(ret), lwd = 1.8)

mu = mean(ret)

sigma = sd(ret)

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

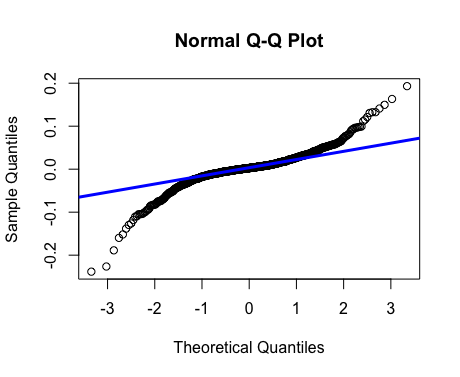
curve(dnorm(x, mean = mean(ret), sd = sd(ret)), add = TRUE, col = "darkblue", lwd = 2)



# qq-plot

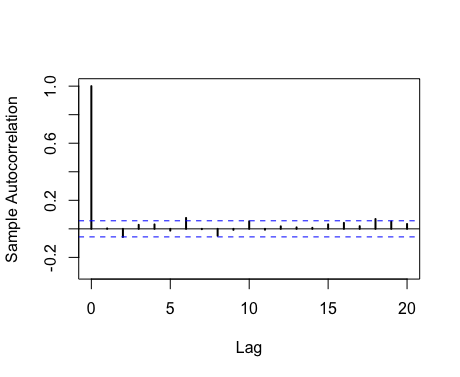
qqnorm(ret)

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



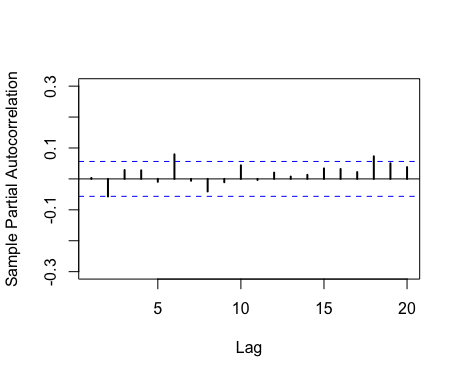
# acf plot

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



# plot of pacf

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



# select p and q order of ARIMA model

fit4 = arima(ret, order = c(2, 0, 3))

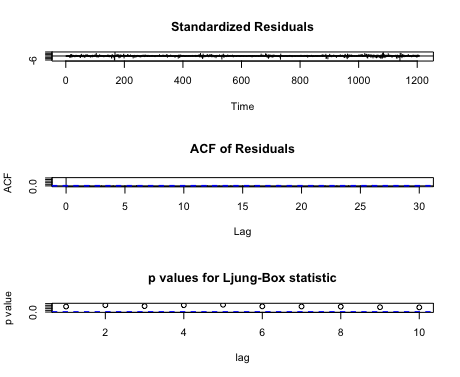
tsdiag(fit4)

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)

# arima202 predict

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

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

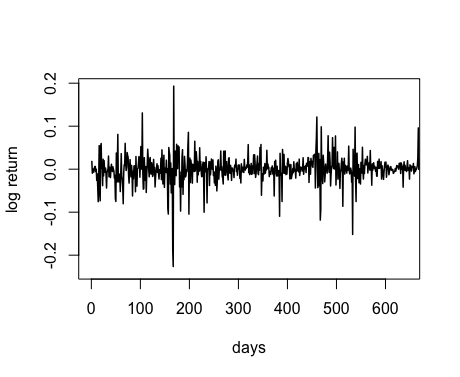
dates = seq(as.Date("02/08/2014", format = "%d/%m/%Y"), by = "days", length = length(ret))

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)



HW5

# do a word cloud

library(RCurl)

library(XML)

library(bitops)

library(stringr)

url="http://www.romeo-and-juliet.org/Romeo-and-Juliet-Script/romeo-and-juliet-play-script-ACT-4-SCENE-1-Friar-Laurences-cell.htm"

abs= lapply(url, 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)

}

cleantxt = lapply(abs,clean\_txt)

vec\_abs = unlist(cleantxt)

install.packages("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)

nstall.packages("wordcloud")

library(ggplot2)

library(wordcloud)

freq = colSums(as.matrix(abs\_dtm))

wf = data.frame(word=names(freq), freq=freq)

plot = ggplot(subset(wf, freq>15), aes(word, freq))

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

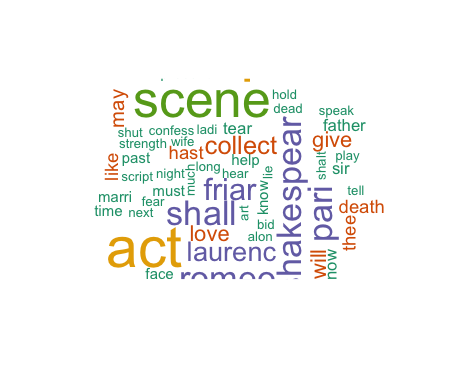
plot = plot + theme(axis.text.x=element\_text(angle=45, hjust=1))

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)



#HW5.2

plot = ggplot(subset(wf, freq>1), aes(word, freq))

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

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

plot

