## HU Extension Assignment 01 E-63 Big Data Analytics

## 

### Handed out: 09/01/2017 Due by 11:59AM on Saturday, 09/09/2017

It is recommended that your solution for this assignment is implemented in R. If you insist, you can submit your solution in any language of your choice.

**Problem 1.** Binomial distribution describes coin tosses with potentially doctored or altered coins. Value of p is the probability that head comes on top. If both the head and the tail have the same probability, p = 0.5. If the coin is doctored or altered, p could be larger or smaller. Plot on three separate graphs the binomial distribution for p = 0.3, p = 0.5 and p = 0.8 for the total number of trials n = 60 as a function of k, the number of successful (head up) trials. Subsequently, place all three curves on the same graph. For each value of p, determine 1st Quartile, median, mean, standard deviation and the 3rd Quartile. Present those values as a vertical box plot with the probability p on the horizontal axis.

**(15%)**

#For each of the probablilities .3,.5,.8 (referred slide 90)

# I get the binomial distribution for it using dbinom ( I also tried pbinom) for all the 60 trials.

#I used ranges (0:60) instead of individual values like in the slide for each of the probabilities mentioned.

set1 <- dbinom(0:60,60,.3)

set2 <- dbinom(0:60,60,.5)

set3 <- dbinom(0:60,60,.8)

#help(par)

#I then tried to plot these setting various parameters for the plot. I used par(new=TRUE) to keep adding to the same plot.

#I used ylim and xlim to set the x and y axis limits, and xlab and ylab for x and y labels,pch for the symbols and cols for color

plot(set1,ylim=c(0,.2),xlim=c(0,60),xlab ="Number of trials", ylab="Binomial distribution with k",pch=21,col="blue")

lines(set1)

par(new=TRUE)

plot(set2,ylim=c(0,.2),xlim=c(0,60),xlab ="Number of trials", ylab="Binomial distribution with k",pch=21,col="red")

lines(set2)

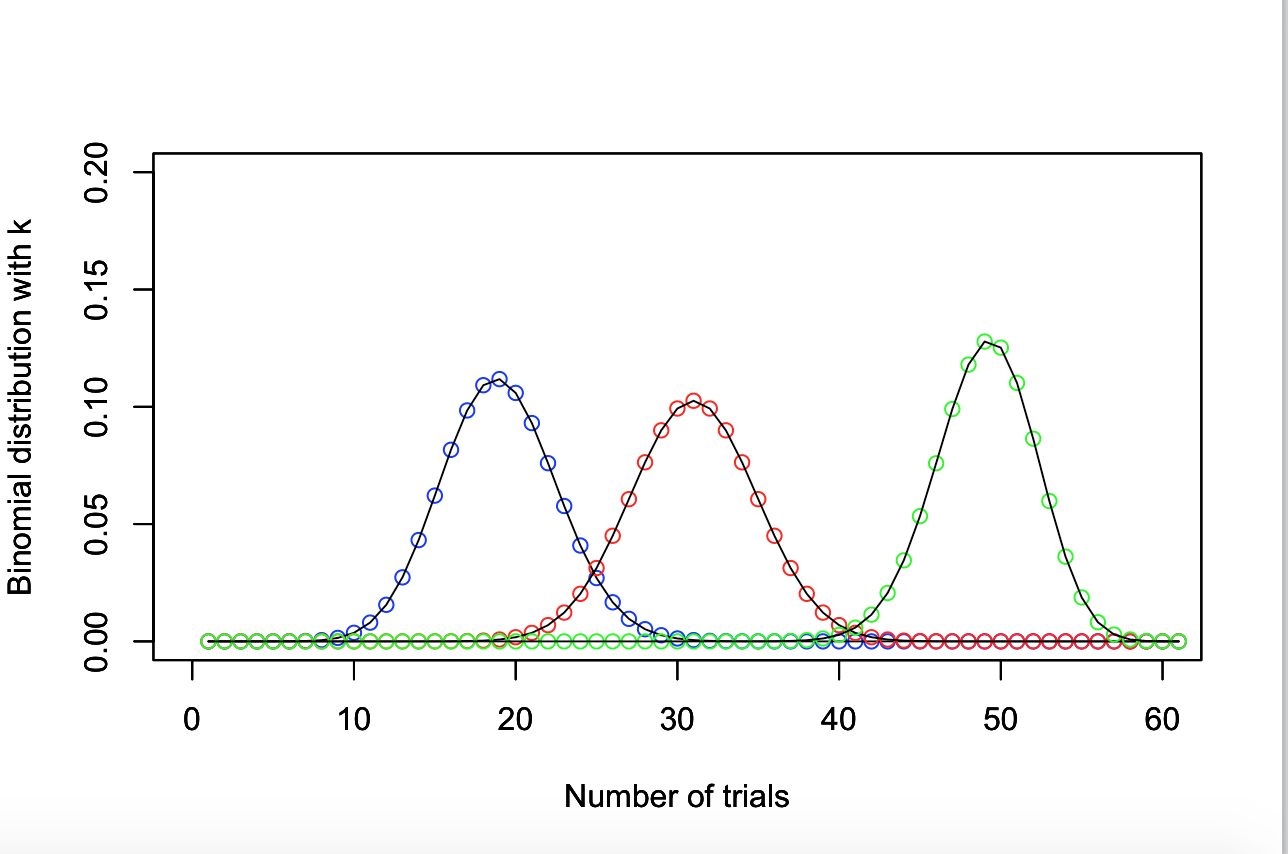
par(new=TRUE)

plot(set3,ylim=c(0,.2),xlim=c(0,60),xlab ="Number of trials", ylab="Binomial distribution with k",pch=21,col="green")

lines(set3)

#set4 <- pbinom(0:60, size=60, prob=0.3,xlab ="Number of trials", ylab="Binomial distribution with k",pch=21)

#help(dbinom)



#Here I calculate the 1st quantile, 3rd quantile,median,mean and standard deviation

#for set1 which is for 0.3

quantile1set1 <- quantile(set1,.25)

quantile3set1 <- quantile(set1,.75)

median1 <- median(set1)

mean1 <- mean(set1)

std1 <- sd(set1)

#Here I calculate the 1st quantile, 3rd quantile,median,mean and standard deviation

#for set2 which is for 0.5

quantile1set2 <- quantile(set2,.25)

quantile3set2 <- quantile(set2,.75)

median2 <- median(set2)

mean2 <- mean(set2)

std2 <- sd(set2)

#Here I calculate the 1st quantile, 3rd quantile,median,mean and standard deviation

#for set1 which is for 0.8

quantile1set3 <- quantile(set3,.25)

quantile3set3 <- quantile(set3,.75)

median3 <- median(set3)

mean3 <- mean(set3)

std3 <- sd(set3)

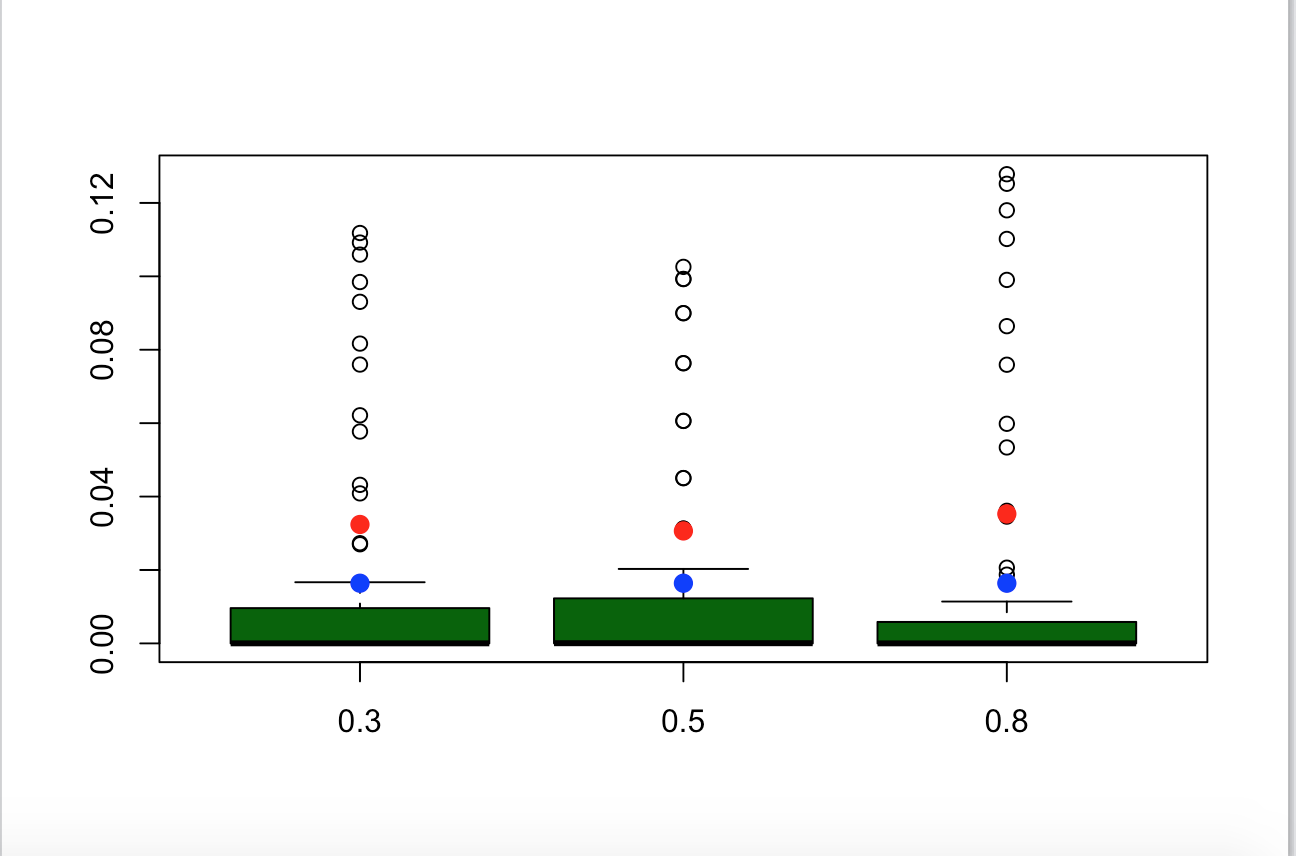
#Here I do the boxplot for distributions on y against the probabilities on x axis. I use points to denote mean and std dev on the plot

boxplot(set1,set2,set3,names=c("0.3","0.5","0.8"),col = "dark green")

points(1:3,c(mean1,mean2,mean3),pch=19,col="blue",cex=1.2)

points(1:3,c(std1,std2,std3),pch=19,col="red",cex=1.2)

#In the diagram below blue dots stand for the mean points and the red dots stand for standard deviations points on the plot



#Improvements :I could define a function and pass the dataframe as output to derive the fields to do the boxplot

#hdrs <- c("1st Quartile", "3rd Quartile", "Median", "Mean", "SD")

#ds <- c(quantile1set1,quantile3set1,median1,mean1,std1)

#df1 <- as.data.frame(ds,hdrs)

#df1 <- data.frame(quantile1set1=quantile1set1,quantile3set1=quantile3set1,median1=median1,mean1 = mean1,std1=std1)

#labels(df1)

**Problem 2**. Finish the plot of the correlation between waiting times and durations of Old Faithful data. Recreate the scatter plot of waiting vs. duration times. As we mentioned in class, the best linear assessment in the sense of the least squares fit of a relationship (proportionality) between two or many variables can be achieved with R function lm(). lm stands for the linear model. The first argument of lm() is called formula accepts a model which starts with the response variable, waiting in our case, followed by a tilde (symbol ~, read as “is modeled as”) followed by the (so called Wilkinson-Rogers) model on the right. In our case we simply assume that waiting time is proportional to the duration time and that “model” reads: formula = waiting ~ duration. The second argument of function lm() is called data and, in our case, will take value faithful, the data set containing our data. Store the result of function lm() in a variable. The name of that variable is not essential. Call it model. Print the variable. The first component of that variable is the intercept of calculated line with the vertical axis (waiting, here) and the second is the slope of the line. Convince yourself that line with those parameters will truly lie on your graph. Function abline() adds a line to the previously created graph. Next, pass the variable model to the function abline(). Make that line somewhat thicker and blue. Use help(functionName) to find details about invocations of both lm() and abline() functions.

**(20%)**

#Referred slide 83

#get the duration and the waiting

duration <- faithful$eruptions

waiting <- faithful$waiting

#Create the scatter plot

plot(duration, waiting, xlab="Eruption duration",

ylab="Time waited")

#used lm to generate linear relationships between the two variables and stored it in variable model

model <- lm(waiting ~ duration,faithful)

#print model

model

#Printed values below for intercept and slope

#Call:

# lm(formula = waiting ~ duration, data = faithful)

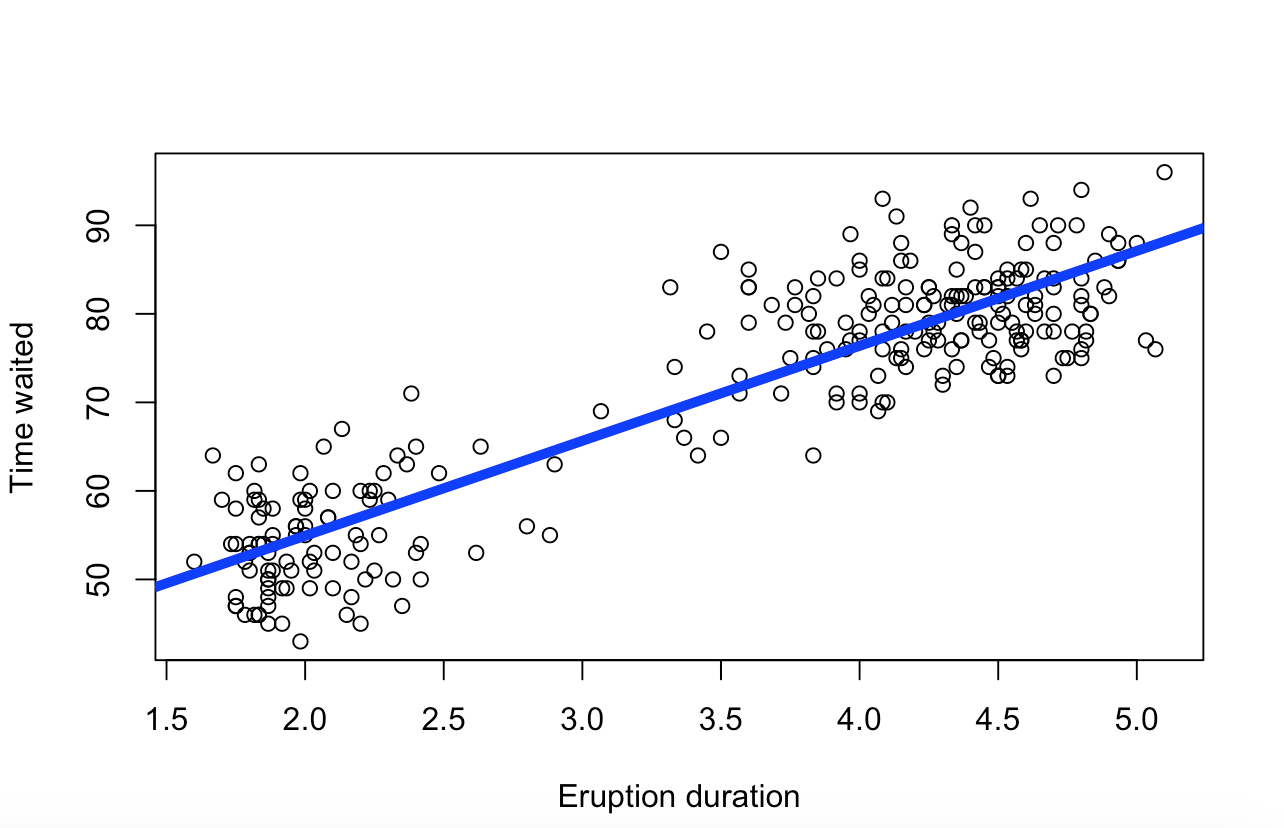
#Coefficients:

# (Intercept) duration

#33.47 10.73

#Used abline to add the line, col to color it blue and lwd to make it thicker

abline(model,col="blue",lwd=5)



**Problem 3**. Calculate the covariance matrix of the faithful data. Determine the eigenvalues and eigenvectors of that matrix. Demonstrate that two eigenvectors are mutually orthogonal. Demonstrate that the eigenvector with the larger eigenvalue is parallel with line discovered by lm() function it the previous problem.

**(15%)**

#Get duration and waiting for faithful dataset

duration = faithful$eruptions; # the eruption

waiting = faithful$waiting;

#The eruption data value pairs with help of function cbind()

matrx <- cbind(duration, waiting)

#do a scatter plot of the data

plot(matrx,xlim = c(1.5,5.5), ylim = c(30,100))

#Get covariance matrix using cov() and then eigen vectors and values

ev <- eigen(cov(matrx))

eigenVector <- ev$vectors

eigenvalue <- diag(ev$values)

#get the slopes

slopeEV1 = eigenVector[2,1]/eigenVector[1,1]

slopeEV1

# [1] 13.20515

# This is nearly equal to the one discovered by lm() (10.73),so its almost parallel

#dot product of two orthogonal vectors is 0

eigenVector[1,] %\*% eigenVector[2,]

# [,1]

#[1,] 0

slopeEV2 = eigenVector[2,2]/eigenVector[1,2]

slopeEV2

#[1] -0.07572801

#orthogonal vectors have slopes that are negative reciprocals of each other

all.equal(slopeEV2,-1/slopeEV1)

#[1] TRUE

**Problem 4.** You noticed that eruptions clearly fall into two categories, short and long. Let us say that short eruptions are all which have duration shorter than 3.1 minute. Add a new column to data frame faithful called type, which would have value ‘short’ for all short eruptions and value ‘long’ for all long eruptions. Next use boxplot() function to provide your readers with some basic statistical measures for waiting. In a separate plot present the box plot for duration times. Please note that boxplot() function also accepts as its first argument a formula such as waiting ~ type, where waiting is the numeric vector of data values to be split in groups according to the grouping variable type. The second argument of function boxplot() is called data, which in our case will take the name of our dataset, i.e. faithful. Find a way to add meaningful legends to your graphs. Subsequently, present both boxplots on one graph.

# check for eruptions less than 3.1 and mark them as shot in type column that I add to the faithful dataframe

faithful$type <- ifelse(faithful$eruptions < 3.1, 'short',"long")

head(faithful)

#do boxplots for waiting and eruptions

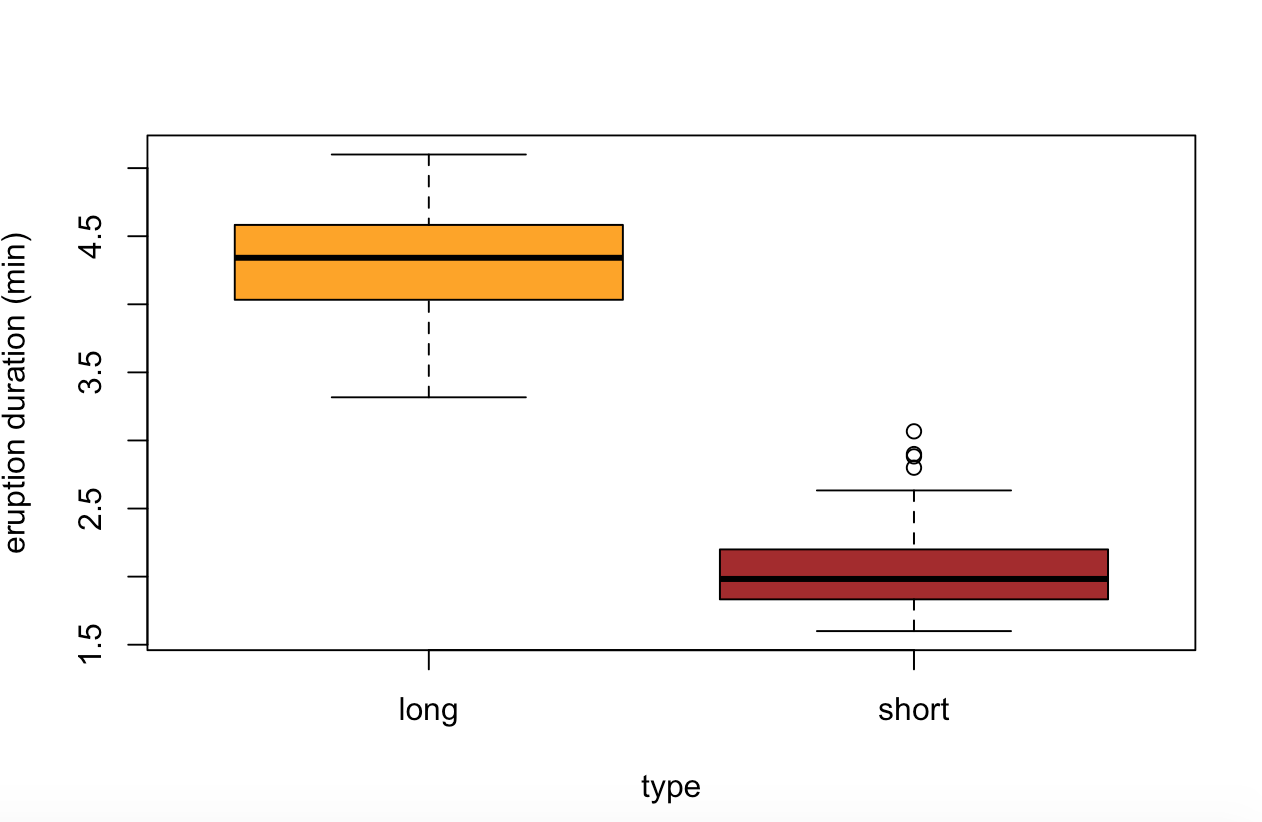
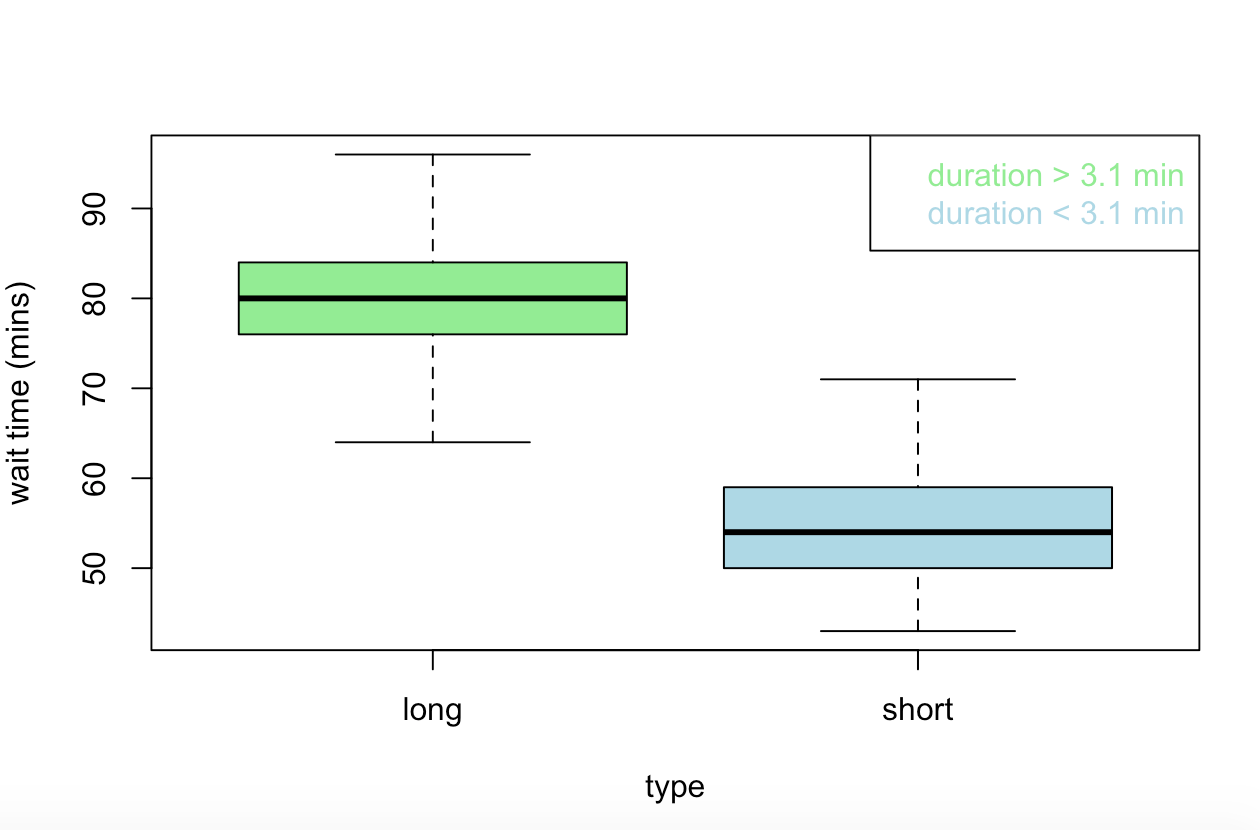
duration = faithful$eruptions; # the eruption

waiting = faithful$waiting;

boxplot(waiting ~ type,faithful,col = c("light green","light blue"),xlab="type", ylab = "wait time (mins)")

legend("topright", c("duration > 3.1 min", "duration < 3.1 min"), text.col = c("light green", "light blue"))

boxplot(eruptions ~ type,faithful,col = c("orange","brown"),xlab = "type", ylab = "eruption duration (min)")



#Referred http://www.statmethods.net/management/reshape.html

require(reshape2)

#melting by id variable "type" (long or short) to make data into long form

faithful.melt <- melt(faithful, id.var = "type")

help(melt)

faithful.melt

# type variable value

#1 long eruptions 3.600

#2 short eruptions 1.800

#3 long eruptions 3.333

#4 short eruptions 2.283

#327 short waiting 54.000

#328 long waiting 83.000

#329 long waiting 71.000

#330 short waiting 64.000

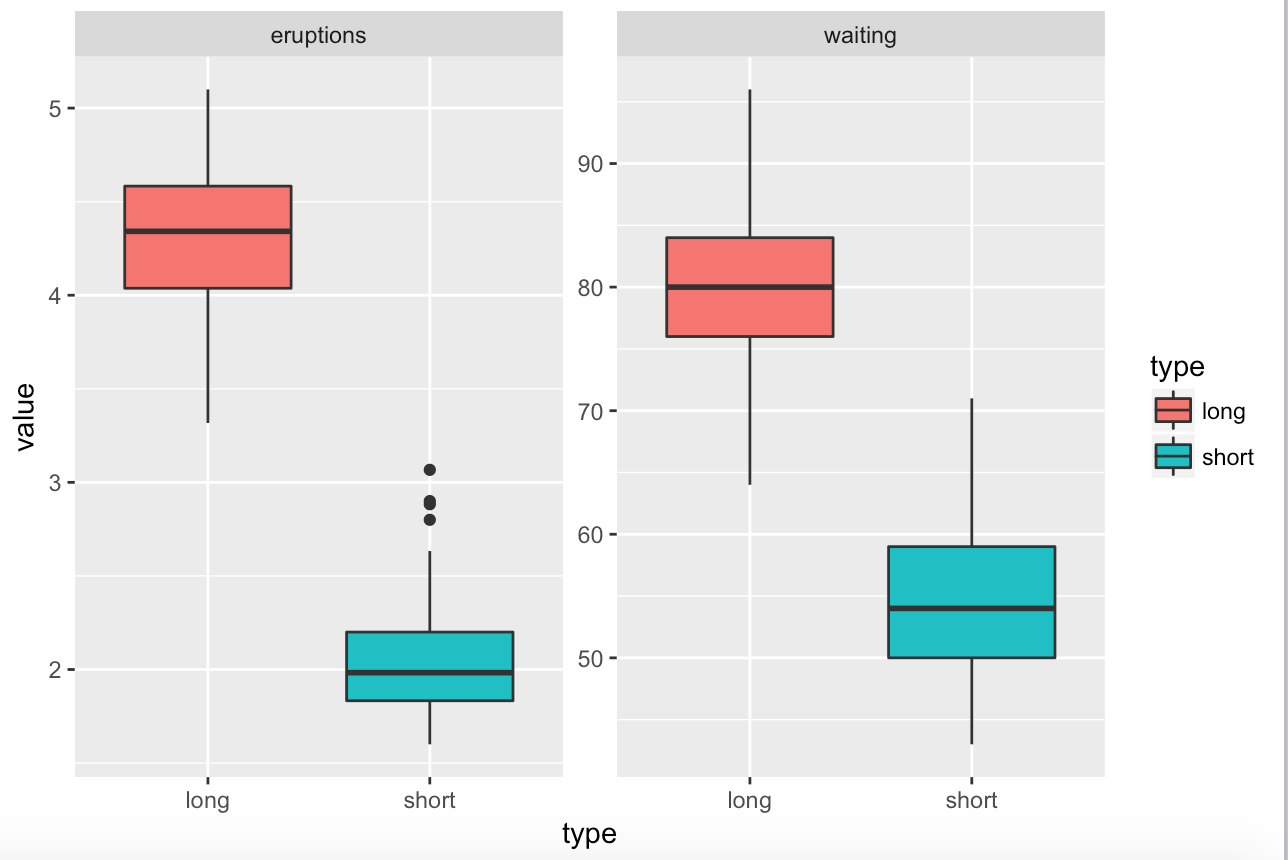
#Now I plot the data using ggplot

require(ggplot2)

faithful.plot <- ggplot(data = faithful.melt, aes(x=type, y=value)) + geom\_boxplot(aes(fill=type))

#facet the data using variable from melt above , variable is type(long/short) to show in the same frame

faithful.plot + facet\_wrap( ~ variable, scales="free")



**(20%)**

**Problem 5.** Create a matrix with 40 columns and 100 rows. Populate each column with random variable of the uniform distribution with values between -1 and 1 (symmetric around zero). Let the distribution for each column appear like the one on slide 92 of the lecture note, except centered around zero. Present two distributions contained in any two randomly selected columns of your matrix on two separate plots. Convince yourself that generated distributions are (close to) uniform.

**(15%)**

# created the matrix with 40 cols and 100 rows. I use runif to get unformly distributed set of values

# between -1 and 1. I make use of cbind after iterating over incremental column numbers and bind them

#with the values generated by runif

matrx <- numeric()

for (i in 1:40){

rand <- runif(100, -1, 1)

matrx <- cbind(matrx, rand)

}

colnames(matrx) <- 1:40

matrx

#1 2 3 4 5 6 7

#[1,] -0.541167965 -0.729353357 0.693015029 -0.187063988 -0.838273645 0.908327759 0.43016205

#[2,] 0.332389109 0.442808316 0.252919568 0.066207462 -0.933559764 -0.347528447 -0.63381283

#[3,] 0.295285904 0.940982255 0.907982845 0.938511828 0.020581467 0.087264028 0.85391076

#[4,] 0.826400214 0.824806898 0.999149071 -0.631431725 0.377455214 0.679243476 0.14695578

#using col 30 as random column plotted histogram for distribution in the column

#hist(x[,30])

#using col 40 as random column plotted histogram for distribution in the column

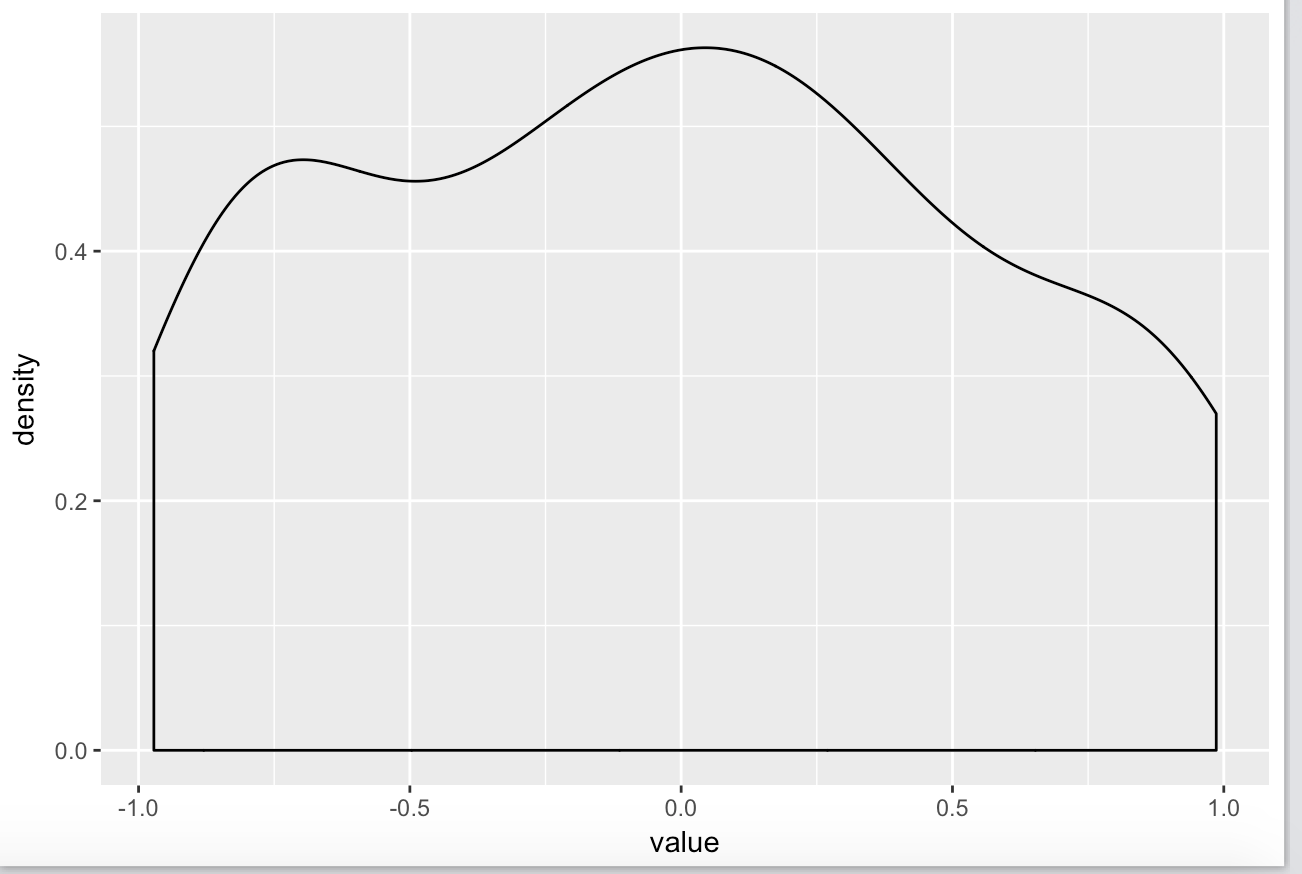
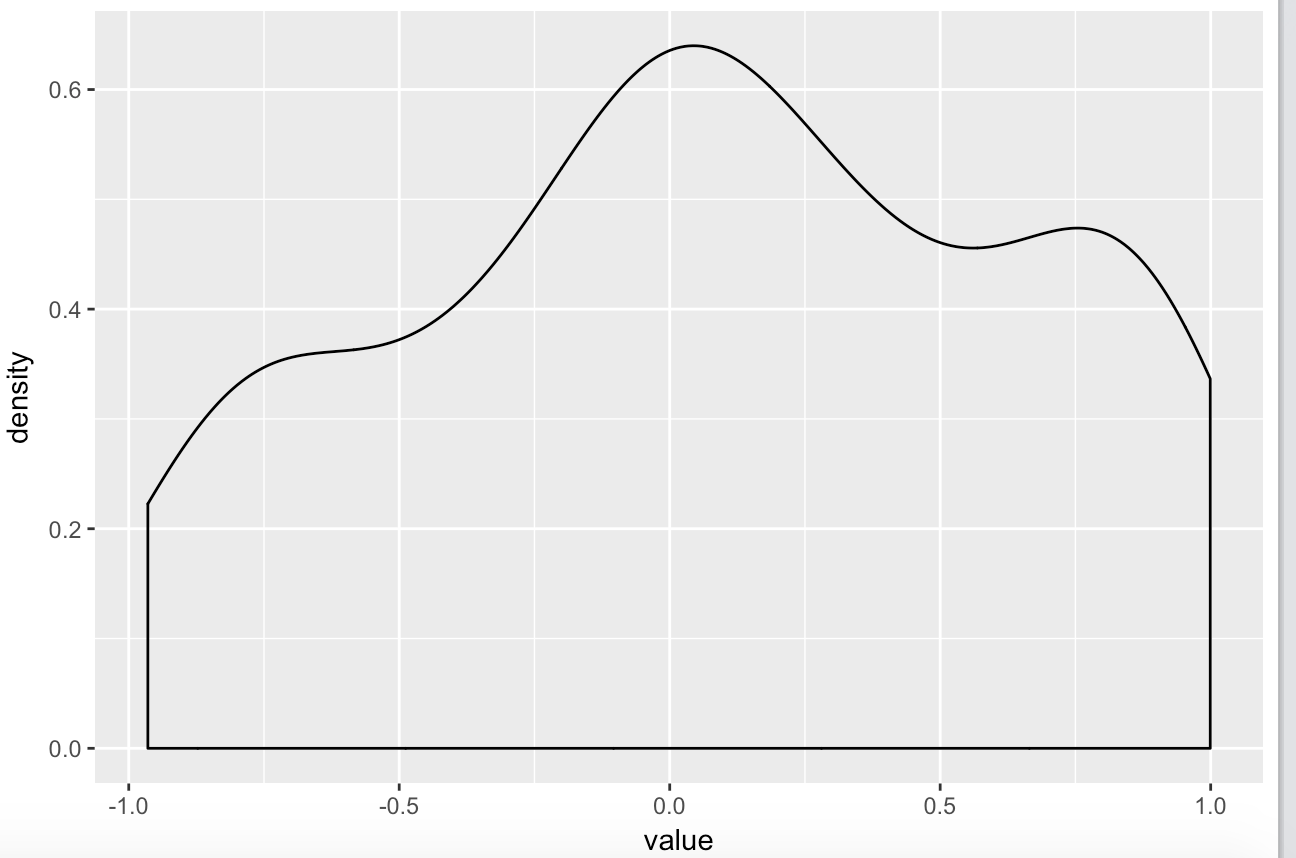
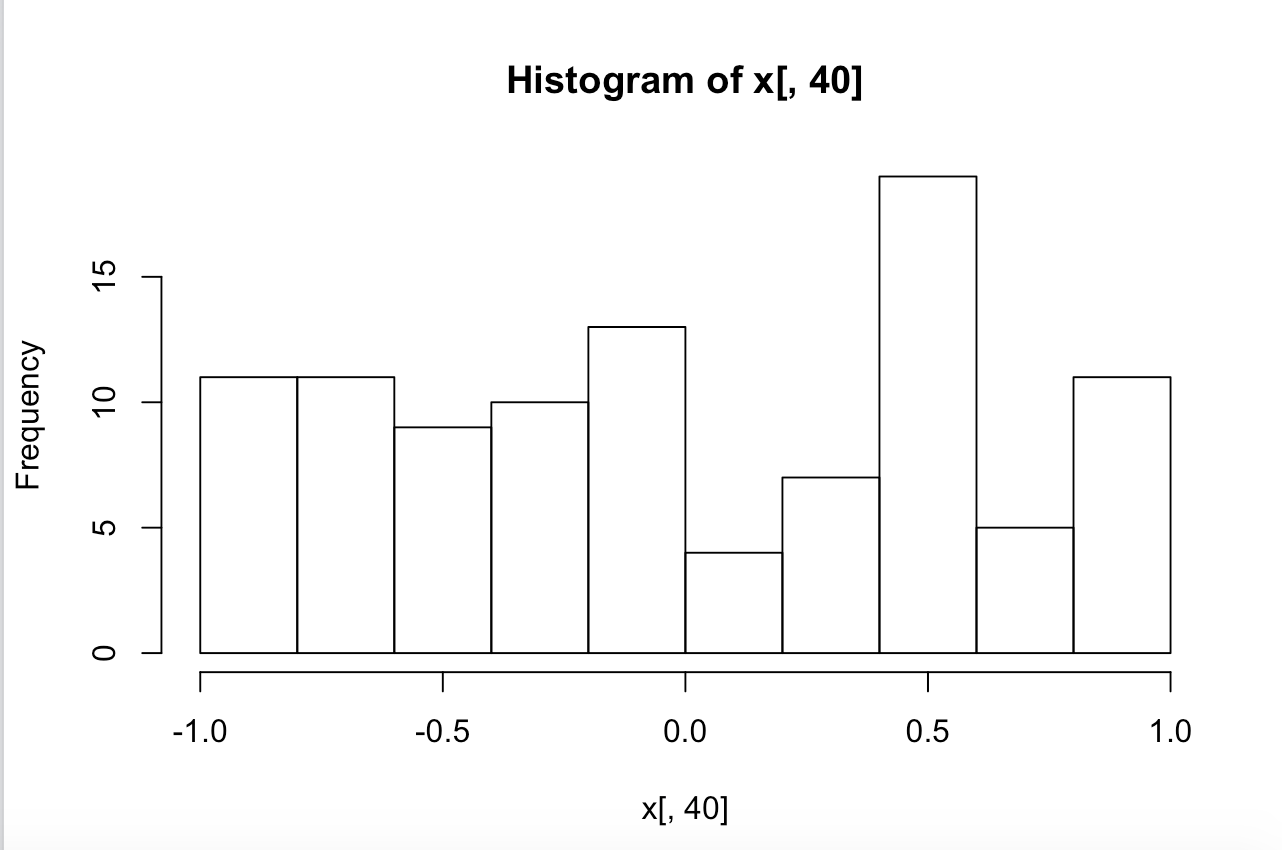
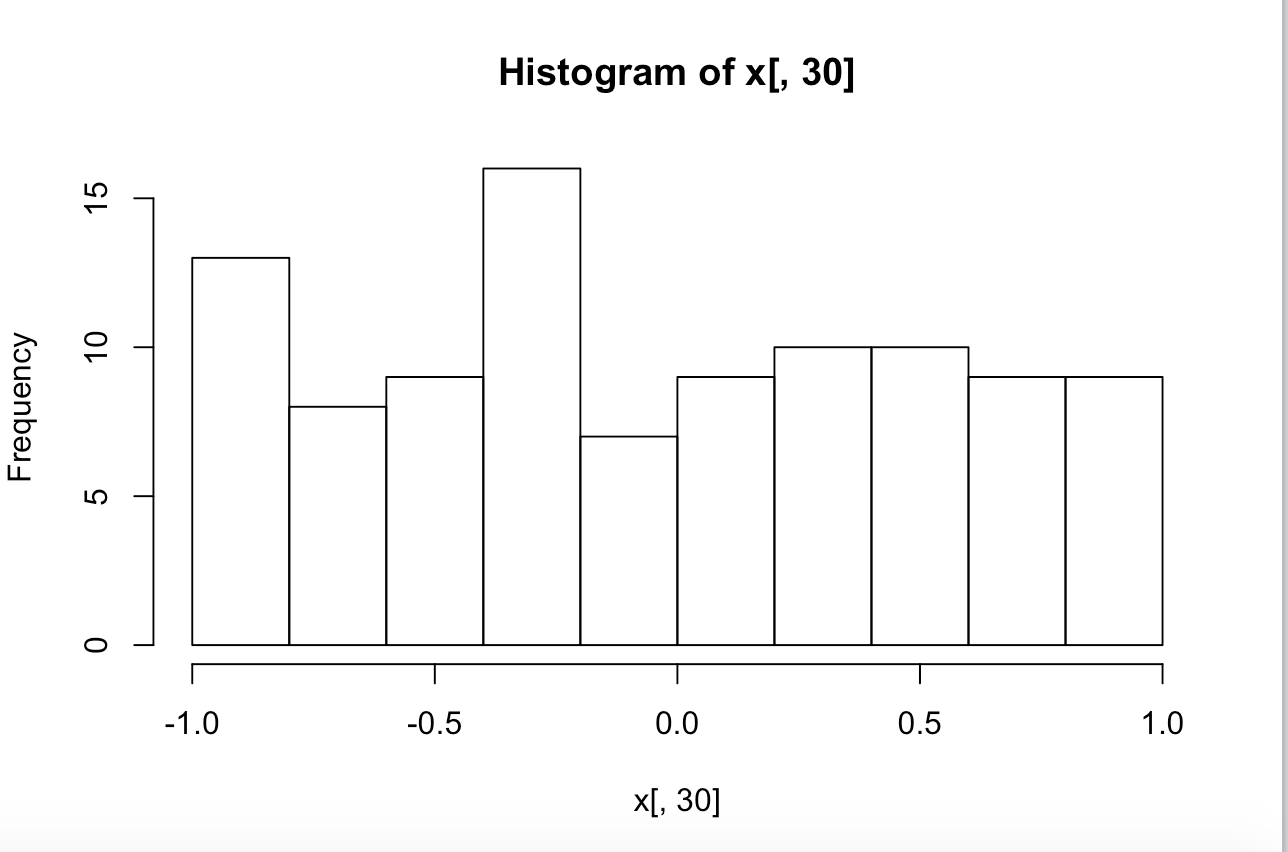
#hist(x[,40])

#using col 30 as random column and creating plot for it using ggplot

ggplot(melt(matrx[,30]), aes (value)) +geom\_density()

#using col 40 as random column and creating plot for it

ggplot(melt(matrx[,40]), aes (value)) +geom\_density()



**Problem 6**. Start with your matrix from problem 5. Add yet another column to that matrix and populate that column with the sum of original 40 columns. Create a histogram of values in the new column showing that the distribution resembles the Gaussian curve. Add a true, calculated, Gaussian curve to that diagram with the parameters you expect from the sum of 40 random variables of uniform distribution **(15%)**

#First I used apply to find the sum of the columns of the matrix.(referred slide 48)

#

sum.cols <- apply(matrx,1,sum)

sum.cols

#Next I bind the vector generated from previous step to the original matrix

mtrx.with.sum <- cbind(matrx,sum.cols)

#mtrx.with.sum

#colnames(mtrx.with.sum)

#get the sum column and separate it out

sum.mtrx.column <- mtrx.with.sum[,41]

#calculate the mean and std deviation for the normal distribution

sum.mean <- mean(sum.mtrx.column)

sum.sd <- sd(sum.mtrx.column)

#Here I set the X axis

x <- seq(min(sum.mtrx.column), max(sum.mtrx.column), length = length(sum.mtrx.column))

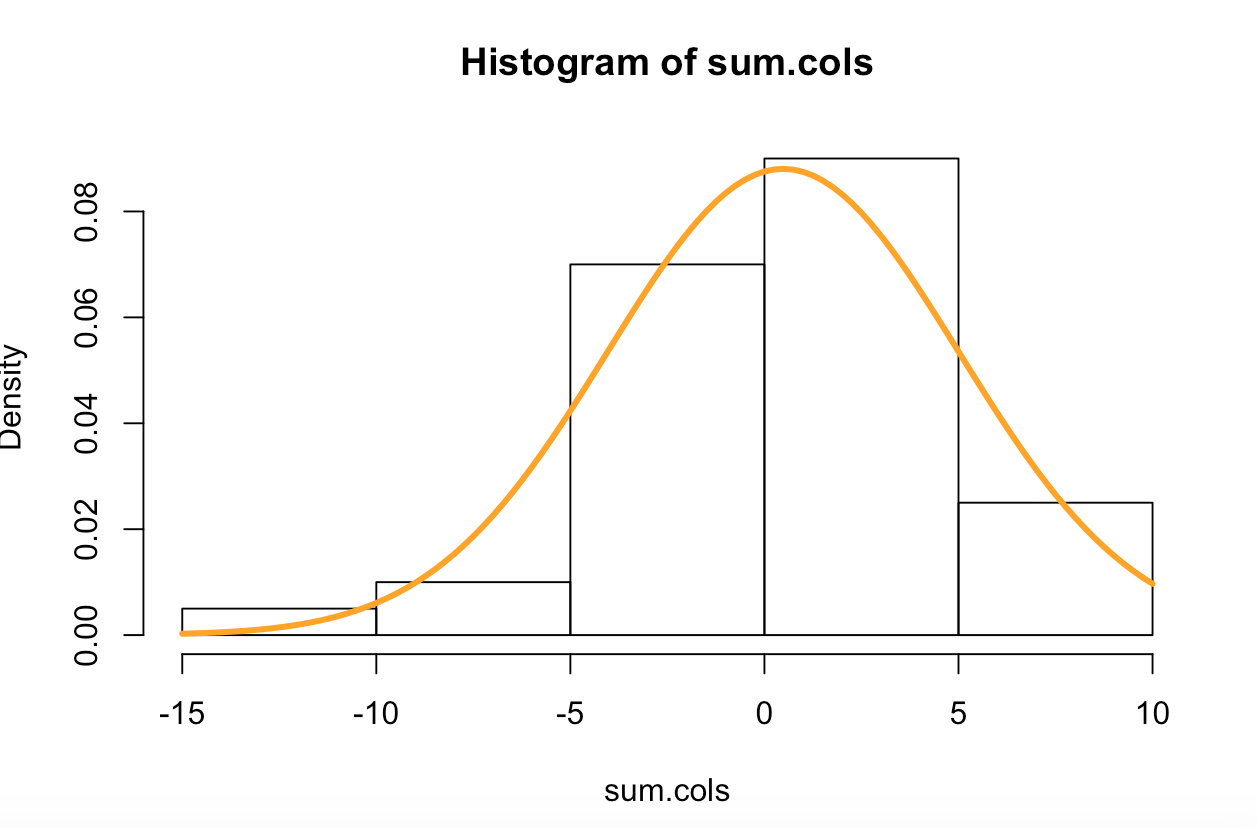
#Draw the histogram for the sums column

hist(sum.cols,prob=TRUE)

#Here I get the standard normal distribution for the sum column using dnorm,

#and pass in the required parameters from previous steps, I then draw a Gaussian curve

curve(dnorm(x, mean=mean(sum.cols), sd=sd(sum.cols)), add=TRUE, col = "orange",lwd=3)



SUBMISSION INSTRUCTIONS:

Your main submission should be an MS Word document containing your code, results produced by that code and brief textual descriptions of what you did and why. Typically, you copy important snippets of your code and the results into this Word document. Describe the purpose of every code snippet and the significance of the results. Start with the text of this homework assignment as the template. Please add any other files that you might have used or generated. Please do not provide ZIP or RAR or any other archives. Canvas cannot open them and they turn into a nuisance for us.