## Donna Strok

## E-63 Big Data Analytics

## Assignment 01

## 

### Handed out: 09/01/2017

### Due by 11:59AM EST on Saturday, 09/09/2017

**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%)**

1. Set x axis

> x <- seq(1,60,by = 1)

1. Set y axis variables for all three altered coin scenarios.

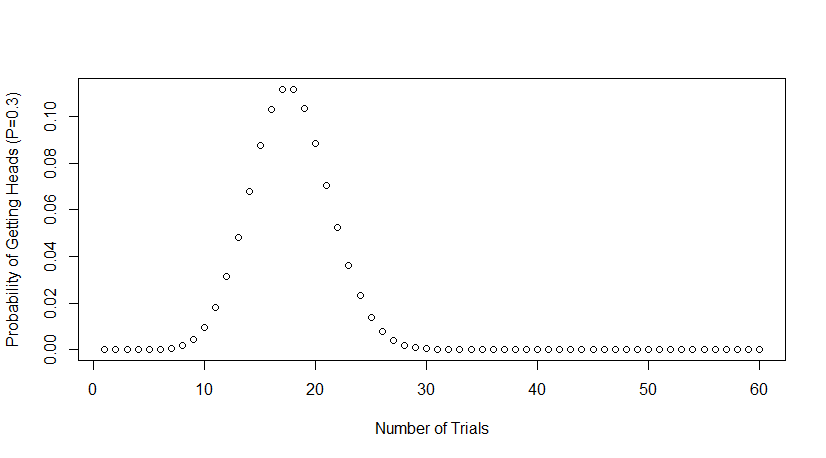
> y1 <- dbinom(x,59,0.3)

> y2 <- dbinom(x,59,0.5)

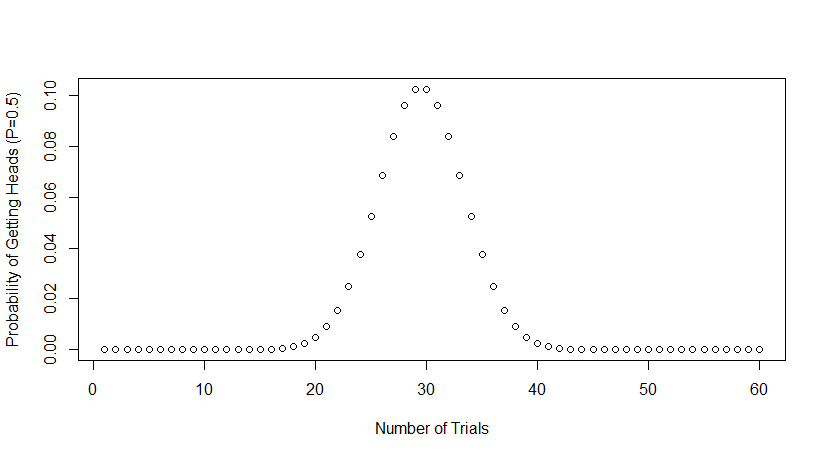
> y3 <- dbinom(x,59,0.8)

1. Plot all three on separate graphs

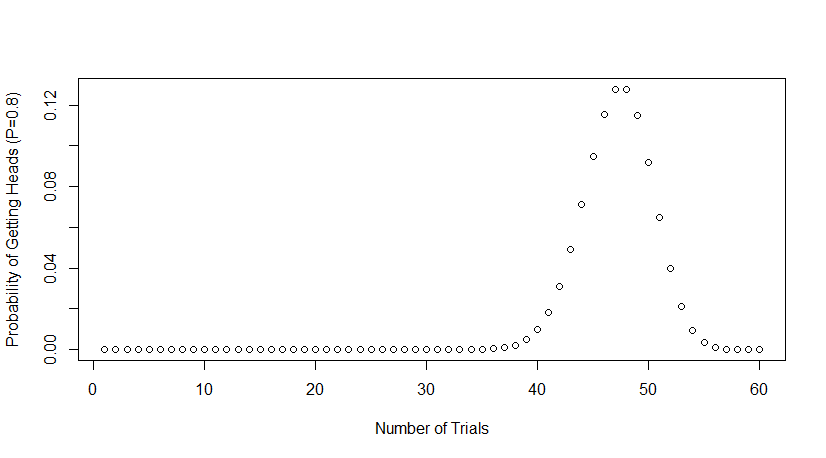
> plot(x, y1, xlab="Number of Trials", ylab="Probability of Getting Heads")



|  |
| --- |
| > plot(x, y2, xlab="Number of Trials", ylab="Probability of Getting Heads (P=0.5)") |
|  |
| |  | | --- | |  | |



> plot(x, y3, xlab="Number of Trials", ylab="Probability of Getting Heads (P=0.8)")



1. Plot all three y-values on same graph

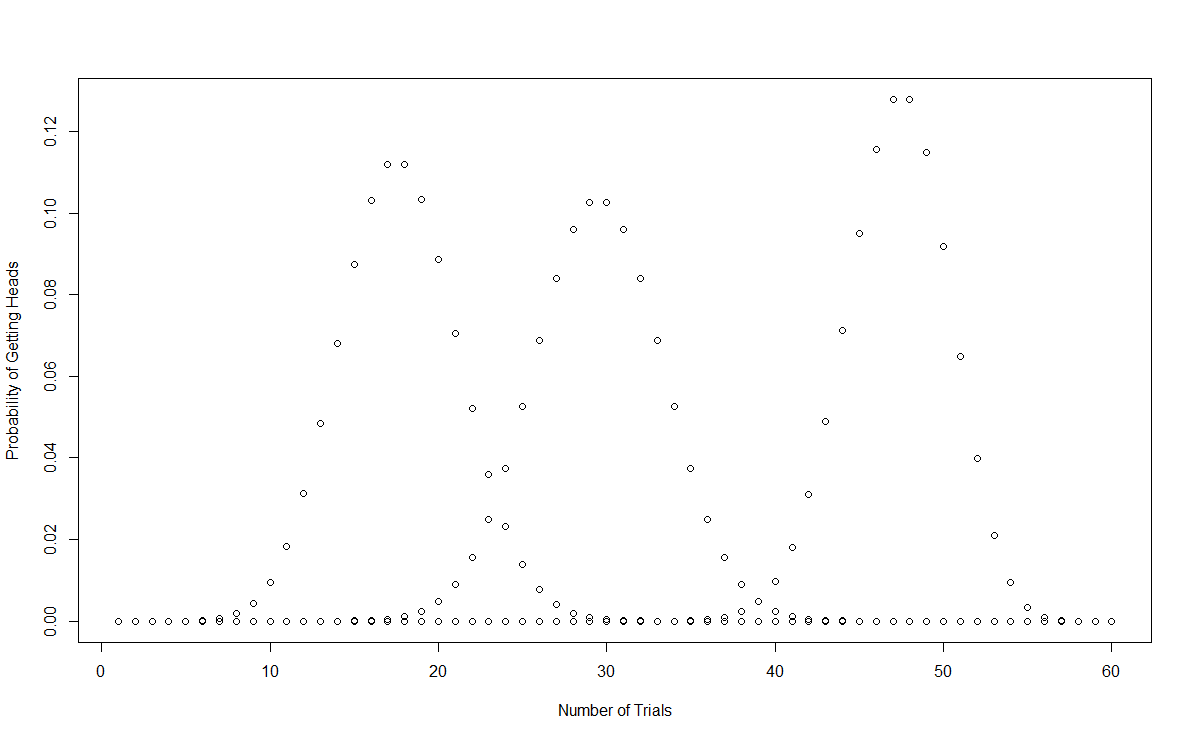
> plot(x, y1, ylim=range(c(y1,y3)), xlab="Number of Trials", ylab="Probability of Getting Heads")

> par(new=TRUE)

> plot(x, y2, ylim=range(c(y1,y3)), axes = FALSE, xlab = "", ylab = "")

> par(new=TRUE)

> plot(x, y3, ylim=range(c(y1,y3)), axes = FALSE, xlab = "", ylab = "")



1. For each value of p, determine 1st Quartile, median, mean, standard deviation and the 3rd Quartile. The boxplot() in R already has 1st Quartile, median, and 3rd Quartile values. The only thing missing is mean and standard deviation. So those will be displayed as colored dots.

#pick random sample size

> samp <- sample(x,1)

> samp

[1] 13

# set the values of each probability

> y1a <- rbinom(x,samp,0.3)

> y2a <- rbinom(x,samp,0.5)

> y3a <- rbinom(x,samp,0.8)

# Plot them

> boxplot(y1a, y2a, y3a ,horizontal=FALSE)

# Add STD and Mean. STD = Red, Mean = Blue

> points(x = 1,y=sd(y1a),pch=19,col="red")

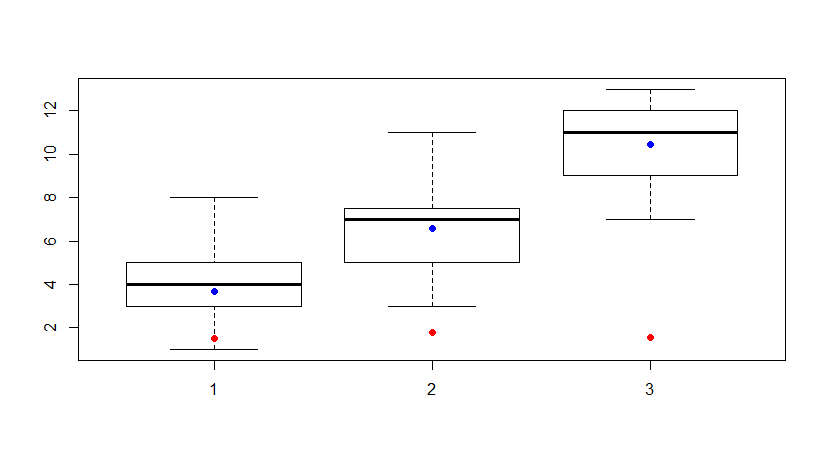
> points(x = 2,y=sd(y2a),pch=19,col="red")

> points(x = 3,y=sd(y3a),pch=19,col="red")

> points(x = 1,y=mean(y1a),pch=19,col="blue")

> points(x = 2,y=mean(y2a),pch=19,col="blue")

> points(x = 3,y=mean(y3a),pch=19,col="blue")



**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.

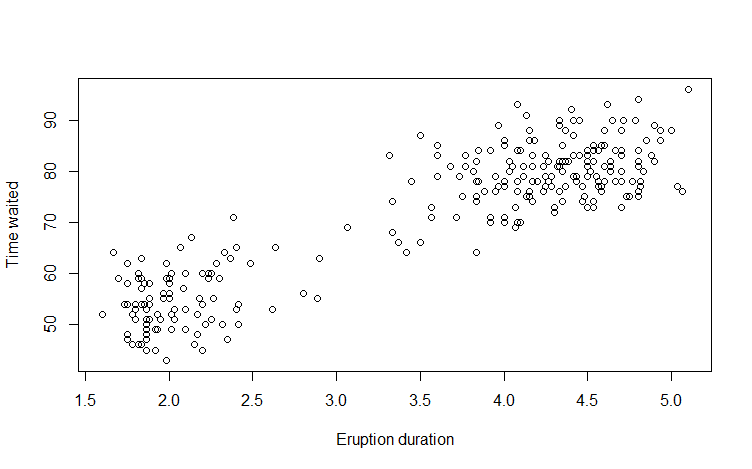
**(20%)**

1. Recreate the scatter plot of waiting vs. duration times.

> duration = faithful$eruptions;

> waiting = faithful$waiting

> plot(duration, waiting, xlab="Eruption duration", ylab="Time waited")



1. Store model and formula values

> formula = waiting ~ duration

> model = lm(formula = formula, data = faithful)

> print(model)

Call:

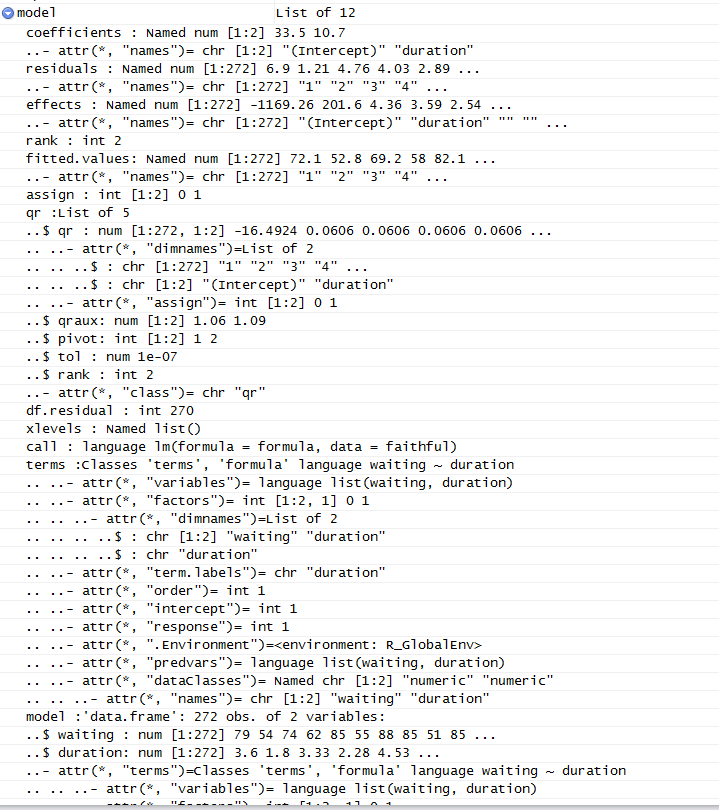
lm(formula = formula, data = faithful)

Coefficients:

(Intercept) duration

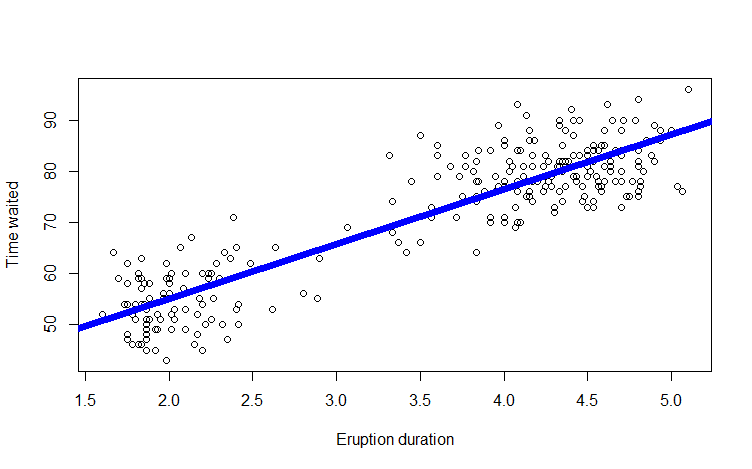
33.47 10.73

But there are more data presented in the global variables page. Here is a screenshot that shows more behind the “model” variable.



1. Add blue line to the graph

> abline(model, col = 4, lwd = 7)



**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. Examine whether the eigenvector with the larger eigenvalue is parallel with line discovered by lm() function it the previous problem.

**(15%)**

1. Calculate the covariance matrix of the “faithful data.

> waiting = faithful$waiting

> duration = faithful$eruptions

> faithful\_matrix = cbind(waiting,duration)

> cov(faithful\_matrix)

waiting duration

waiting 184.82331 13.977808

duration 13.97781 1.302728

1. Determine the eigen values and eigen vectors.

> eigen(cov(faithful\_matrix))

eigen() decomposition

$values

[1] 185.8818239 0.2442167

$vectors

[,1] [,2]

[1,] -0.9971449 0.0755118

[2,] -0.0755118 -0.9971449

1. In order to show vectors are orthogonal, the dot product of the two vectors must equal 0

|  |
| --- |
| > eig <- eigen(cov(faithful\_matrix))  > evector <- eig$vectors  > evector[,1] %\*% evector[,2]  [,1]  [1,] 0 |

1. See if the larger eigenvalue has a parallel line with the abline done in previous problem

> eVal1 <- eigen(cM)$values[1]

> eVal2 <- eigen(cM)$values[2]

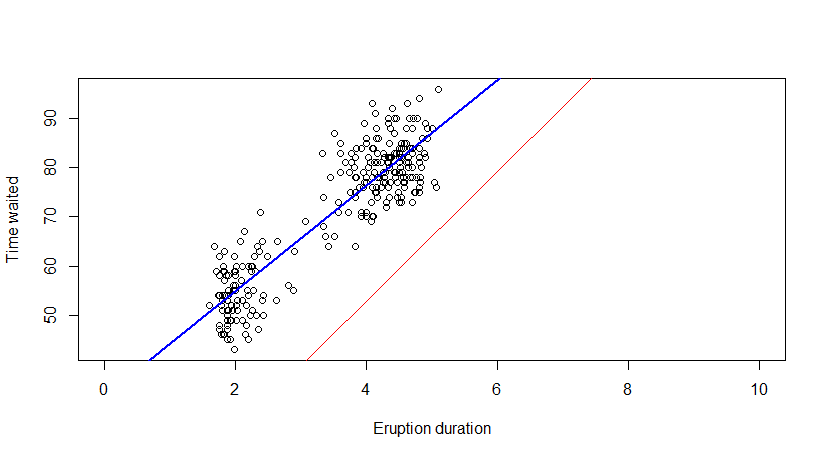
> eVal1 > eVal2

[1] TRUE

> plot(duration, waiting, xlab="Eruption duration", ylab="Time waited", xlim=c(0,10))

> abline(model, col = 4, lwd = 2)

> lines(x= c(0,eVal1\*evector[1,1]), y= c(0, eVal1\*evector[2,1]), col="red")

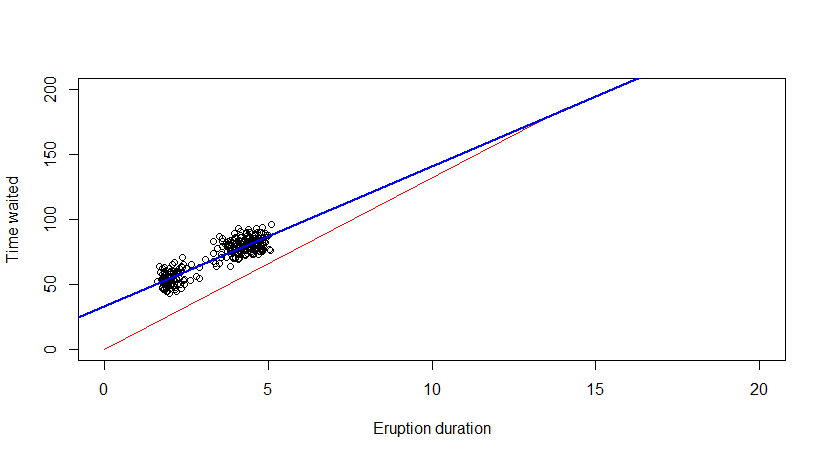


Doesn’t look parallel but let’s zoom out a bit to be sure.

> plot(duration, waiting, xlab="Eruption duration", ylab="Time waited", xlim=c(0,20), ylim=c(0,200))

> lines(x= c(0,eVal1\*evector[1,1]), y= c(0, eVal1\*evector[2,1]), col="red")

> abline(model, col = 4, lwd = 2)

The lines look like they meet however it looks like the eigen vector line stops at the max waiting and duration ccovariance value.

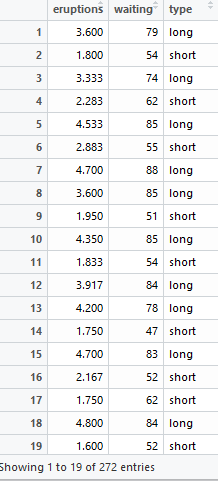
**Problem 4.** You noticed that eruptions clearly fall into two categories, short and long. Let us say that short eruptions are all which have duration shorter than 3.1 minute. Add a new column to data frame faithful called type, which would have value ‘short’ for all short eruptions and value ‘long’ 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.

**(20%)**

1. Add new colum to data frame “faithful” and name it “type”. Each value will either be “short” if shorter than 3.1 minutes or “long” if greater than or equal to for all eruptions

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

> View(faithful)

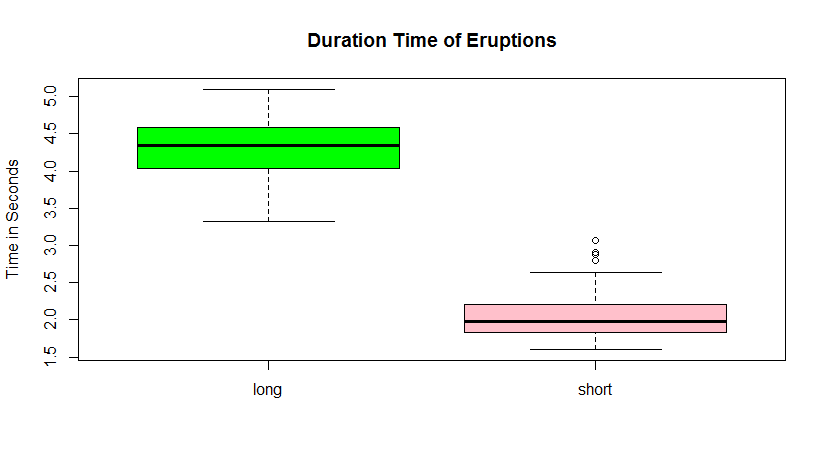


1. Use Box Plot to show basic statistics.

> boxplot(waiting ~ type, data = faithful, main = "Waiting Time Between Eruptions", ylab="Time in Seconds", col= c("green","pink"))



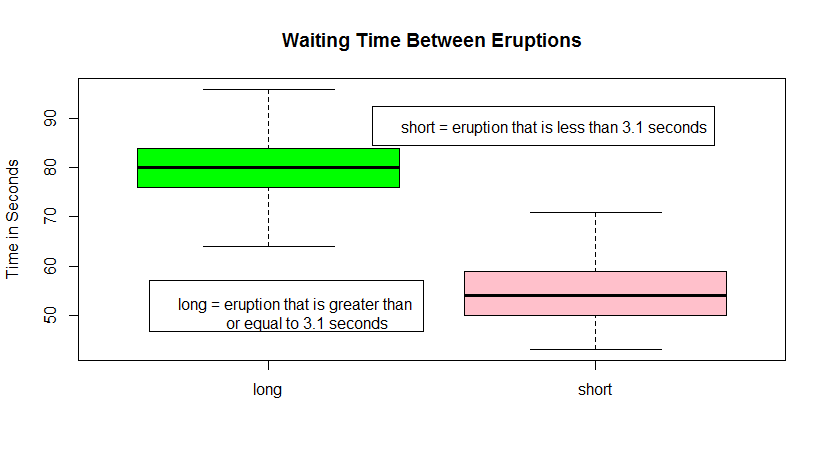
> boxplot(duration ~ type, data = faithful, main = "Duration Time of Eruptions", ylab="Time in Seconds", col= c("green","pink"))



1. Add legends

> legend("bottomleft", "long = eruption that is greater than \n or equal to 3.1 seconds", inset = .10)

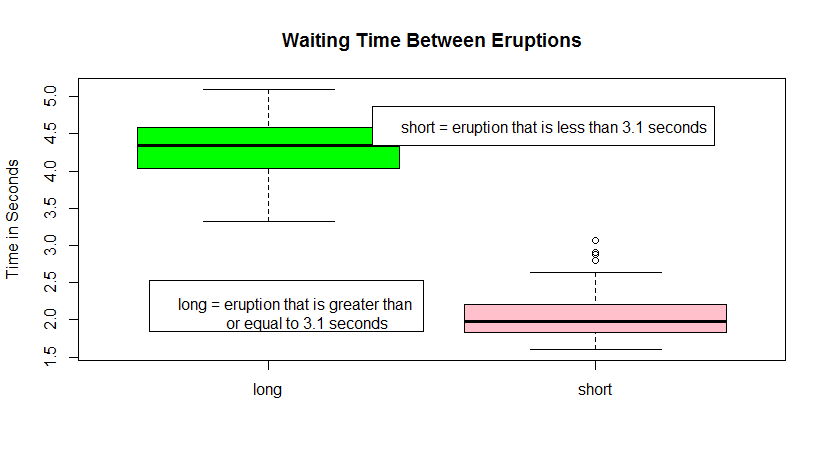
> legend("topright", "short = eruption that is less than 3.1 seconds", inset = .10)



> boxplot(duration ~ type, data = faithful, main = "Waiting Time Between Eruptions", ylab="Time in Seconds", col= c("green","pink"))

> legend("topright", "short = eruption that is less than 3.1 seconds", inset = .10)

> legend("bottomleft", "long = eruption that is greater than \n or equal to 3.1 seconds", inset = .10)

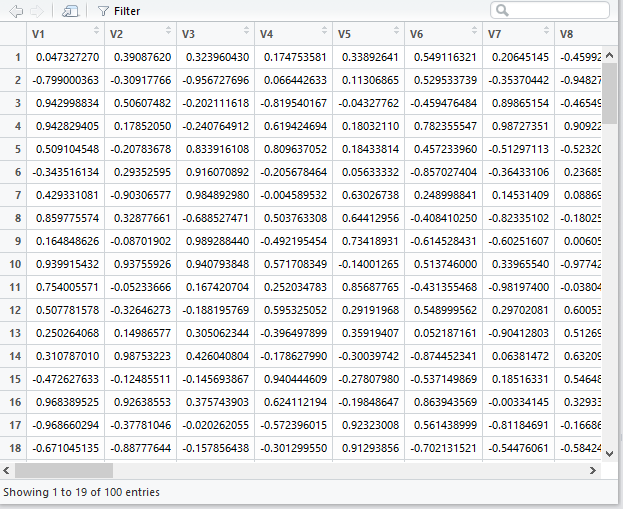


**Problem 5.** Create a matrix with 40 columns and 100 rows. Populate each column with random variable of the uniform distribution with values between -1 and 1 (symmetric around zero). Let the distribution for each column appear like the one on slide 92 of the lecture note, except centered around zero. 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%)**

**1. Creating the matrix with 40 columns and 100 rows with values between -1 and 1.**

> m <- matrix(runif(4000,min=-1,max=1),nrow=100)



**2. Pick two random columns**

> rand\_columns <- sample(1:40, 2, replace=F)

> rand\_columns

[1] 21 19

>

> col1 <- rand\_columns[1]

> col2 <- rand\_columns[2]

> col1

[1] 21

> col2

[1] 19

**3. Present two distributions contained in any two randomly selected columns of your matrix on two separate plots.**

> x1 <- m[,col1]

> x1

[1] 0.324747285 0.794363386 -0.146954033 -0.617341919 0.868468564

[6] 0.979583937 -0.928426730 0.530968652 0.854183334 -0.174696536

[11] 0.004308385 -0.724885581 -0.620996031 -0.095722630 0.941387473

[16] 0.668593673 -0.555343998 0.609494262 0.201359353 0.745277751

[21] -0.939955022 0.542033351 -0.622125422 0.572838253 0.184635441

[26] 0.348997004 -0.256234029 -0.958481105 -0.139260061 0.931651684

[31] -0.998303848 -0.088610842 0.162807093 -0.877294263 0.565667297

[36] 0.581605402 -0.625004950 -0.361543725 -0.238484115 0.419283865

[41] -0.798678590 -0.182870022 0.905364715 0.331263087 0.889946167

[46] -0.702066714 0.917062306 -0.615757965 -0.241422089 -0.993358679

[51] -0.526408319 -0.800252040 -0.717594972 -0.050425591 0.548135507

[56] -0.148375327 0.460785030 -0.006297458 -0.663447484 -0.468850412

[61] 0.320907260 -0.581933103 0.211051341 0.792032300 0.304981324

[66] -0.857305076 -0.733890727 -0.008599647 -0.627548558 -0.574017118

[71] -0.439504930 -0.312451731 -0.798320571 -0.396100755 -0.211793071

[76] 0.867825648 0.855610556 0.852069738 -0.727072505 -0.616490126

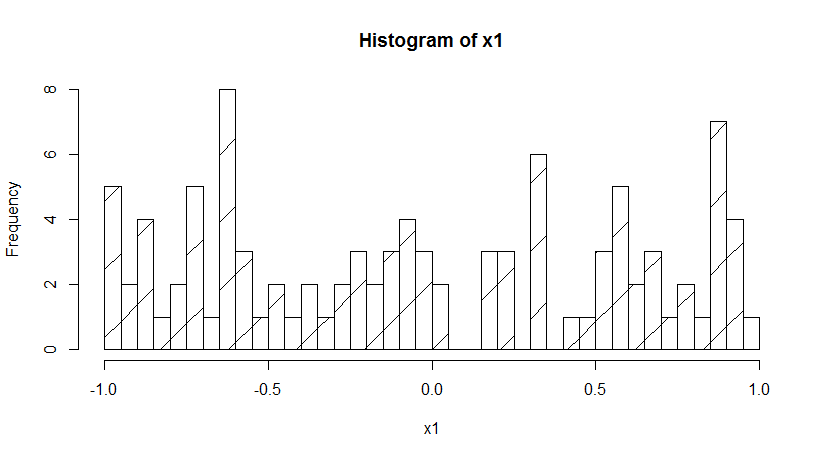
[81] -0.025005741 -0.891447303 -0.270633201 -0.601607916 0.237862127

[86] -0.950865296 0.692277703 0.845038061 0.624740459 0.693355443

[91] -0.087846479 0.560543271 -0.463920918 0.015299359 0.862129238

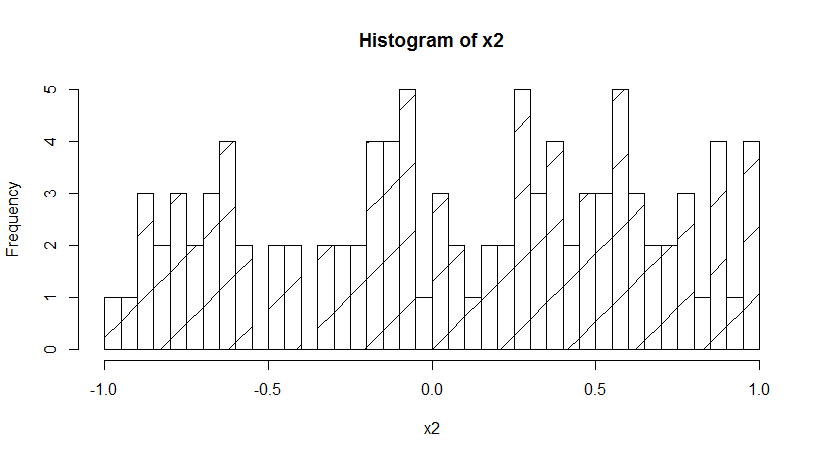
[96] 0.345455918 0.573095249 -0.962867690 0.179936438 -0.895756990

> hist(x1, breaks = 10, density=2)



> x2 <- m[,col2]

> hist(x2, breaks = 50, density=2)

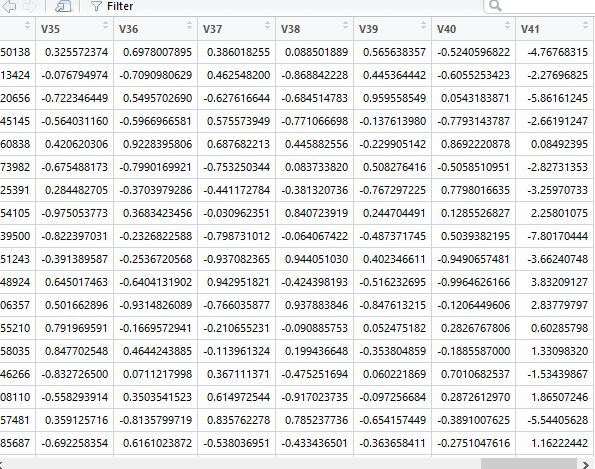


These do not have a normal curve thereby being uniform.

**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%)**

1. Add column with row sum

> m <- cbind(m,rowSums(m[,1:40]))



**2. Create histogram with the new column. Add Gaussian curve to histogram.**

> hist(d, density=20, breaks=50, xlab="Sum Value", main="Gaussian Distribution for Summed Uniform Values", prob=TRUE)

> curve(dnorm(x, mean=mn, sd=std), add=TRUE)

