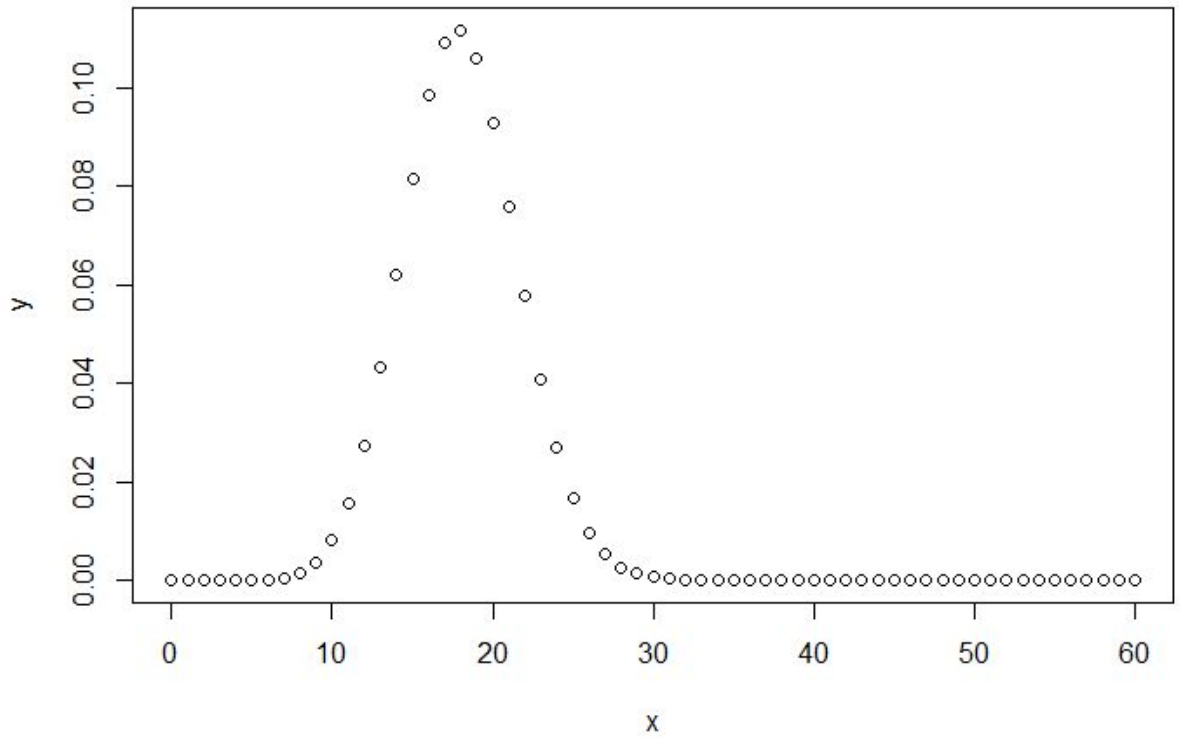
Problem 1. Plot the binomial distribution for p = 0.3, p = 0.5 and p = 0.8 and the total number of trials n = 60 as a function of k the number of successful trials. For each value of p, determine 1st Quartile, median, mean, standard deviation and the 3rd Quartile.Present those values as a vertical box plot with the probability p on the horizontal axis.

p = 0.3

```
> x<-seq(0,60,by=1)
> y<-dbinom(x,60,0.3)
> y
 [1] 5.080219e-10 1.306342e-08 1.651589e-07 1.368460e-06 8.357380e-06 4.011542e-05 1.575963e-04
5.210327e-04 1.479361e-03
[10] 3.663179e-03 8.006662e-03 1.559739e-02 2.729544e-02 4.319278e-02 6.214472e-02 8.167591e-02
9.844865e-02 1.092035e-01
[19] 1.118036e-01 1.059192e-01 9.305760e-02 7.596539e-02 5.771396e-02 4.086579e-02 2.700061e-02
1.666323e-02 9.613404e-03
[28] 5.188186e-03 2.620564e-03 1.239281e-03 5.488246e-04 2.276231e-04 8.840718e-05 3.214807e-05
1.094115e-05 3.483304e-06
[37] 1.036698e-06 2.881940e-07 7.475708e-08 1.807314e-08 4.066457e-09 8.501303e-10 1.648212e-10
2.956925e-11 4.896207e-12
[46] 7.460887e-13 1.042670e-13 1.331069e-14 1.544990e-15 1.621564e-16 1.528904e-17 1.284793e-18
9.530057e-20 6.164996e-21
[55] 3.424998e-22 1.601298e-23 6.127415e-25 1.842832e-26 4.085095e-28 5.934762e-30 4.239116e-32
> plot(x,y)
```



1st quartile, median (2nd quartile), 3rd quartile are shown below

> *quantile(y)*

0%	25%	50%	75%	100%
4.239116e-32	7.460887e-13	8.357380e-06	9.613404e-03	1.118036e-01

> sd(y)

[1] 0.03239755

> *mean(y)*

[1] 0.01639344

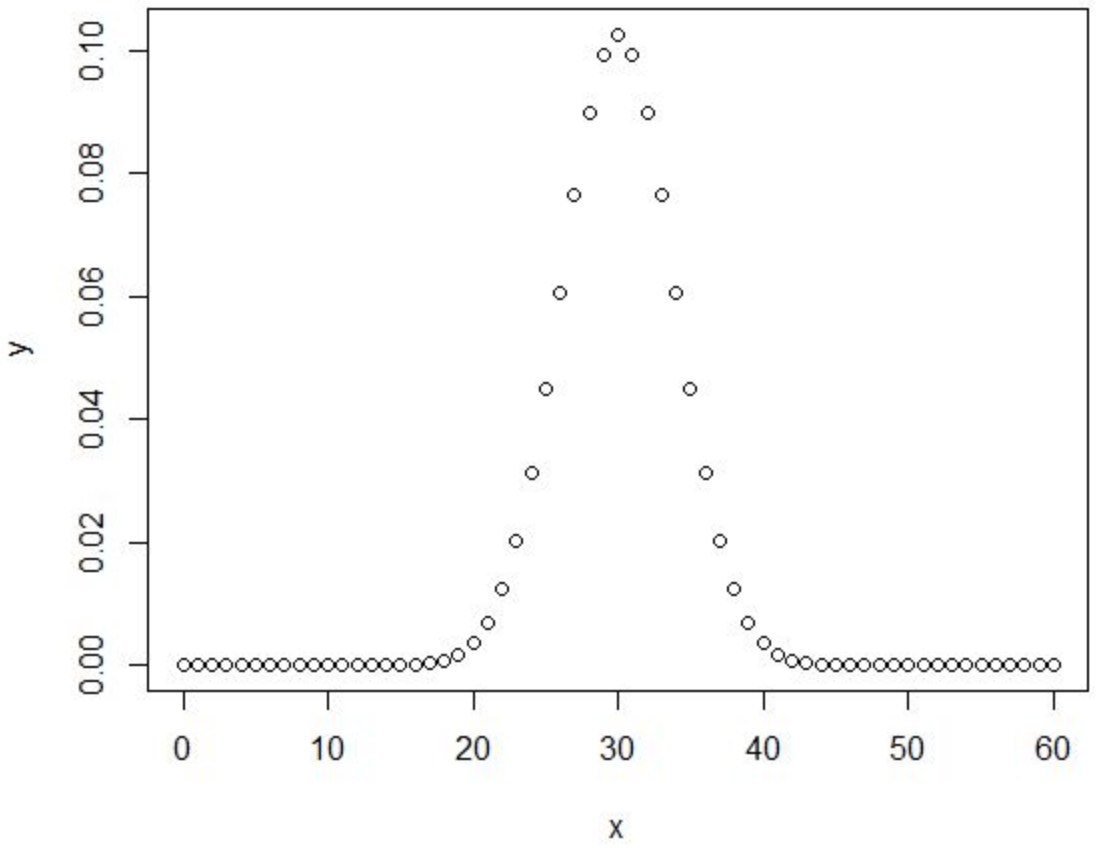
---

p = 0.5

> x<-seq(0,60,by=1)

> y<-dbinom(x,60,0.5)

> plot(x,y)

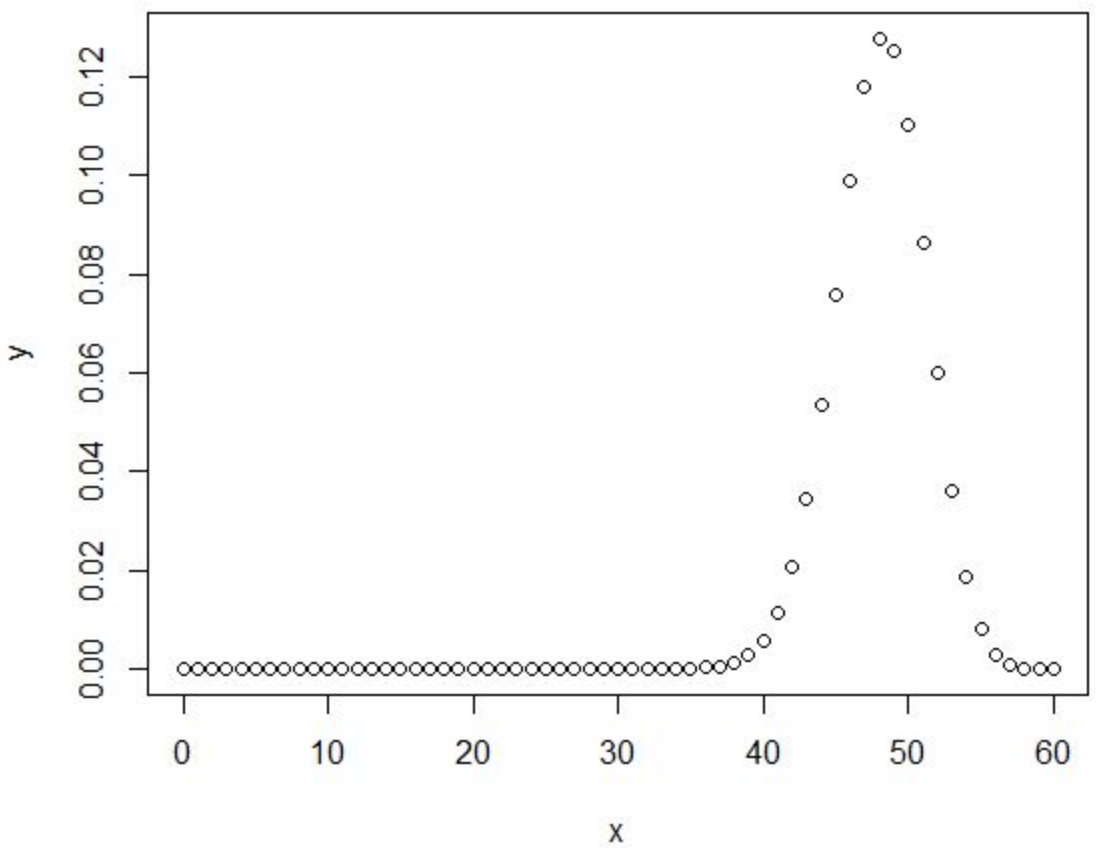



```
> quantile(y)
        0%         25%         50%         75%        100%
8.673617e-19 3.349811e-10 4.613852e-05 1.227688e-02 1.025782e-01
> sd(y)
[1] 0.03062992
> mean(y)
[1] 0.01639344
```

---

p = 0.8

```
> x<-seq(0,60,by=1)
> y<-dbinom(x,60,0.8)
> plot(x,y)
```

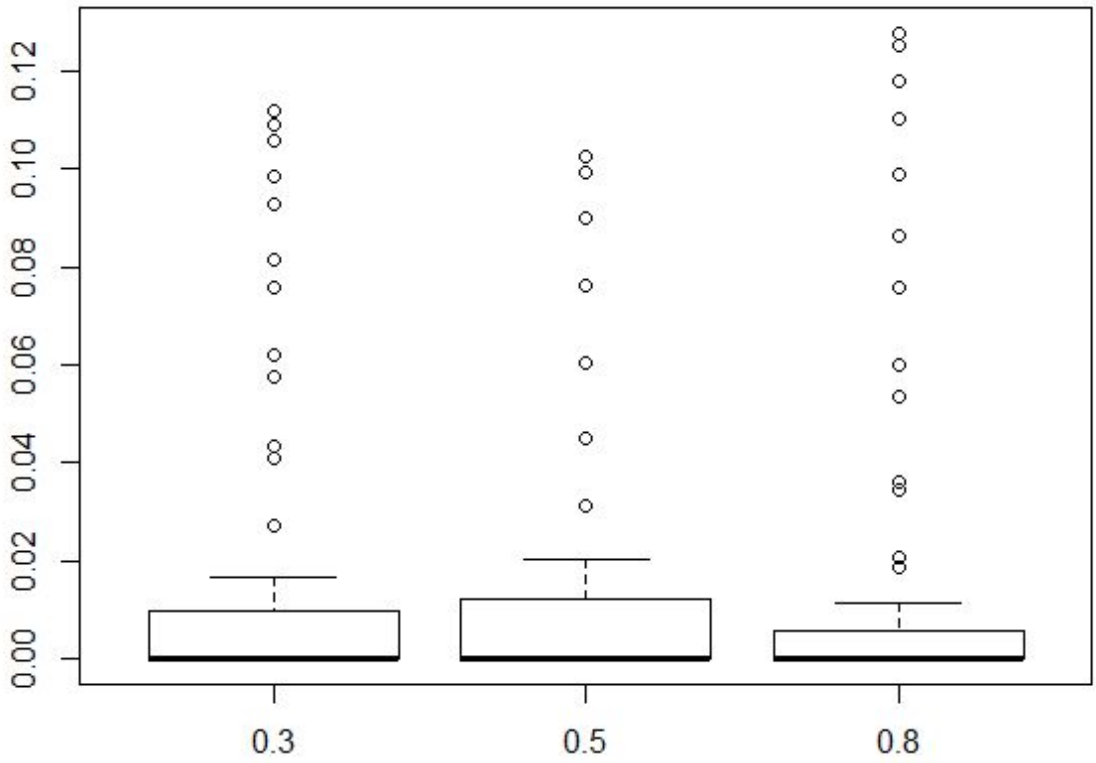



```
> quantile(y)
        0%         25%         50%         75%        100%
1.152922e-42 6.585109e-20 1.572006e-07 5.842579e-03 1.278228e-01
> sd(y)
[1] 0.03527981
> mean(y)
[1] 0.01639344
>
```

BoxPlots (p = 0.3, 0.5, 0.8)

```
> y1<-dbinom(x,60,0.3)
> y2<-dbinom(x,60,0.5)
> y3<-dbinom(x,60,0.8)
> df<-data.frame(y1, y2, y3)
```

> names(df) <- c(0.3, 0.5, 0.8)
> boxplot(df)



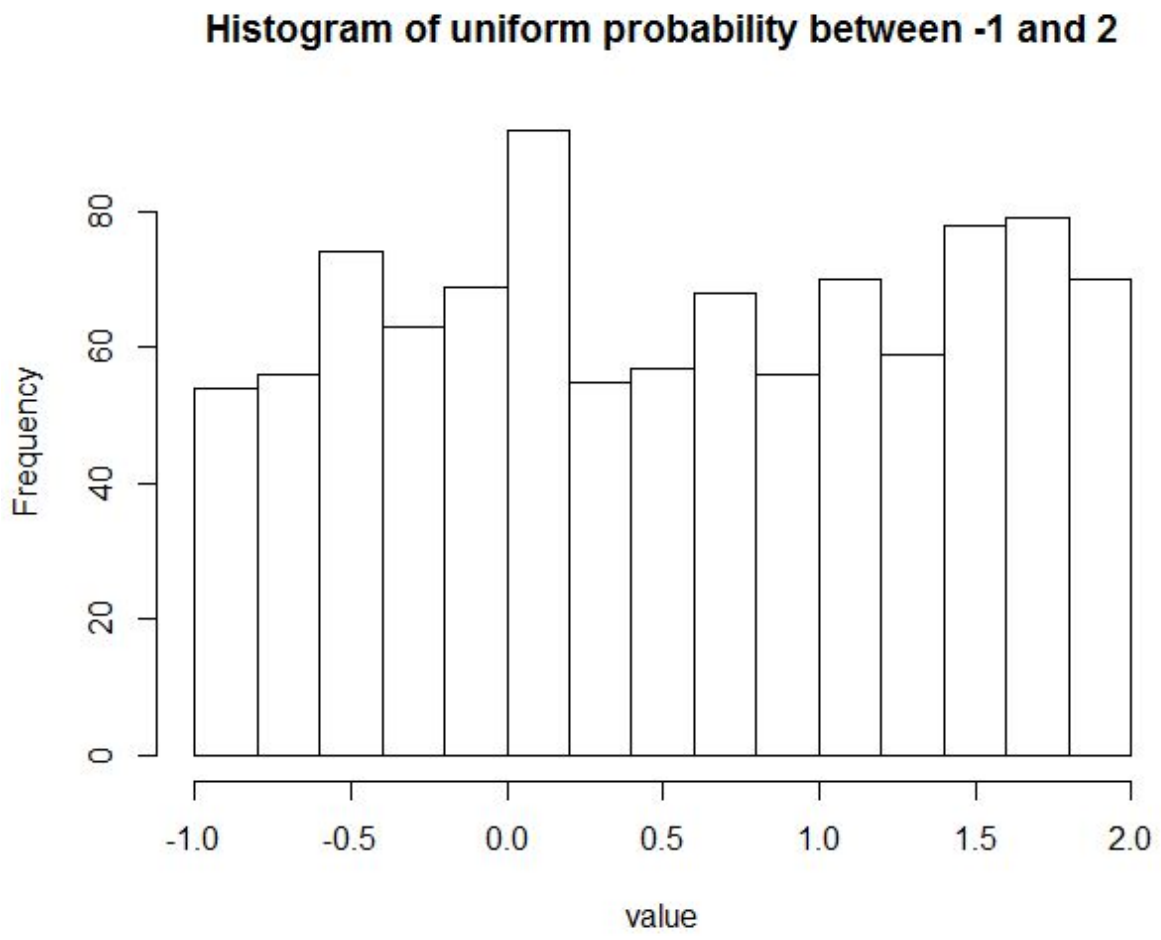
Problem 2. Let us finish the plot of the correlation between waiting times and durationsof Old Faithful data. Recreate the scatter plot of waiting vs. duration times. As we mentioned in class, the best linear assessment in the sense of the least squares fit of a relationship (proportionality) between two or many variables can be achieved with function lm() in R, where 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 thevertical axis (waiting, here) and the second if the slope of the line. Convince yourself thatline with those parameters will truly lie on your graph. Function abline() adds a line tothe previously created graph. Next, pass the variable model to the function abline(). Make that line somewhat thicker and red. Use help(functionName) to find detailsabout invocations of both lm() and abline() functions.

Problem 3. 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 anew 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 and thenin a separate plot for duration times. Please note that boxplot() function also accepts asits first argument a formula such as waiting ~ type, where waiting is the numericvector of data values to be split in groups according to the grouping variable type. Thesecond argument of function boxplot() is called data, which in our case will take thename of our dataset, i.e. faithful. Find a way to add meaningful legends to yourgraphs.

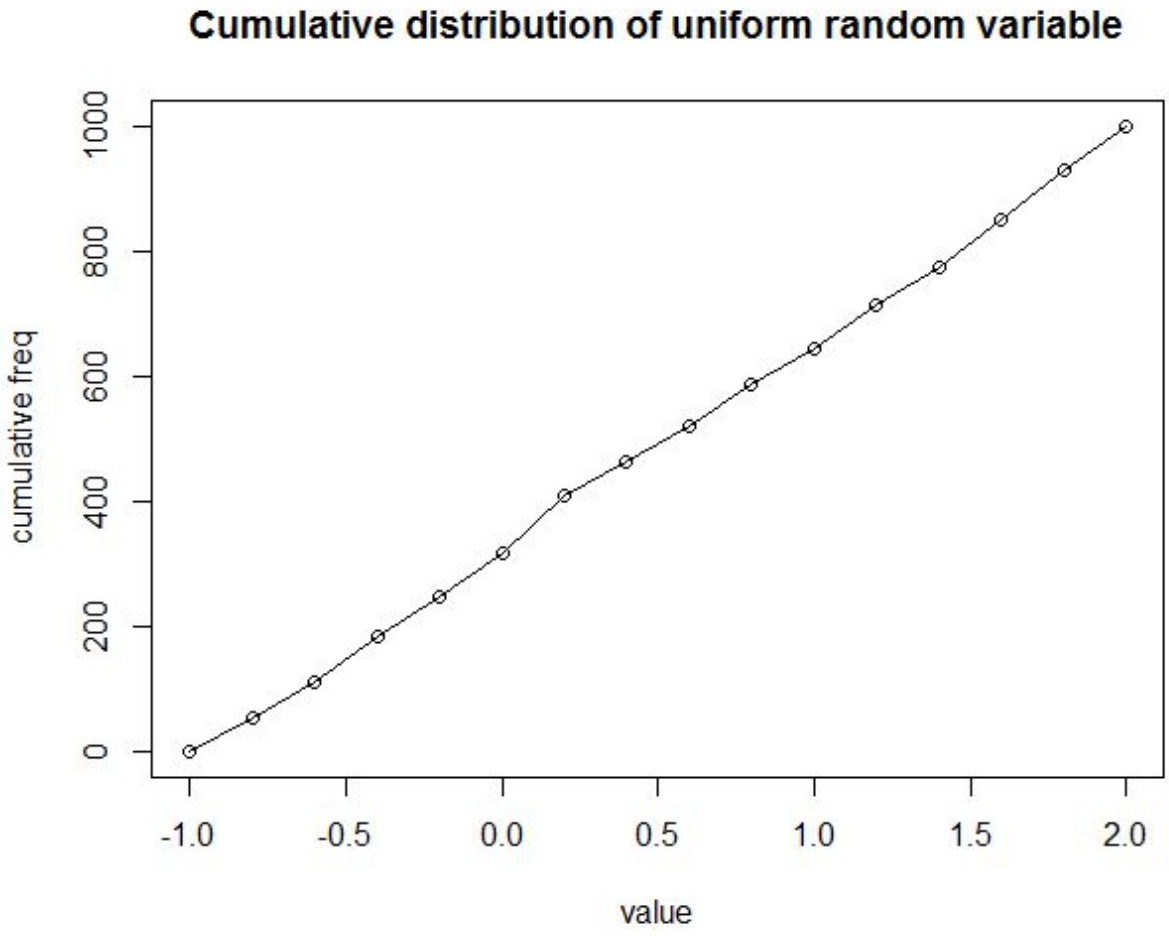
Problem 4. Find a way to generate a random variable with a uniform probability between-1 and 2. Create a histograms with 20 bars to convince yourself that generated values truly fall under a uniform distribution. Create a histogram presenting the relative cumulative distribution of generated data.

```
> x<-runif(1000, min=-1, max=2)
> hist(x, 20, xlab='value', main='Histogram of uniform probability between -1 and 2')
```



Having bigger sample size (>1000) shows that distribution is indeed uniform. For smaller sizes, this might not be evident clearly.

```
> breaks<-seq(-1, 2, 0.2)
> x.cut<-cut(x, breaks, right=FALSE)
> x.freq<-table(x.cut)
> cumfreq<-c(0, cumsum(x.freq))
> plot(breaks, cumfreq, main="Cumulative distribution of uniform random variable",
xlab="value", ylab="cumulative freq")
> lines(breaks, cumfreq)
```

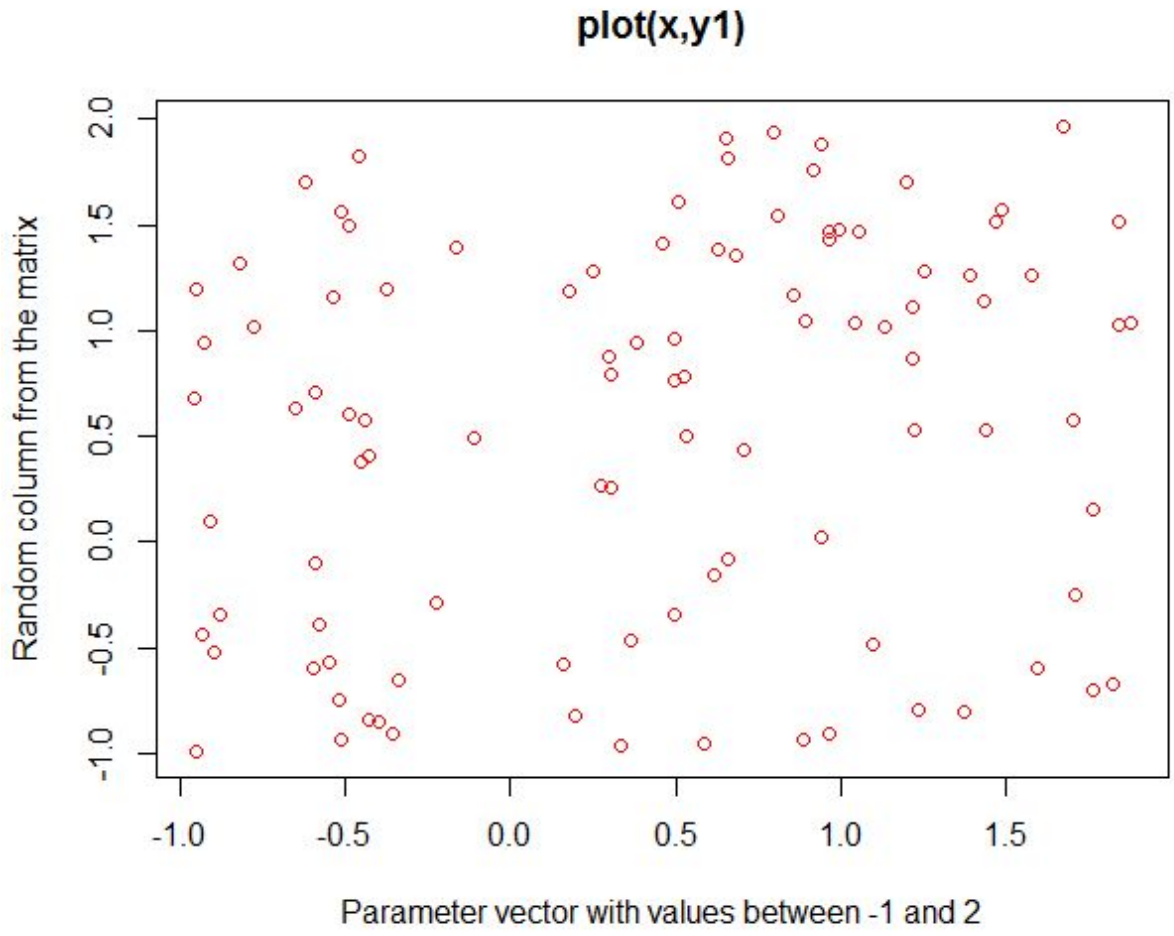


The line plot is almost linear showing that generated values fall under uniform distribution.

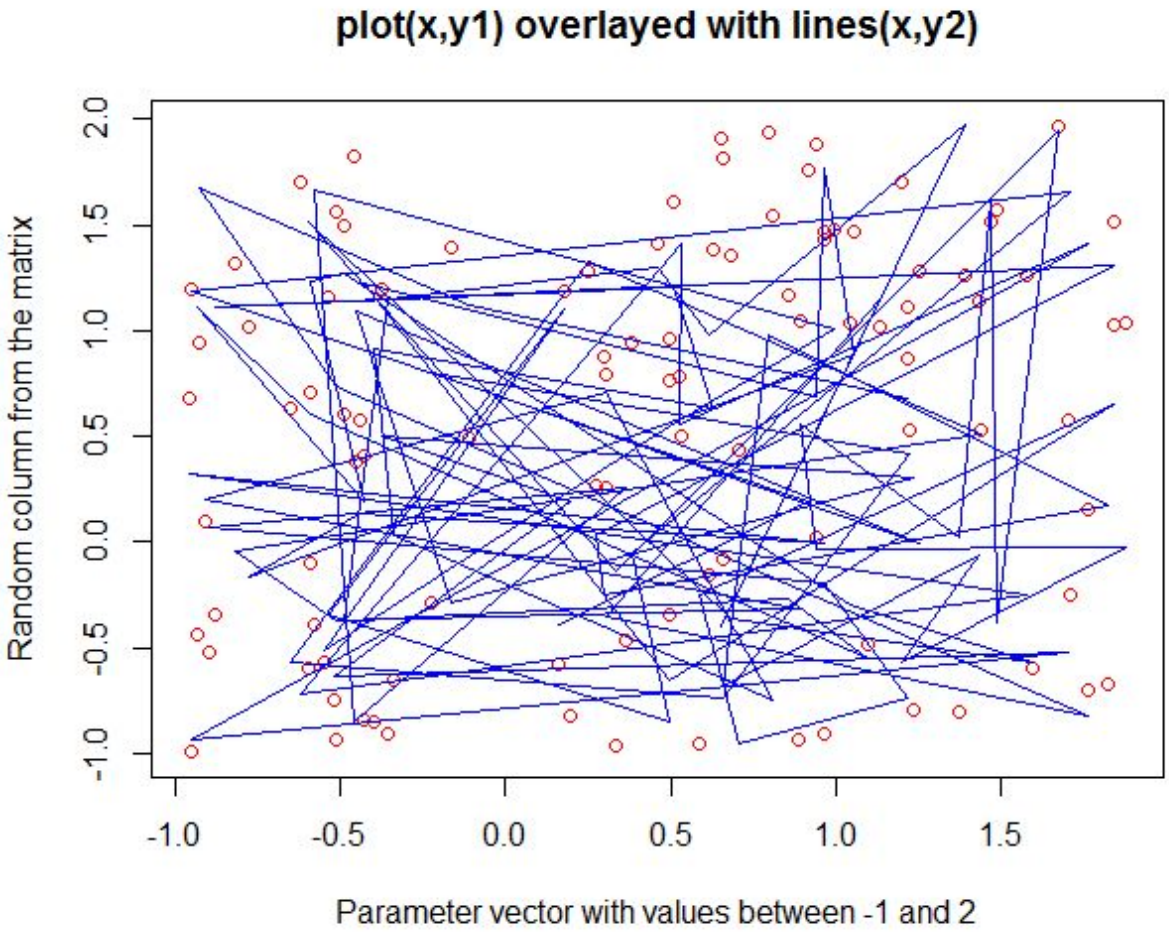
Problem 5. Create a matrix with 40 columns and 100 rows. Populate each column with random variable of the type created in problem 4. Do not create each vector manually.Try to find a way to present two distributions contained in any two of the columns of your matrix on a single plot. To do that you might want to export the distribution data from two columns into two stand-alone vectors of equal length, e.g. y1 and y2. Plot one distribution first using a call to plot(x,y1), where vector x contains the parameter vector with values between -1 and 2 you selected above. To add the next curve(distribution y2) try invoking function lines(x,y2). To improve your diagram,present two curves in different colors and add labels on x and y axis, as well as the title to your graph.

```
> myframe<-data.frame(replicate(40, runif(100, min=-1, max=2)))
> y1<-myframe$X10
> y2<-myframe$X20

> x<-runif(100, min=-1, max=2)
> plot(x, y1, xlab="Parameter vector with values between -1 and 2", ylab="Random
column from the matrix", main="plot(x,y1)", col="red")
```



```
> plot(x, y1, xlab="Parameter vector with values between -1 and 2", ylab="Random
column from the matrix", main="plot(x,y1) overlayed with lines(x,y2)", col="red")
> lines(x, y2, col="blue")
```



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 starts to resemble 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 with values between-1 and 2.

```
> myframe<-data.frame(replicate(40, runif(100, min=-1, max=2)))
> # sum row wise and produce a new column (#41)
> myframe$X41<-rowSums(myframe)

> # overlay with a gaussian (normal) distribution.
> sum<-myframe$X41
```

```
> xfit<-seq(min(sum), max(sum), length=20)
> yfit<-dnorm(xfit, mean=mean(sum), sd=sd(sum))
> yfit<-yfit*diff(h$mids[1:2])*length(sum)
> lines(xfit, yfit, col="blue", lwd=2)
> h<-hist(sum, breaks=10, col="red", main="Histogram with normal curve", xlab="Row wise sum of matrix")
> lines(xfit, yfit, col="blue", lwd=2)
```