MATH50010 Coursework: Old Faithful

Lucinda Khalil

10/12/2021

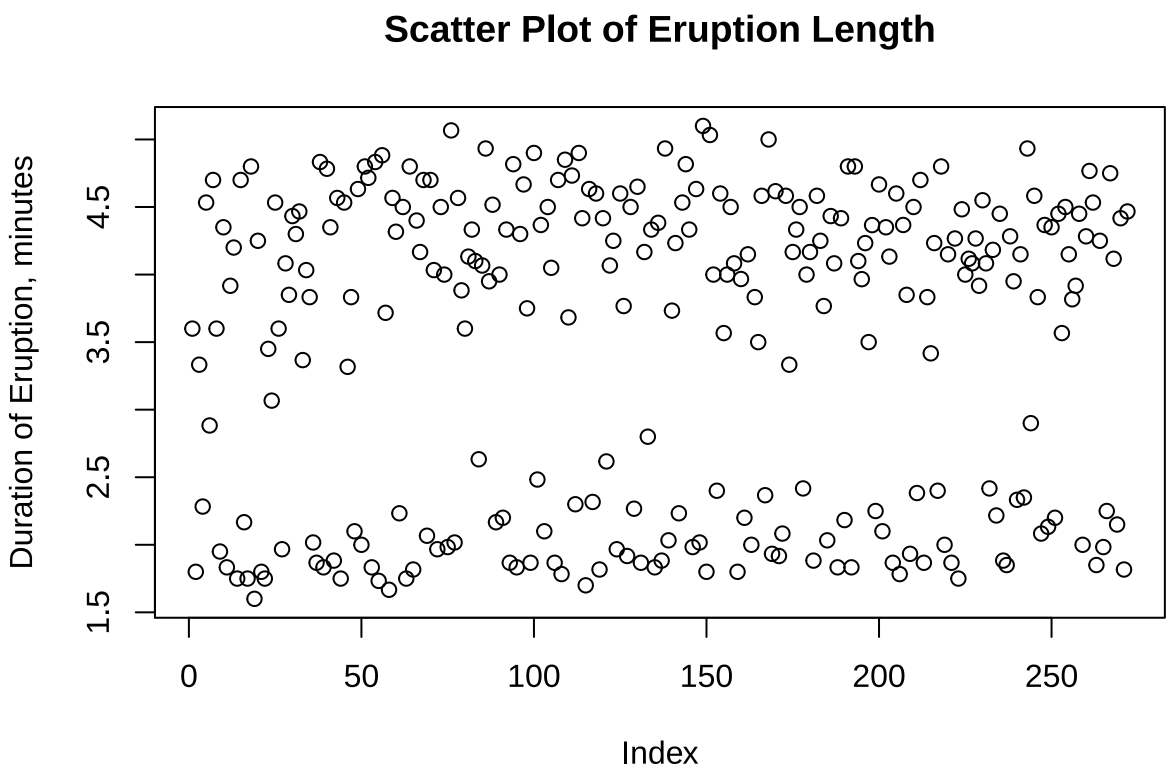
Loading and exploration

1) The following code chuck reads the data into R and selects the time data into a column:

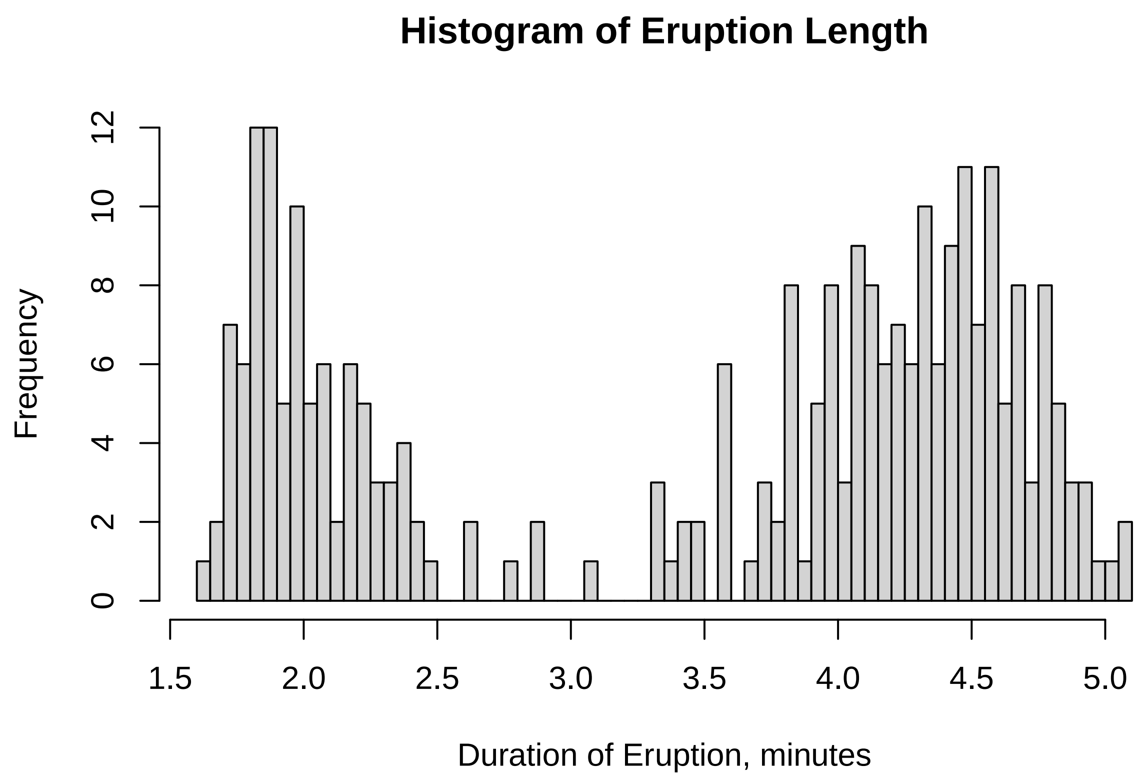
```
old_faithful <- read.csv('old_faithful.txt',header = TRUE, sep = "")
time <- old_faithful$time
```

2) The following code creates a scatter plot and histogram of the distribution of the eruption length:

```
plot(time, main = 'Scatter Plot of Eruption Length',
     ylab = 'Duration of Eruption, minutes')
```



```
hist(time, breaks=50, main = "Histogram of Eruption Length",
     xlab = 'Duration of Eruption, minutes')
```

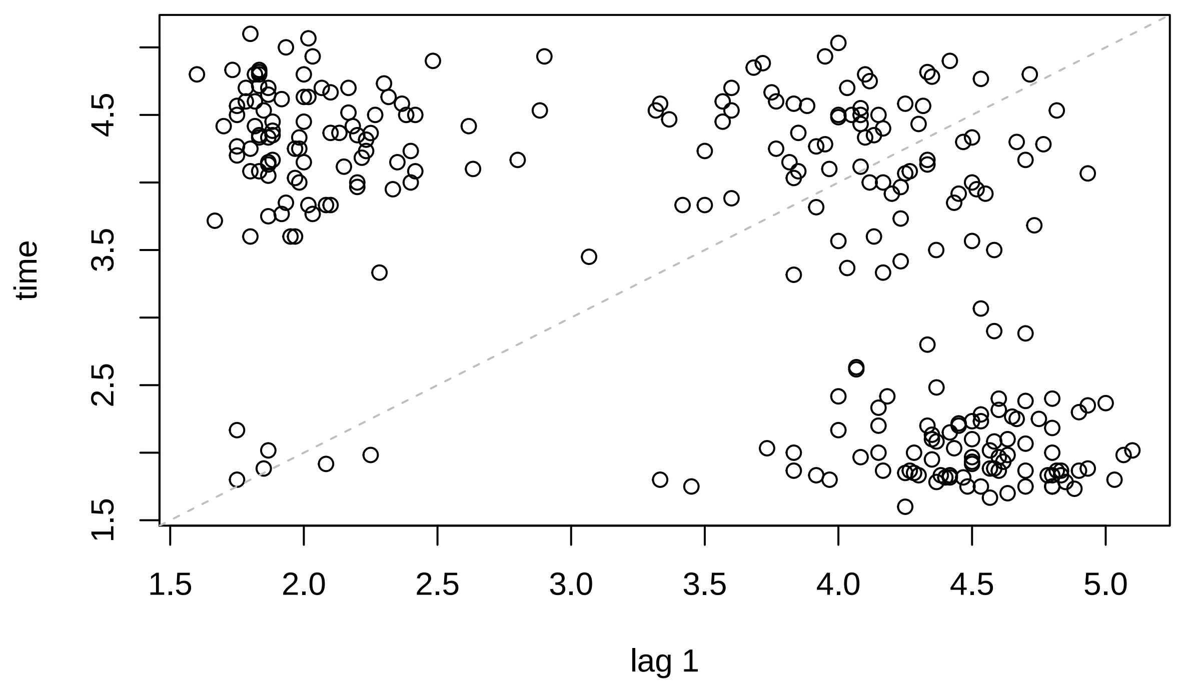


We can see from the scatter plot and histogram that the majority of the eruption times are either low (at around 2 minutes) or high (at around $4 - 5$ minutes) but with very few eruption lengths being between these.

3) The following is a plot of successive times (a lag plot):

```
lag.plot(time, main = 'Lag Plot of Successive Times', asp = NA)
```

Lag Plot of Successive Times



The plot is roughly divided into 4 sections. We see that the duration of an eruption may be related to the duration of the previous eruption. Given that t_i is short, t_{i+1} is likely to be long (≥ 3 minutes). On the other hand, given that t_i is long, we cannot say much about the length of t_{i+1}.

4) By considering the earlier histogram, we can set the *short* threshold to be ≤ 3 minutes while *long* eruptions are > 3 minutes. The following code creates a vector to indicate whether each eruption is *short* (0) or *long* (1). The head of time and state are printed to check for errors.

```
state <- c(1:272)*0

for(i in 1:272){
  if(time[i]>3) state[i]=1
}

head(time)
```

```
[1] 3.600 1.800 3.333 2.283 4.533 2.883
```

```
head(state)
```

```
[1] 1 0 1 0 1 0
```

5) The following calculates the proportion of eruptions in each of the states

```
#creating a string of the states
state_string = paste(state,collapse="")
```

```r
#counting the instances of 0 and 1
library(stringr)
instances = str_count(state_string, c("0","1"))

#calculating proportions
proportions = instances/272

print(paste0("The proportion of long eruptions is: ", proportions[2]))
```

```
## [1] "The proportion of long eruptions is: 0.643382352941177"
```

```r
print(paste0("The proportion of short eruptions is: ", proportions[1]))
```

```
## [1] "The proportion of short eruptions is: 0.356617647058824"
```

6) The following code counts the number of each possible transition of state:

```r
transition_count = str_count(state_string,paste0("(?=",c("00","01","10","11"),")"))

print(paste0("The number of transitions from short to short is: ", transition_count[1]))
```

```
## [1] "The number of transitions from short to short is: 6"
```

```r
print(paste0("The number of transitions from short to long is: ", transition_count[2]))
```

```
## [1] "The number of transitions from short to long is: 91"
```

```r
print(paste0("The number of transitions from long to short is: ", transition_count[3]))
```

```
## [1] "The number of transitions from long to short is: 91"
```

```r
print(paste0("The number of transitions from long to long is: ", transition_count[4]))
```

```
## [1] "The number of transitions from long to long is: 83"
```

Evaluating an Independence Model

7) Assuming that successive states are independent, we can estimate the proportions of the pairs of possible successive states.

Let p_s be the proportion of *short* states and p_l be the proportion of *long*, such that $p_s = 1 - p_l$.

Starting with the transition from *short* to *short*, $(0, 0)$, we want to evaluate the probability $P(X_i = 0, X_{i-1} = 0)$. Since we have assumed independence:

$$P(X_i = 0, X_{i-1} = 0) = P(X_i = 0)^2 = {p_s}^2$$

Similarly for the other states:

$$P(X_i = 1, X_{i-1} = 0) = P(X_i = 1)P(X_{i-1} = 0) = p_s p_l$$

$$P(X_i = 0, X_{i-1} = 1) = P(X_i = 0)P(X_{i-1} = 1) = p_l p_s$$
$$P(X_i = 1, X_{i-1} = 1) = P(X_i = 1)^2 = {p_l}^2$$

So the estimated proportions of the possible successive states are:

$$(0,0) \approx 0.1271761462...$$
$$(0,1) \approx 0.2294415009...$$
$$(1,0) \approx 0.2294415009...$$
$$(1,1) \approx 0.4139408521...$$

8) Still working under the assumption of independence, we can work out the log likelihood ratio statistic for the data. Firstly, we have the probabilities corresponding to the null model as calculated in the previous question:

$$\hat{q}_{00} \approx 0.1271761462...$$
$$\hat{q}_{01} \approx 0.2294415009...$$
$$\hat{q}_{10} \approx 0.2294415009...$$
$$\hat{q}_{11} \approx 0.4139408521...$$

Then, using the more general multinomial model:

$$\widehat{p}_{00} = \frac{n_{00}}{n-1} = \frac{6}{271}$$
$$\hat{p}_{01} = \frac{n_{01}}{n-1} = \frac{91}{271}$$
$$\hat{p}_{10} = \frac{n_{10}}{n-1} = \frac{91}{271}$$
$$\hat{p}_{11} = \frac{n_{11}}{n-1} = \frac{83}{271}$$

We can now use these probabilities to calculate the log likelihood ratio statistic:

```r
#observed frequencies
nij <- c(6,91,91,83)

#null probabilities:
qij <- c(0.1271761462,0.2294415009,0.2294415009,0.4139408521)

#multinomial probabilities:
pij <- c(6/271,91/271,91/271,83/271)

#initializing likelihoods
Lq = 1
Lp = 1

#calculating form of likelihoods
for (i in 1:4) {
  Lq = Lq*qij[i]^nij[i]
}

for (i in 1:4) {
  Lp = Lp*pij[i]^nij[i]
}

#calculating the log likelihood ratio statistic

llrs = log(Lp) - log(Lq)

print(paste0("The log likelihood ratio statistic for the null and multinomal models is: ",
             llrs))
```

```
## [1] "The log likelihood ratio statistic for the null and multinomal models is: 33.8218281347421"
```

9) The following generates 1000 random permutations of the data:

```
#initializing permutations vector
permutations <- c(1:1000)

#creating vector of uniform probabilities
prob = c(1:272)*0+1/272

#creating string of each sample
for (i in 1:1000) {
  permutations[i] = paste(sample(state,size=272,replace=FALSE,prob),collapse="")
}
```

Now, for each permutation we count the successive pairs and calculate the likelihood ratio:

```
#initializing matrix
instances_q9 = matrix(nrow = 1000, ncol = 4)

#initializing vector for stings of states
permutations_str = c(1:1000)

#counting the number of each pair of states
for (n in 1:1000) {
  for (i in 1:4) {
    instances_q9[n,i] = (str_count(permutations[n],
                                   paste0("(?=",c("00","01","10","11"),")"))[i])
  }
}

#the null probabilities are the same as in Q8
#calculating the multinomial probabilities for each permutations

pij_q9 = instances_q9/271

#calculating null and multinomial likelihood

Lq_q9 = c(1:1000)*0+1

for (n in 1:1000) {
  for (i in 1:4) {
```

```r
    Lq_q9[n] = Lq_q9[n]*qij[i]^instances_q9[n,i]

  }
}

Lp_q9 = c(1:1000)*0+1

for (n in 1:1000) {
  for (i in 1:4) {

    Lp_q9[n] = Lp_q9[n]*pij_q9[n,i]^instances_q9[n,i]

  }
}

#calculating the log likelihood ratio statistics

llrs_q9 = log(Lp_q9) - log(Lq_q9)
```

10)

```r
print(paste0("The mean of the 1000 log likelihood ratio statistics obtained above is: ",
             mean(llrs_q9)))
```

```
## [1] "The mean of the 1000 log likelihood ratio statistics obtained above is: 0.557843704090403"
```

We see that the test statistic obtained from the given data of around 33.8 is much greater than 1 and much greater than the log likelihood ratio obtained from the random permutations. This means that we have evidence against the null hypothesis, and we can say that the volcano eruption times may not follow the null distribution.

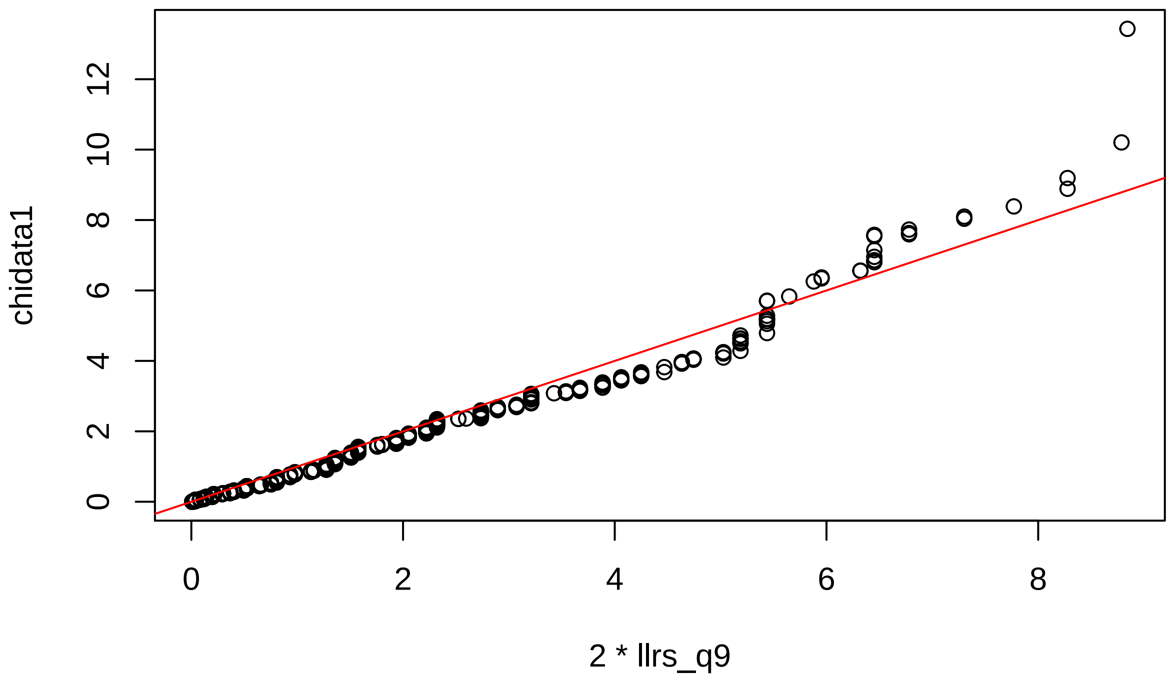
11) The multinomial distribution has 4 degrees of freedom, since once we have determined n, n_{00}, n_{01} and n_{10}, we have also determined n_{11} and all of the probability estimates $\hat{p}_{ij}$.

The null distribution has 3 degrees of freedom, due to the threshold, the number of 0 (or 1) in the data and the assumption of independence.

Therefore, we can estimate twice the log likelihood ratio test statistic as having a chi-square distribution with $r = 4 - 3 = 1$ degrees of freedom.

The QQ-plot below shows that this is roughly true, specifically for $2LLRS < 5$, which includes most of the test statistics. At the upper tail when $2LLRS > 5$, we see that this is not a very good estimate, but this is to be expected due to variations in the random data.

```r
#generating random chi-squared data with df=1
chidata1 <- rchisq(1000,1)

#creating qqplot
qqplot(2*llrs_q9,chidata1)
abline(0,1,col='red')
```



A two-state Markov model

12) As proven in lectures, we can estimate the parameters using:

$$\hat{p}_{ij} = \frac{n_{ij}}{n_{i1} + n_{i2}}$$

So an estimate of the the transition matrix of the Markov chain is given by:

$$\begin{pmatrix} \frac{6}{97} & \frac{91}{97} \\ \frac{91}{174} & \frac{83}{174} \end{pmatrix}$$

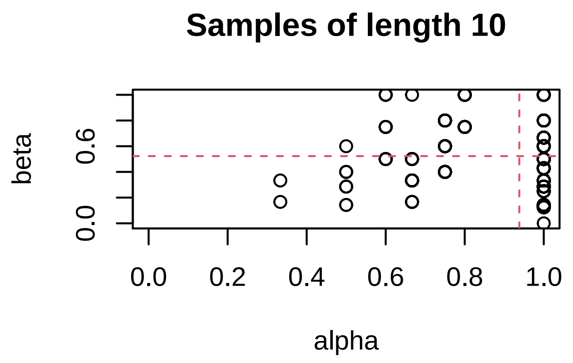
13) Writing the function:

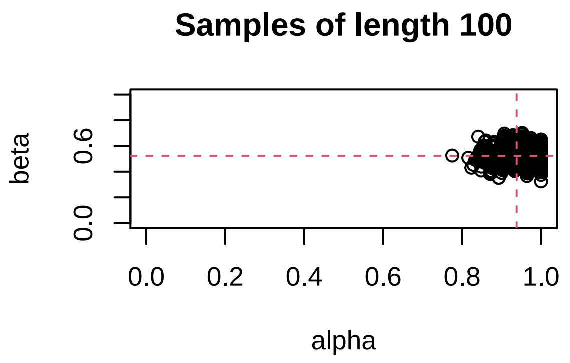
```r
#function to make a markov chain
markov_chain <- function(P,initial_dist,m) {

  #initialising chain
  chain <- c(1:m)

  #filling in first entry according to initial distribution
  chain[1]=sample(c(1:nrow(P)),size=1,prob=initial_dist)

  #finding chain corresponding to transition matrix
  for(i in 2:m) {
    chain[i] <- sample(c(1:nrow(P)),size=1,prob=P[(chain[i-1]),],replace=TRUE)
  }

  #return markov chain
  chain
}

#function to estimate parameters from a markov chain
ml_est <- function(chain, P) {

  #finding number of possible states
  no_of_states = nrow(P)

  no_of_pairs = 2^no_of_states

  #initializing pairs vector
  pairs = c(1:no_of_pairs)

  #all possible combinations
  matrix_of_pairs = expand.grid(1:no_of_states, 1:no_of_states)

  #forming strings of each pair
  for (i in 1:no_of_pairs) {
    pairs[i] = paste(matrix_of_pairs[i,],collapse = "")
  }

  #creating string of transition states
  state_string=paste(chain,collapse = "")

  #counting number of each transition pair
  pair_counts=str_count(state_string,paste0("(?=",pairs,")"))

  #initializing variables
  parameter_estimates = c(1:no_of_pairs)
  total_from_j = 0

  #iterating through each possible pair and finding the corresponding probability estimate
  for (j in 1:no_of_states) {
    for (i in 0:(no_of_states-1)) {
      total_from_j = 0
      for (k in 0:(no_of_states-1)) {
        total_from_j = total_from_j+pair_counts[j+k*no_of_states]
      }
      parameter_estimates[j+i*no_of_states] = pair_counts[j+i*no_of_states]/total_from_j #from j to (i+
    }
  }

  #return estimates
  parameter_estimates
}


#overall function to generate parameter estimates for multiple samples
```

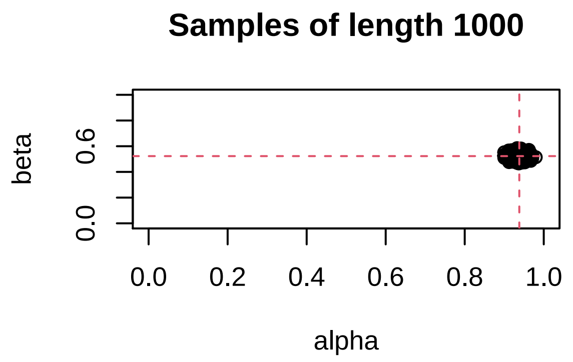
```r
par_est <- function(P,initial_dist,m) {

  #initializing reuqired variables
  n_sample = 1000
  #mchains = c(1:n_sample)

  #iterating through each sample and finding estimates
  par_est = matrix(nrow=n_sample,ncol=(2^ncol(P)))
  for (i in 1:n_sample) {
    x = markov_chain(P,initial_dist,m)
    #mchains[i] = paste(markov_chain(P,initial_dist,m),collapse="")
    par_est[i,] = ml_est(x,P)
  }

  #returning parameter estimates
  par_est
}
```

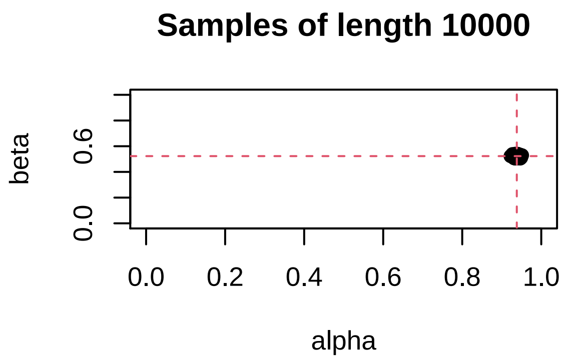
Now, let's consider the specific case of the transition matrix given in Q12. Let *alpha* be the probability estimate of a transition from state 0 to 1, and β be the probability estimate of a transition from state 1 to 0. We also see that as the size of the samples increases, the estimates become more accurate and tend to the expected values.

```r
#generating samples anf estimates for different values of m
estimates1 <- par_est(P,c(0.5,0.5),10)
estimates2 <- par_est(P,c(0.5,0.5),100)
estimates3 <- par_est(P,c(0.5,0.5),1000)
estimates4 <- par_est(P,c(0.5,0.5),10000)

#crreating scatter plots
par(mfrow=c(2,2))

plot(estimates1[,3],estimates1[,2],xlim=c(0,1),ylim=c(0,1),xlab="alpha",ylab="beta",
     main="Samples of length 10")
abline(v=91/97,col=2,lty=2)
abline(h=91/174,col=2,lty=2)

plot(estimates2[,3],estimates2[,2],xlim=c(0,1),ylim=c(0,1),xlab="alpha",ylab="beta",
     main="Samples of length 100")
abline(v=91/97,col=2,lty=2)
abline(h=91/174,col=2,lty=2)

plot(estimates3[,3],estimates3[,2],xlim=c(0,1),ylim=c(0,1),xlab="alpha",ylab="beta",
     main="Samples of length 1000")
abline(v=91/97,col=2,lty=2)
abline(h=91/174,col=2,lty=2)

plot(estimates4[,3],estimates4[,2],xlim=c(0,1),ylim=c(0,1),xlab="alpha",ylab="beta",
     main="Samples of length 10000")
abline(v=91/97,col=2,lty=2)
abline(h=91/174,col=2,lty=2)
```



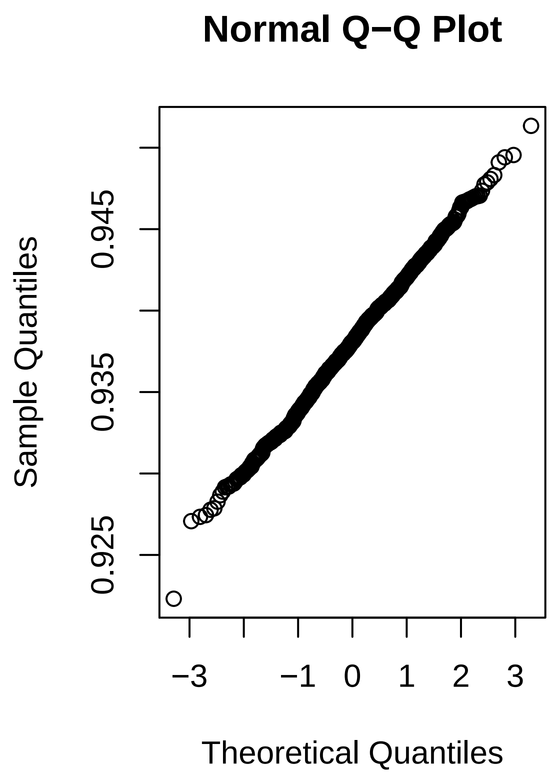


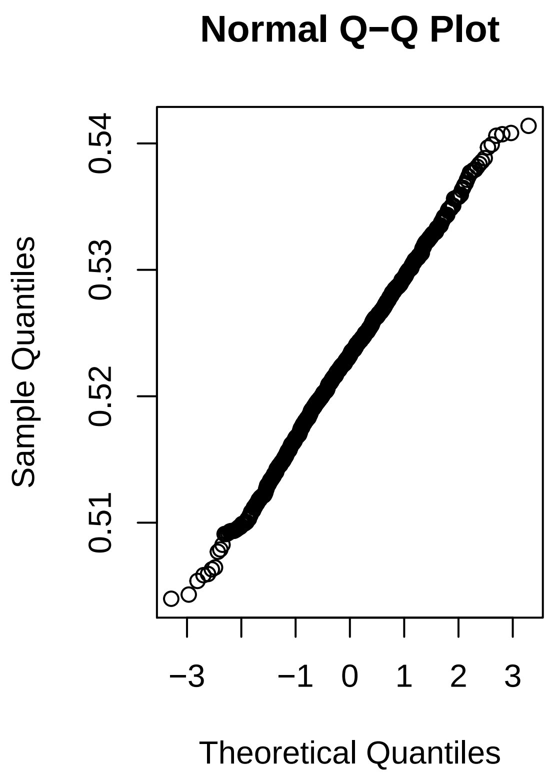




We also see that the estimates of the sample form an elliptical shape, this is because that random nature of the sampling produces a bivariate normal distribution for the estimates. This shown by the QQ-plots below. They form a straight line but do not match up with $y = x$ in this case since the values have not been standardised.

```r
par(mfrow=c(1,2))
qqnorm(estimates4[,3])
qqnorm(estimates4[,2])
```

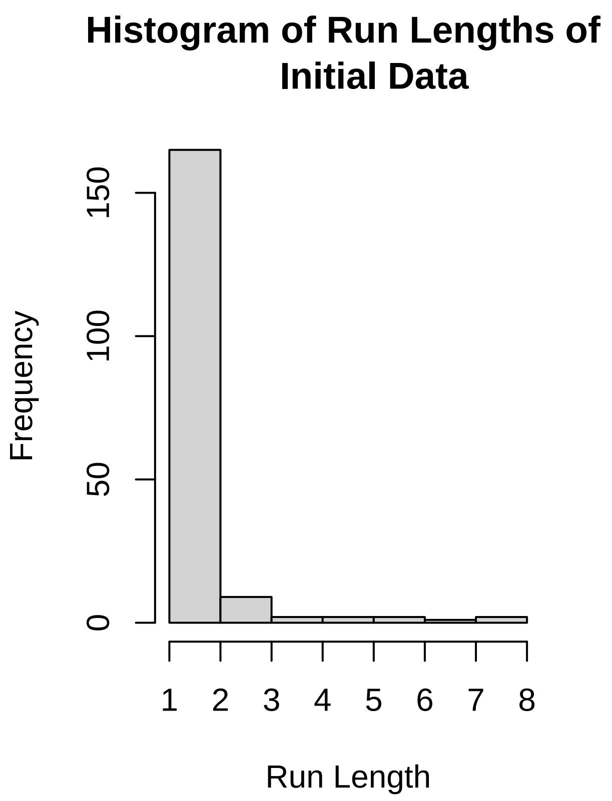


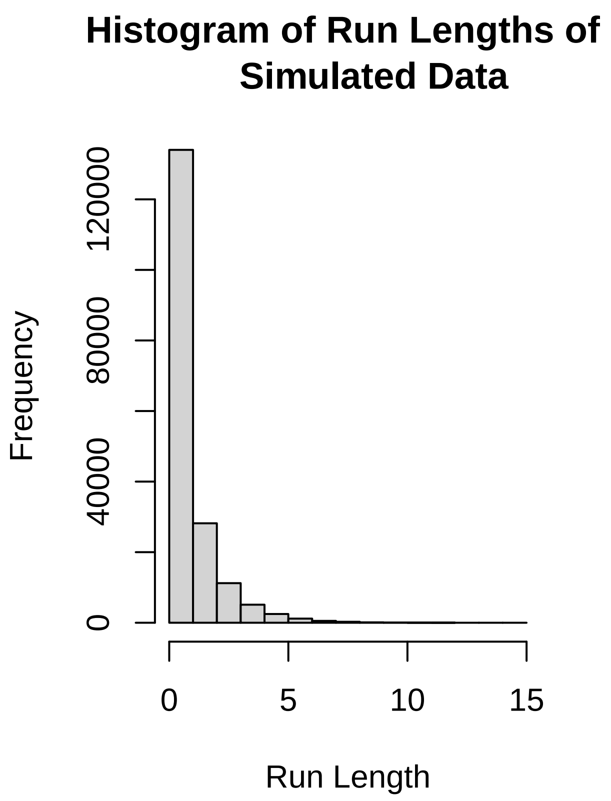


14) Now, we consider the run lengths of the simulations:

```
simulations <- matrix(nrow=1000,ncol=272)

#generating simulated data
for (i in 1:1000) {
  simulations[i,] = markov_chain(P,c(0.5,0.5),272)
}

#initializing vector
run_lengths <- c(0)

#finding run lengths using rle function and appending them to vector
for (i in 1:1000) {
  run_lengths = append(run_lengths,rle(simulations[i,])$lengths)
}
```

```
run_lengths_initial_data = rle(state)$lengths

par(mfrow=c(1,2))

hist(run_lengths_initial_data, main="Histogram of Run Lengths of
     Initial Data",
     xlab="Run Length")
hist(run_lengths, main="Histogram of Run Lengths of
     Simulated Data", xlab="Run Length")
```





The histograms show that the distributions of the run lengths of the initial data and the simulated data are very similar. These seem to follow and exponential distibution where most of the run lengths are very short. This seems reasonable as the lag plot from earlier has the majority of its data points at the top left or bottom right, meaning that the eruption times commonly change state.

15) Firstly, we explored the given data to find that the eruption times were mostly either very short or long, few had an eruption time in the middle. So we categorized each piece of data as long or short. We then looked at whether a generic null model where the length of each eruption is assumed to be independent of the other would be a good fit for the given data. This model assumed that long eruptions were more likely than short ones since this is what was observed in the given data. We found evidence that this model may *not* be a good fit.

Then, we instead modeled data as a two-state time-homogeneous Markov chain, meaning that the next eruption time is independent of all previous eruption times, given that we know the current eruption time. The parameters were estimated using the log likelihood. We found that this was a good fit for the data. We also looked at whether the run lengths (number of eruptions in a row which are all short or all long) were distributed in the same way, which they were.