ex 3b

Christoffer Mondrup Kramer

2023-05-20

Exervcise 3b- Access normality for stiffness data in table 4.3 (p. 186)

Let us read the data

```
df <- read.table("T4-3.dat", header = FALSE)
df
```

```
##      V1   V2   V3   V4    V5
## 1  1889 1651 1561 1778  0.60
## 2  2403 2048 2087 2197  5.48
## 3  2119 1700 1815 2222  7.62
## 4  1645 1627 1110 1533  5.21
## 5  1976 1916 1614 1883  1.40
## 6  1712 1712 1439 1546  2.22
## 7  1943 1685 1271 1671  4.99
## 8  2104 1820 1717 1874  1.49
## 9  2983 2794 2412 2581 12.26
## 10 1745 1600 1384 1508  0.77
## 11 1710 1591 1518 1667  1.93
## 12 2046 1907 1627 1898  0.46
## 13 1840 1841 1595 1741  2.70
## 14 1867 1685 1493 1678  0.13
## 15 1859 1649 1389 1714  1.08
## 16 1954 2149 1180 1281 16.85
## 17 1325 1170 1002 1176  3.50
## 18 1419 1371 1252 1308  3.99
## 19 1828 1634 1602 1755  1.36
## 20 1725 1594 1313 1646  1.46
## 21 2276 2189 1547 2111  9.90
## 22 1899 1614 1422 1477  5.06
## 23 1633 1513 1290 1516  0.80
## 24 2061 1867 1646 2037  2.54
## 25 1856 1493 1356 1533  4.58
## 26 1727 1412 1238 1469  3.40
## 27 2168 1896 1701 1834  2.38
## 28 1655 1675 1414 1597  3.00
## 29 2326 2301 2065 2234  6.28
## 30 1490 1382 1214 1284  2.58
```

Check normality for each of the four attributes

```
library(MASS)

get_best_lambda <- function(data_vector) {
    # Transform data
    # Make box cox plot on our data
    df_box_cox<- boxcox(data_vector~1,lambda=seq(-2, 2, 1/10)) # Can also be seq(-.5, 1.5,.0
1)
    df_box_cox

    # Get best lambda
    max_lambda <- df_box_cox$x[which.max(df_box_cox$y)]
    print(paste("The best lambda is ", max_lambda))
    return(max_lambda)
}

box_cox_transformation <- function(data_vector){

  lambda <- get_best_lambda(data_vector)
  if (lambda == 0){
    transformed_vector <- log(data_vector)
  }

  else {
  transformed_vector <- ( (data_vector^lambda) - 1)/lambda
  }

  return(transformed_vector)
}


test_norm <- function(data_vector, signigicance = 0.05, transform_data = FALSE) {
  # You can chose the following significance levels
  # 0.01
  # 0.05
  # 0.10

  if (signigicance == 0.01){
    signigicance_col <- 2
  }

  else if (signigicance == 0.05){
    signigicance_col <- 3
  }

  else if (signigicance == 0.1){
    signigicance_col <- 4
  }



  # ----- QQ plot -----
  qq <- qqnorm(data_vector, main = "Original Data QQ plot")
  qqline(data_vector)
```

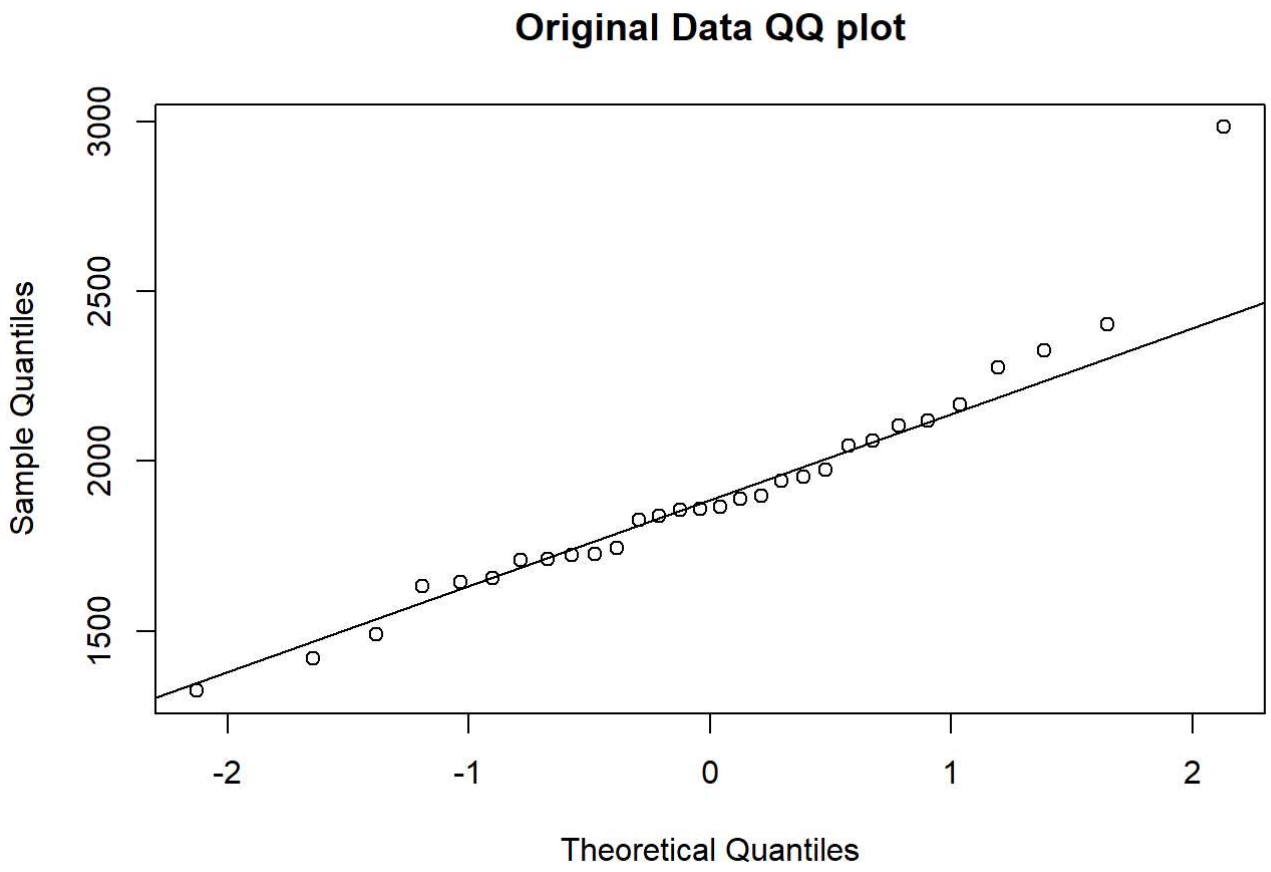
```
  print(paste("Length of data is ", length(data_vector)))
  # ----- Hypothesis test ------
  # Create Testing table
  n <- c(5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 75, 100, 150, 200, 300)
  one <- c(0.8299, 0.8801, 0.9126, 0.9269, 0.9410,
        0.9479, 0.9538, 0.9599, 0.9632, 0.9671,
        0.9695, 0.9720, 0.9771, 0.9822, 0.9879, 0.9905, 0.9935)

  five <- c(0.8788, 0.9198, 0.9389,
        0.9508, 0.9591, 0.9652,
        0.9682, 0.9726, 0.9749,
        0.9768, 0.9787, 0.9801,
        0.9838, 0.9873, 0.9913, 0.9931, 0.9953)

  ten <- c(0.9032, 0.9351, 0.9503,
        0.9604, 0.9665, 0.9715,
        0.9740, 0.9771, 0.9792,
        0.9809, 0.9822, 0.9836,
        0.9866, 0.9895, 0.9928, 0.9942, 0.9960)
  testing_tbl <- data.frame(
  n,
  one,
  five,
  ten
  )

  # Find index of testing (n)
  sample_size <- length(data_vector)
  i <- 1
  prev_value = NaN
  for (n in testing_tbl$n){

    if (sample_size > testing_tbl$n[length(testing_tbl$n)]){
      i <- length(testing_tbl$n)
      break
    }

    if (n == sample_size){
      exact_sample_size <- TRUE
      break
    }

    else if ( n > sample_size & prev_value < sample_size){
      break
    }
    i <- i + 1
    prev_value <- n
  }

  print(testing_tbl[(i - 1) : i, ])
  # ----- Normal -------
  cor_coef <- cor(qq$x, qq$y)
  normality = FALSE
  if (cor_coef > testing_tbl[i, signigicance_col]){
    print(paste("With a correlation coefficient of ", cor_coef, "The data is normal within si
gnificance levels of", signigicance))
```

```
    normality = TRUE
    return(normality)
  }


  # ----- Not normal -----
  else {

    if (transform_data) {
      print(paste("With a correlation coefficient of ", cor_coef,
            "The data is not normal within significance levels of", signigicance, "Transformi
ng data ... "))

      # Transform data
      z_vector <- box_cox_transformation(data_vector)

      # QQ plot on transformed data
      qqz <- qqnorm(z_vector, main = "Transformed Data QQ plot")
      qqline(z_vector)

      # Hypothesis test on transformed data
      z_cor_coef <- cor(qqz$x, qqz$y)
      if (z_cor_coef > testing_tbl[i, signigicance_col]){
        print(paste("With a correlation coefficient of ", z_cor_coef,
                    "The data is normal within significance levels of", signigicance,
                    "For the box cox transformed data"))
        normality = TRUE
        return(normality)
      }
      else {
          print(paste("With a correlation coefficient of ", z_cor_coef,
                "The data is normal within significance levels of", signigicance,
                "For the box cox transformed data"))
      }
    }
    else {
      print(paste("With a correlation coefficient of ", cor_coef,
      "The data is not normal within significance levels of", signigicance))
    }
  }
}
```

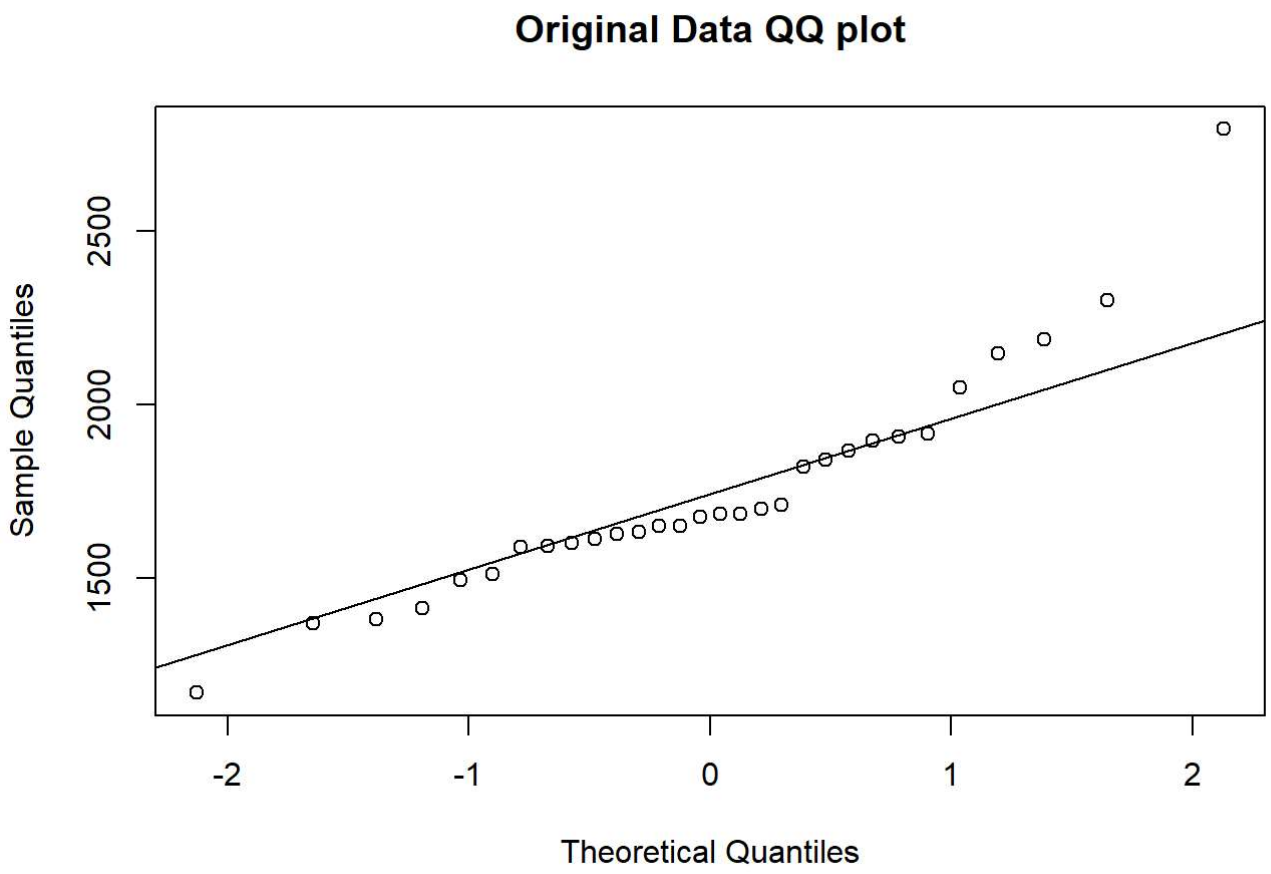
Let us check each attribute

```
for (i in colnames(df)){
  print(i)
  test_norm(df[[i]], signigicance = 0.01)
}
```

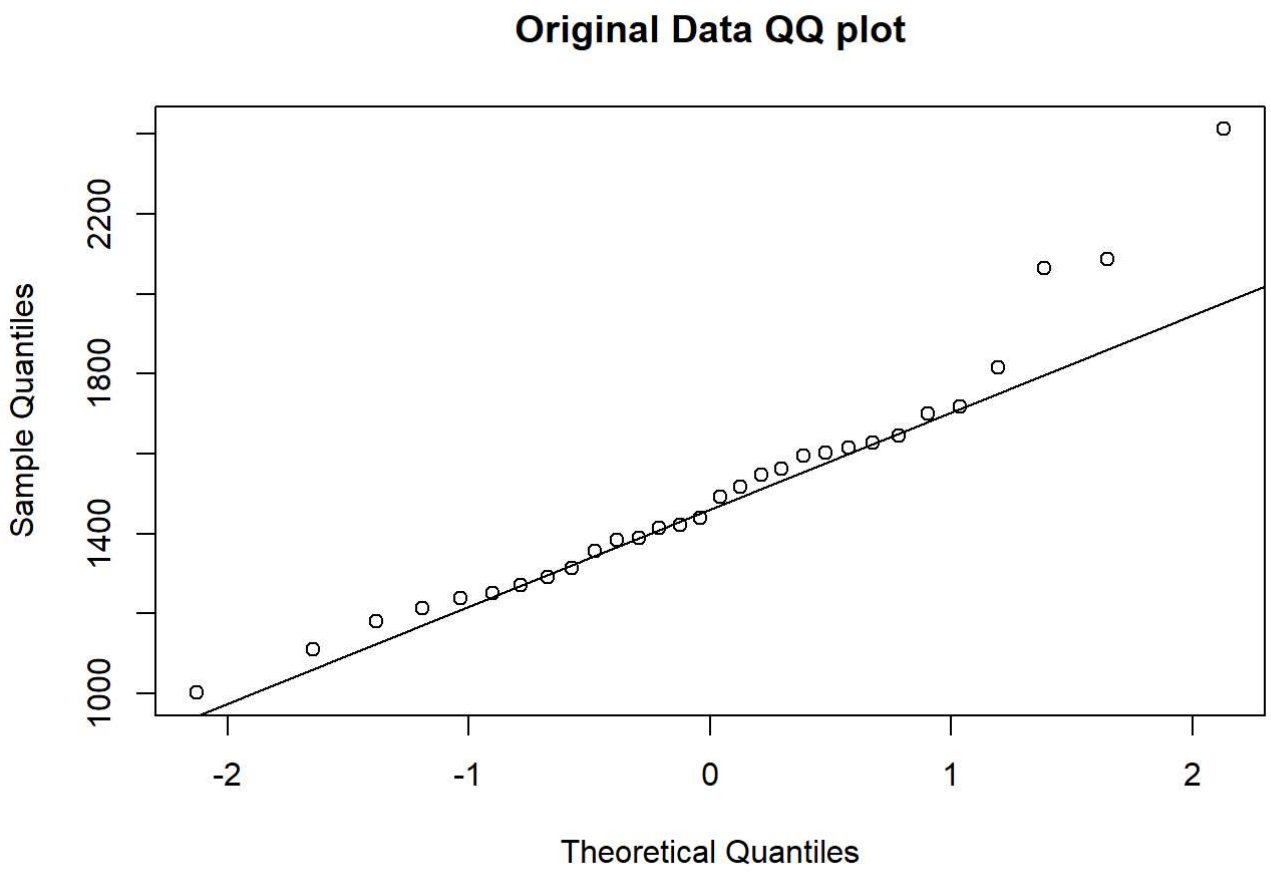
```
## [1] "V1"
```



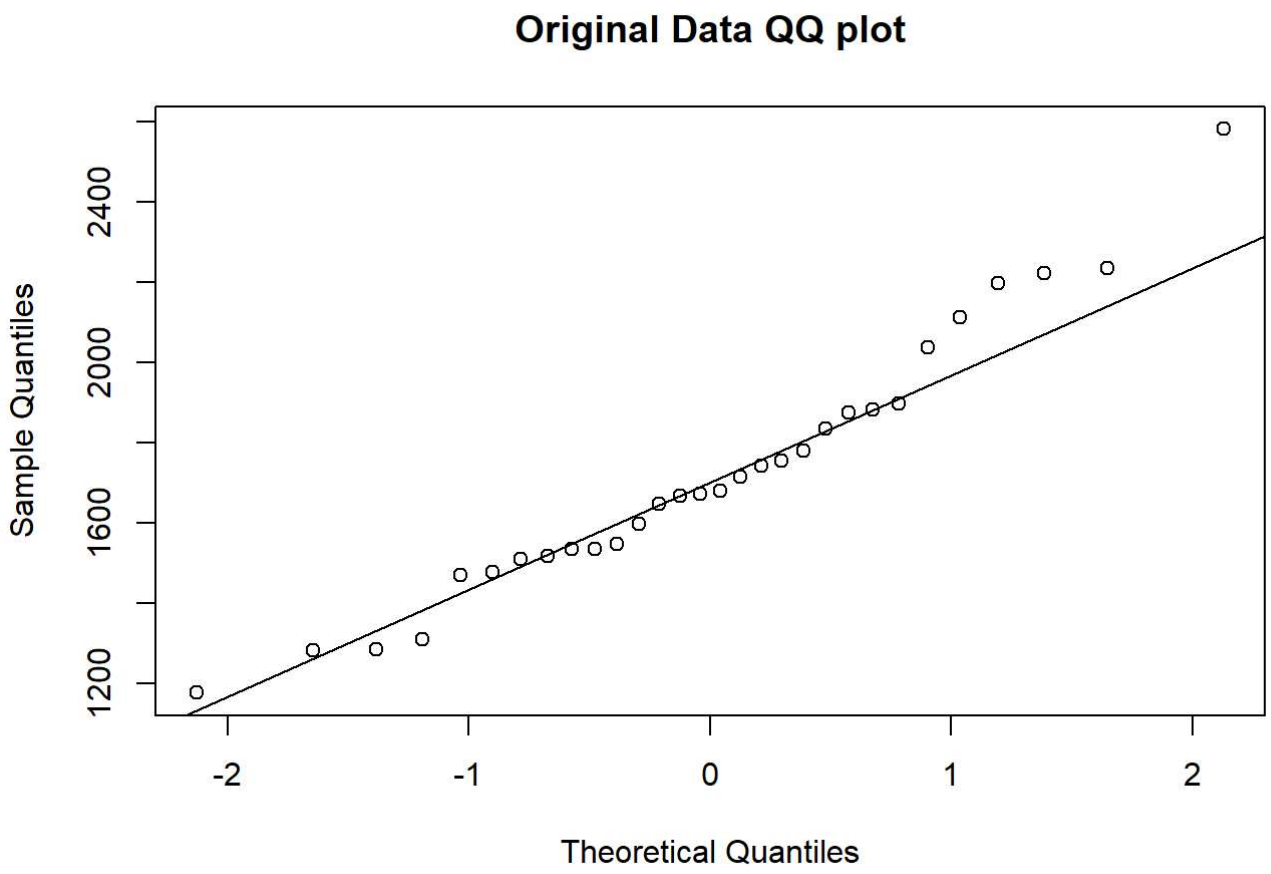

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.959864226926573 The data is normal within signif
icance levels of 0.01"
## [1] "V2"
```



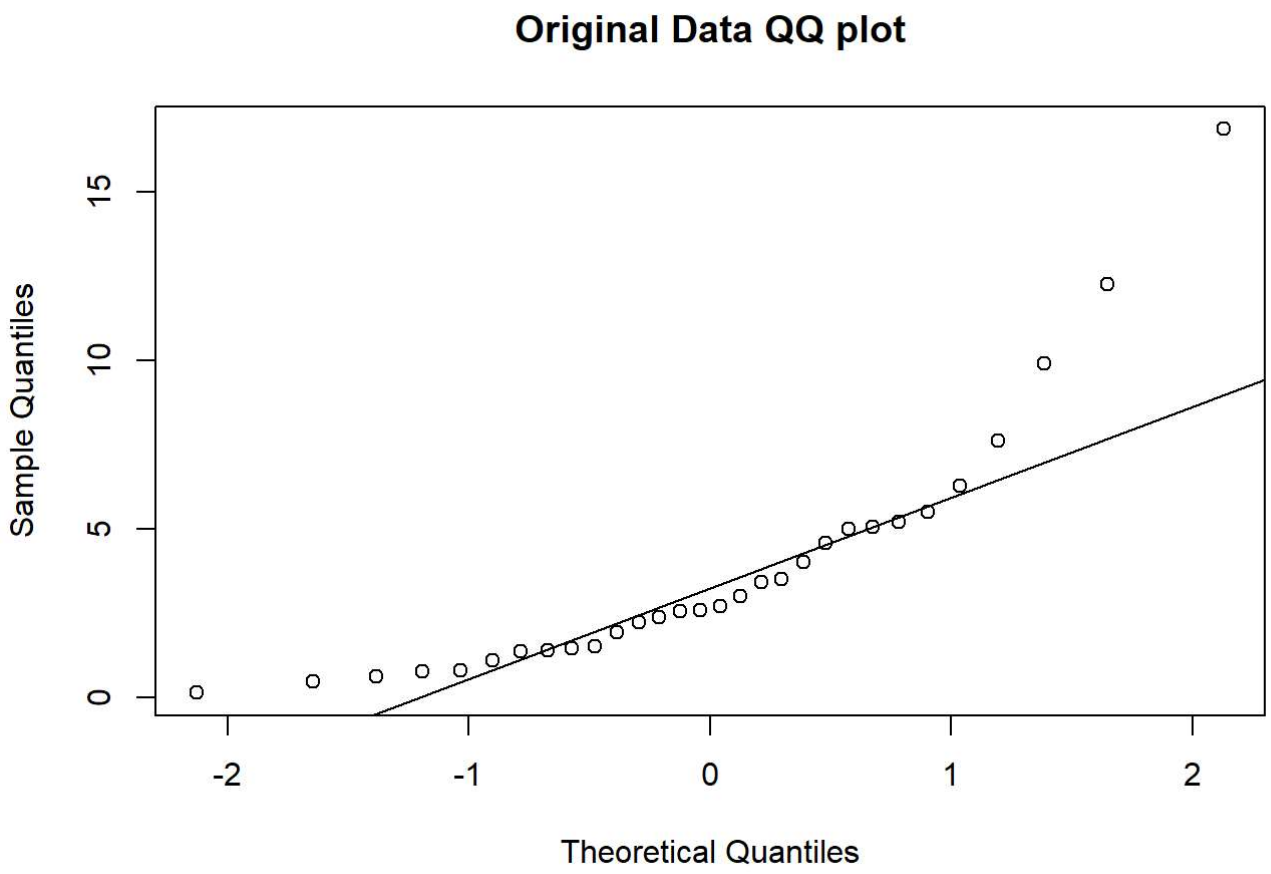

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.950390941215503 The data is normal within signif
icance levels of 0.01"
## [1] "V3"
```




```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.963410375639771 The data is normal within signif
icance levels of 0.01"
## [1] "V4"
```



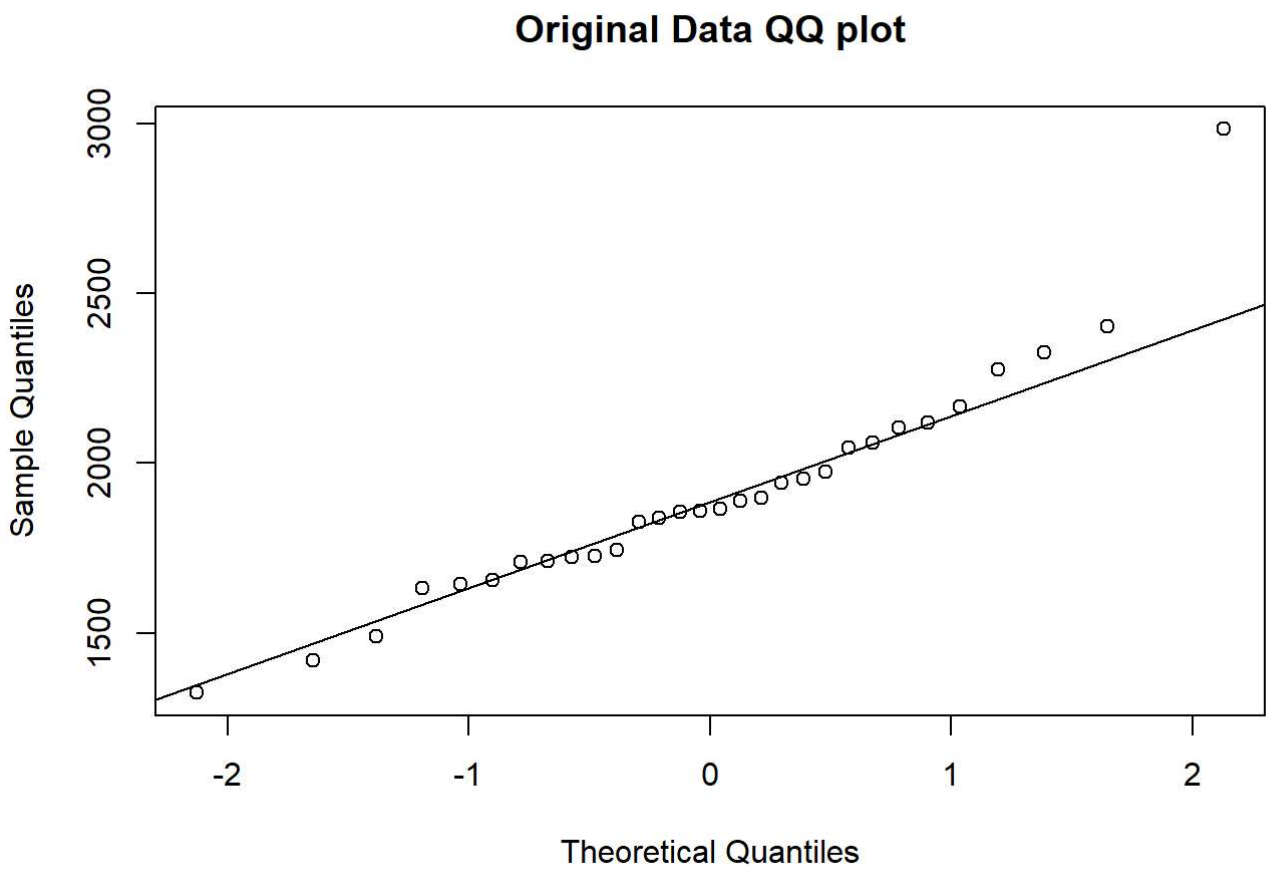

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.980292022100608 The data is normal within signif
icance levels of 0.01"
## [1] "V5"
```



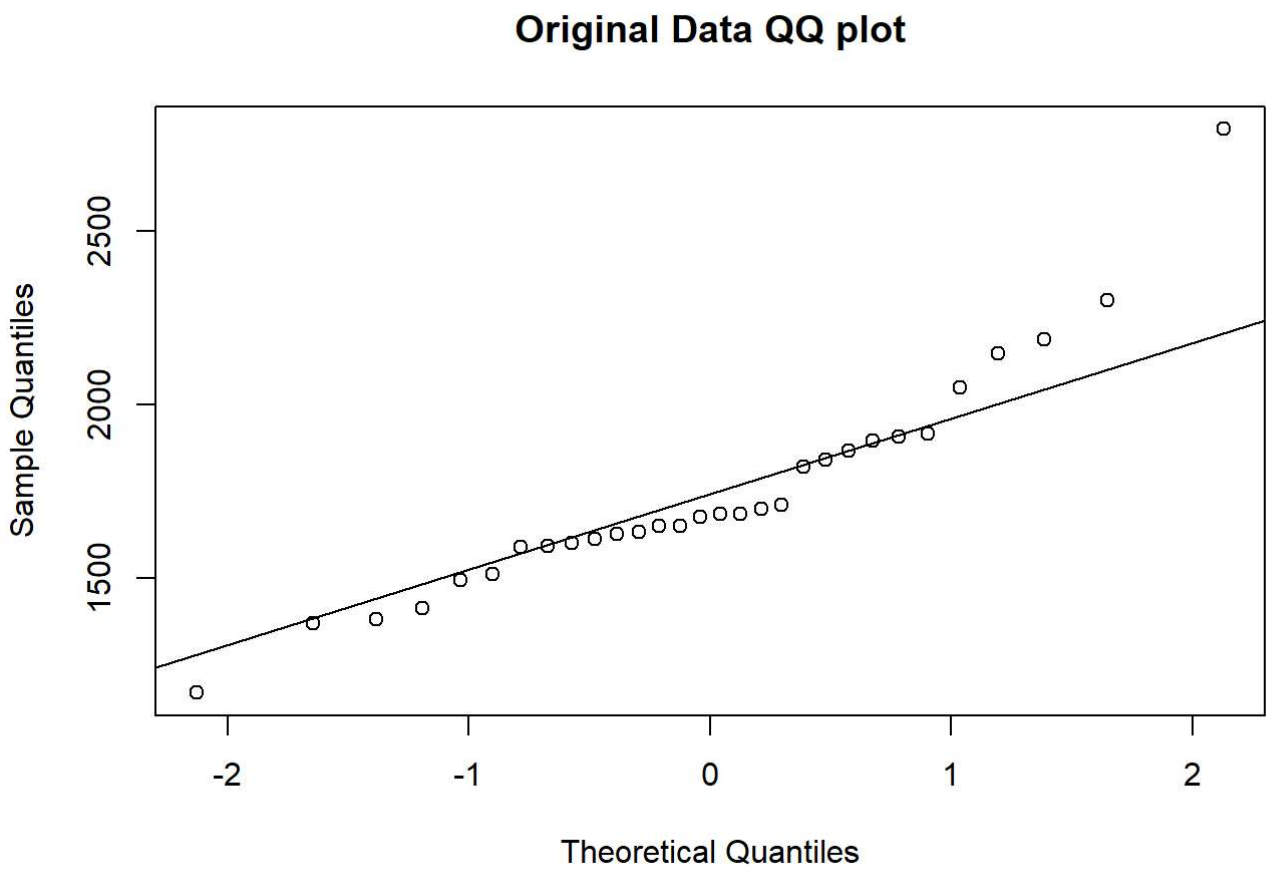

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.891223192804262 The data is not normal within si
gnificance levels of 0.01"
```

```
for (i in colnames(df)){
  print(i)
  test_norm(df[[i]], signigicance = 0.05)
}
```

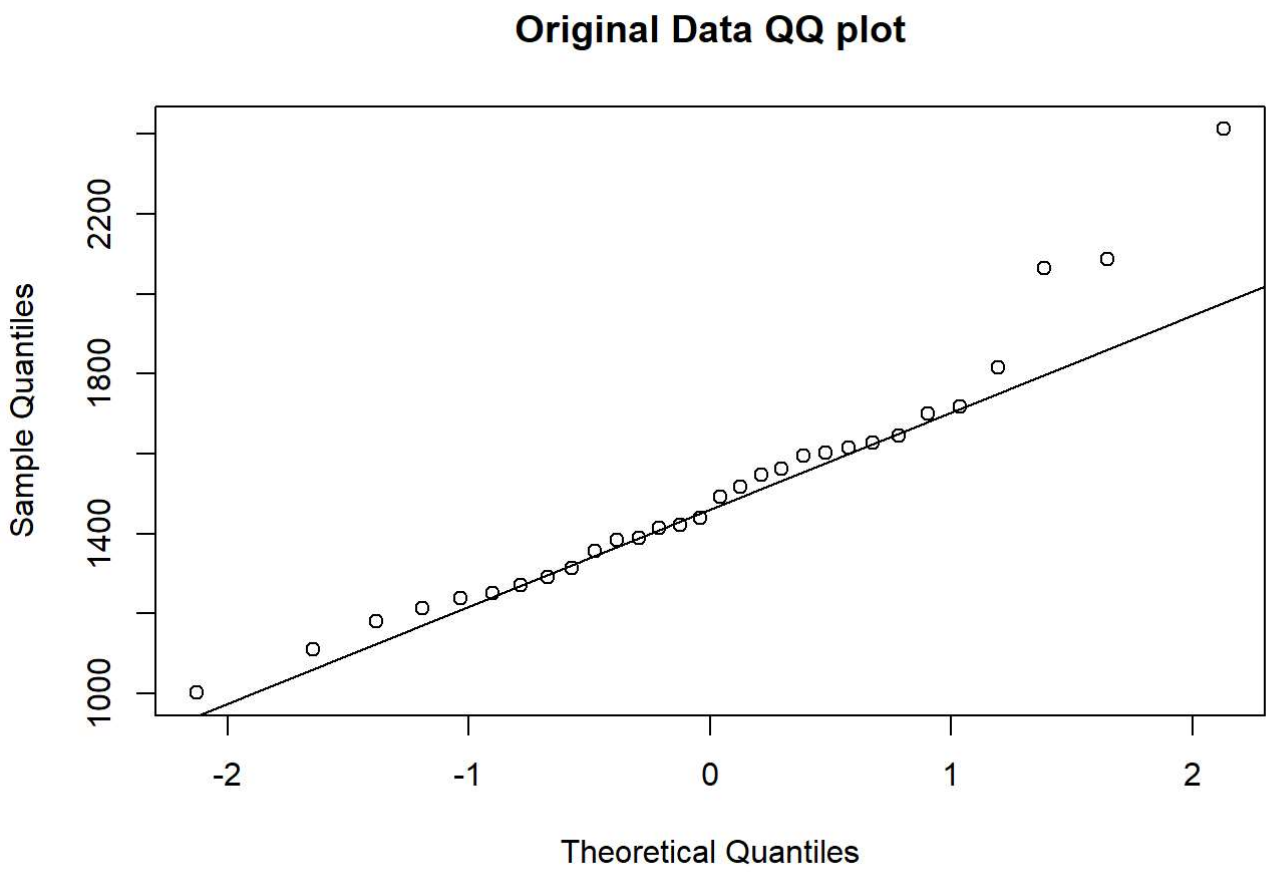
```
## [1] "V1"
```



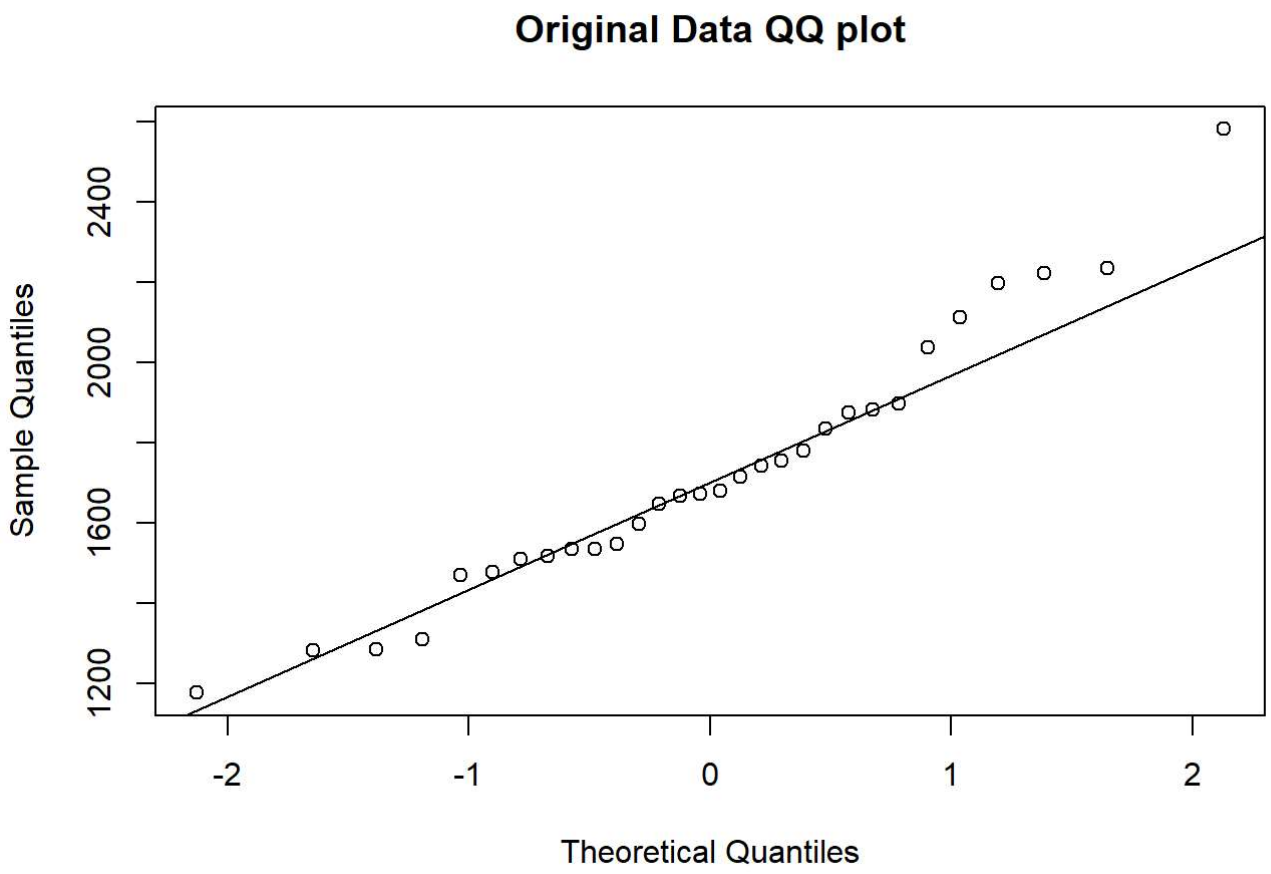

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.959864226926573 The data is not normal within si
gnificance levels of 0.05"
## [1] "V2"
```



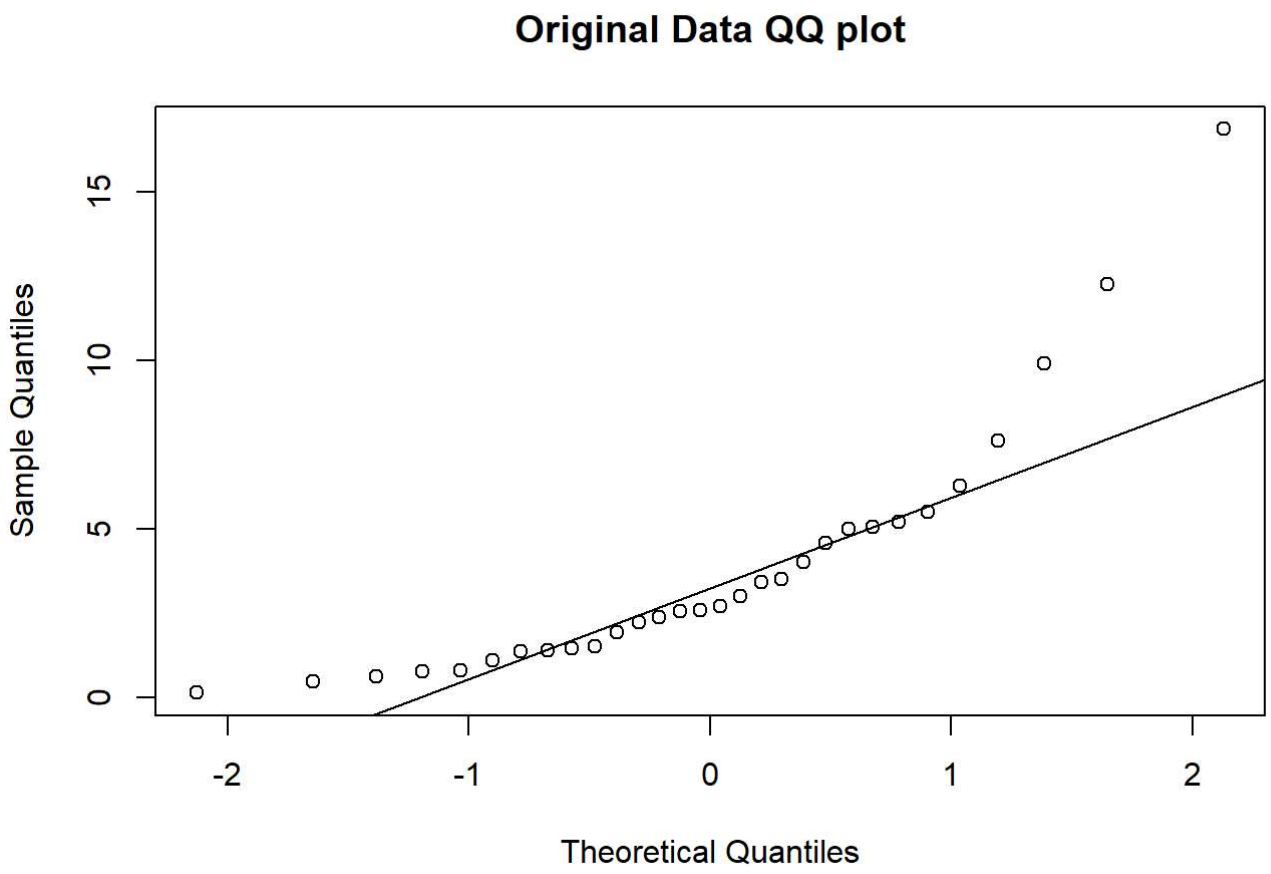

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.950390941215503 The data is not normal within si
gnificance levels of 0.05"
## [1] "V3"
```




```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.963410375639771 The data is not normal within si
gnificance levels of 0.05"
## [1] "V4"
```



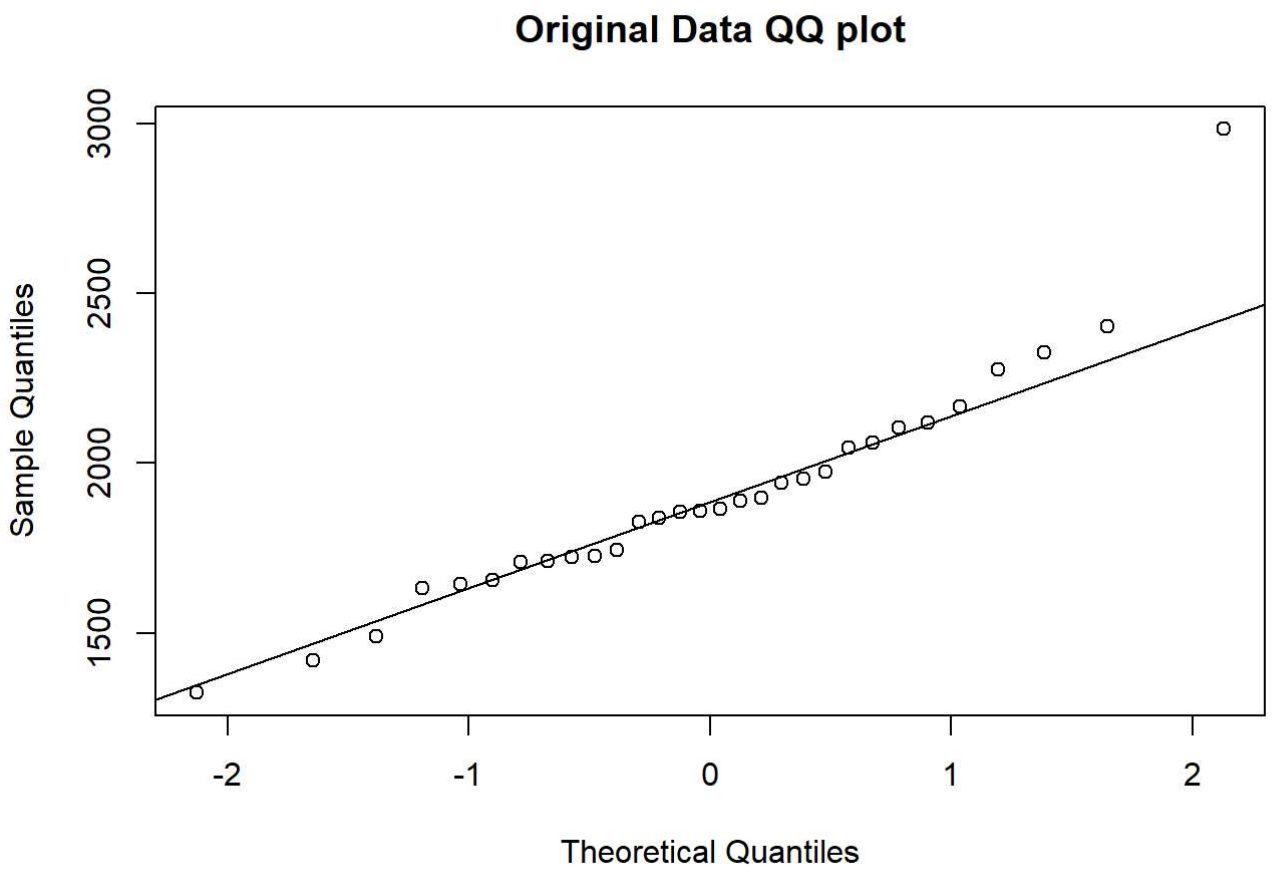

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.980292022100608 The data is normal within signif
icance levels of 0.05"
## [1] "V5"
```



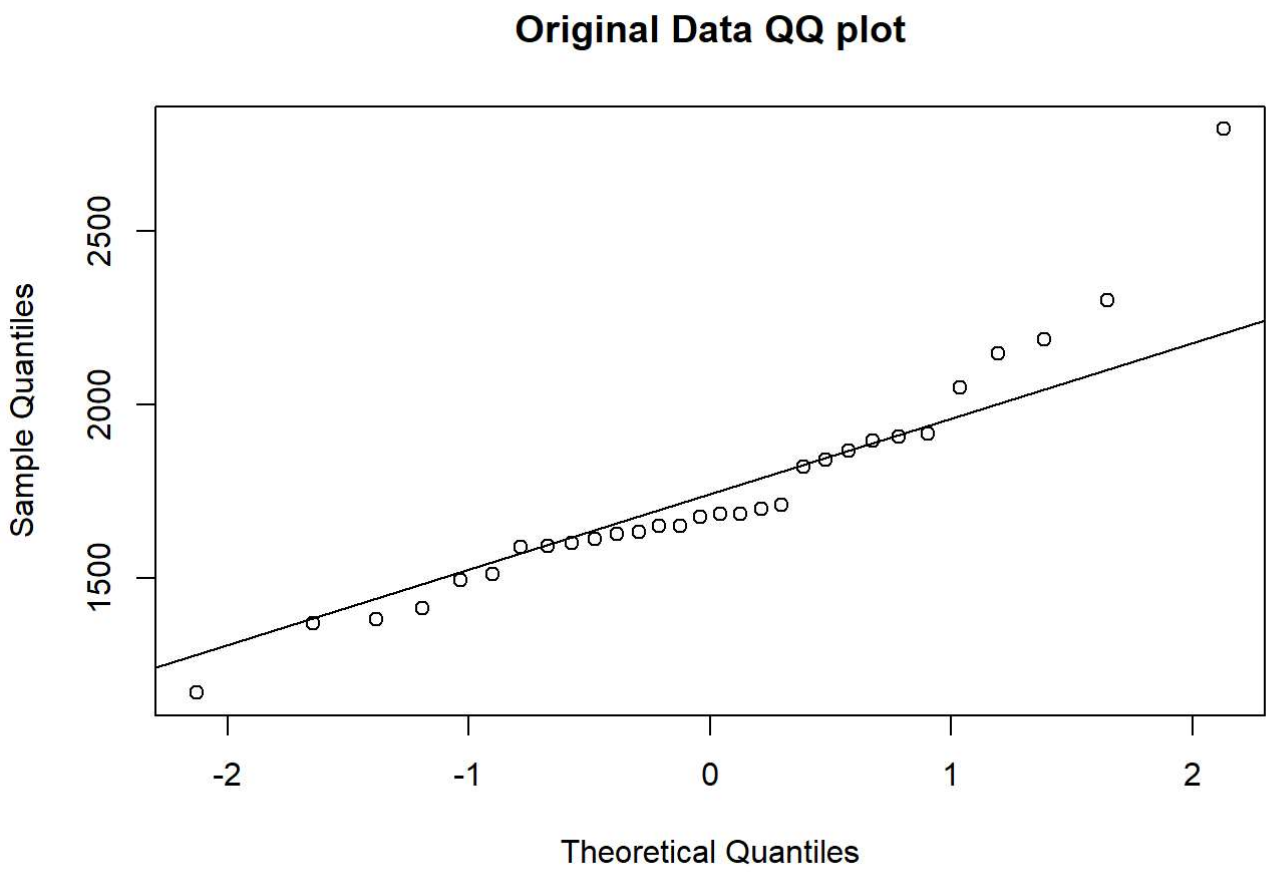

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.891223192804262 The data is not normal within si
gnificance levels of 0.05"
```

```
for (i in colnames(df)){
  print(i)
  test_norm(df[[i]], signigicance = 0.10)
}
```

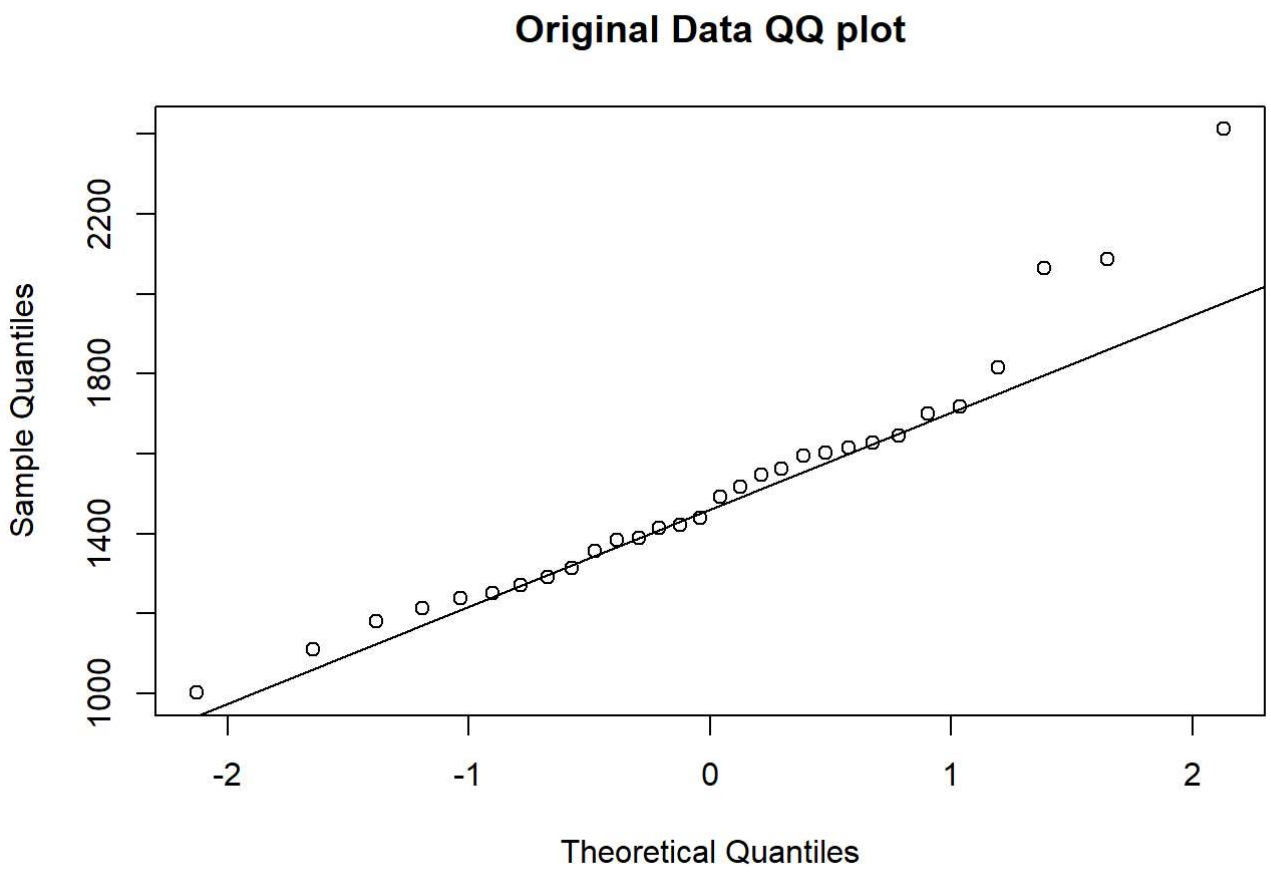
```
## [1] "V1"
```



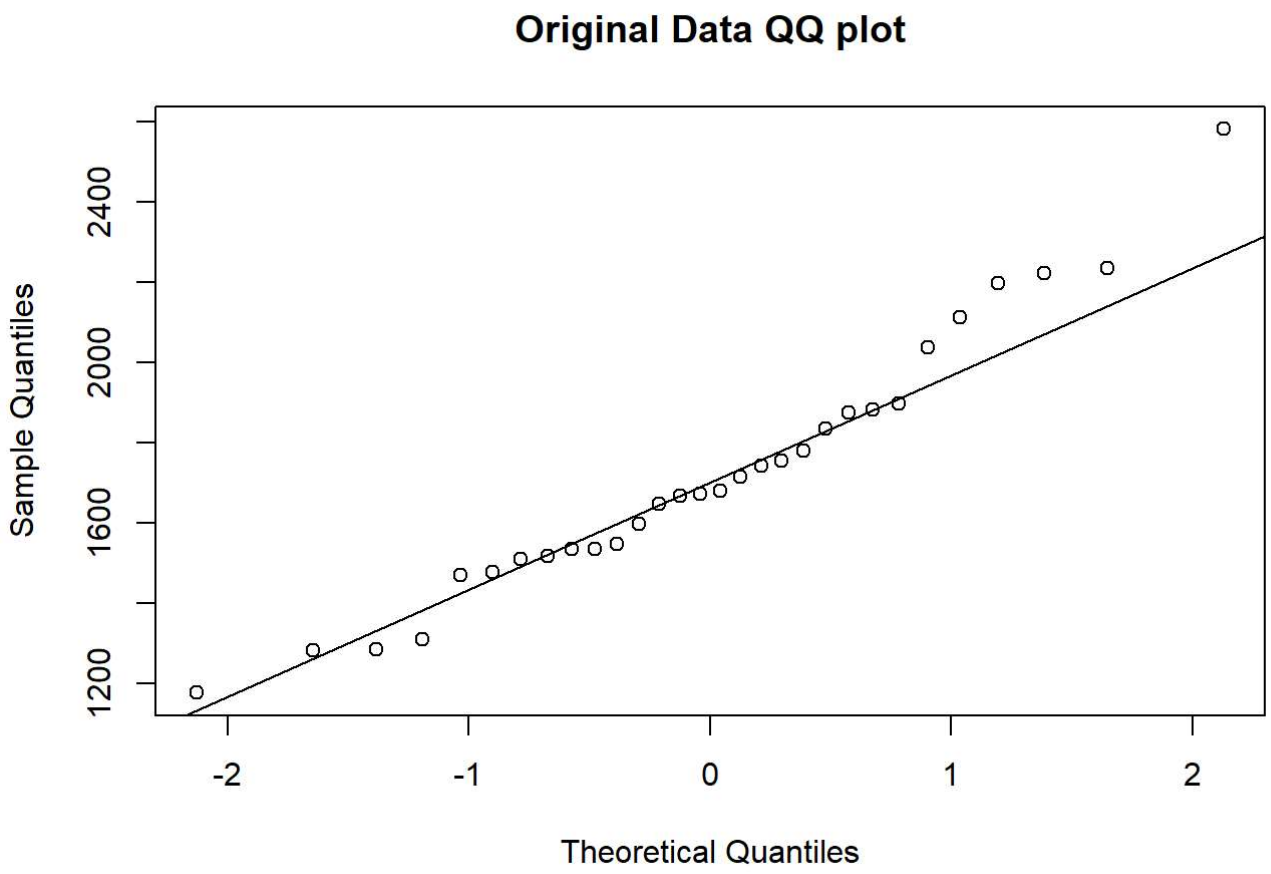

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.959864226926573 The data is not normal within si
gnificance levels of 0.1"
## [1] "V2"
```



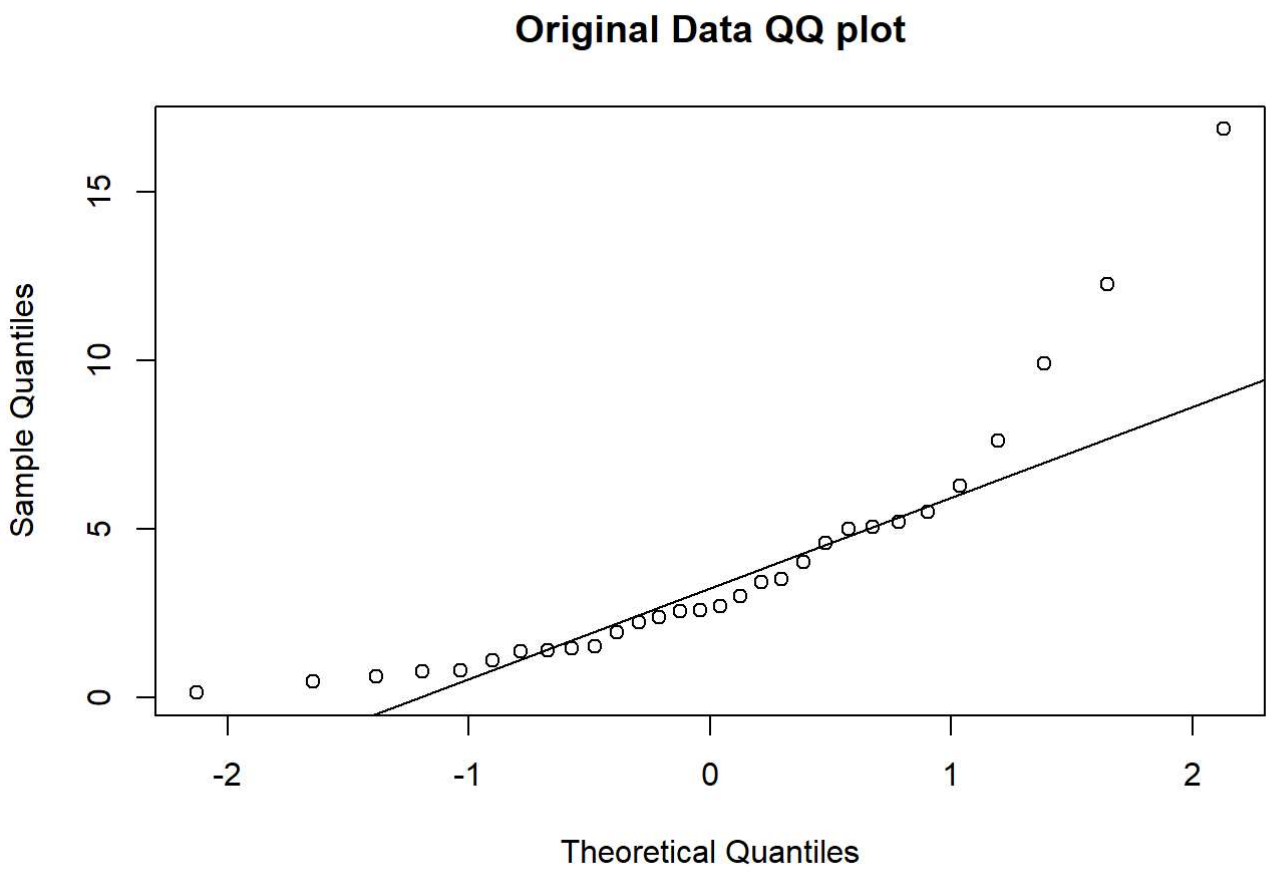

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.950390941215503 The data is not normal within si
gnificance levels of 0.1"
## [1] "V3"
```




```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.963410375639771 The data is not normal within si
gnificance levels of 0.1"
## [1] "V4"
```




```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.980292022100608 The data is normal within signif
icance levels of 0.1"
## [1] "V5"
```




```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.891223192804262 The data is not normal within si
gnificance levels of 0.1"
```

significance 0.01:
- x1: Normal
- x2: Normal
- x3: normal
- x4: normal
- x5: NOT normal

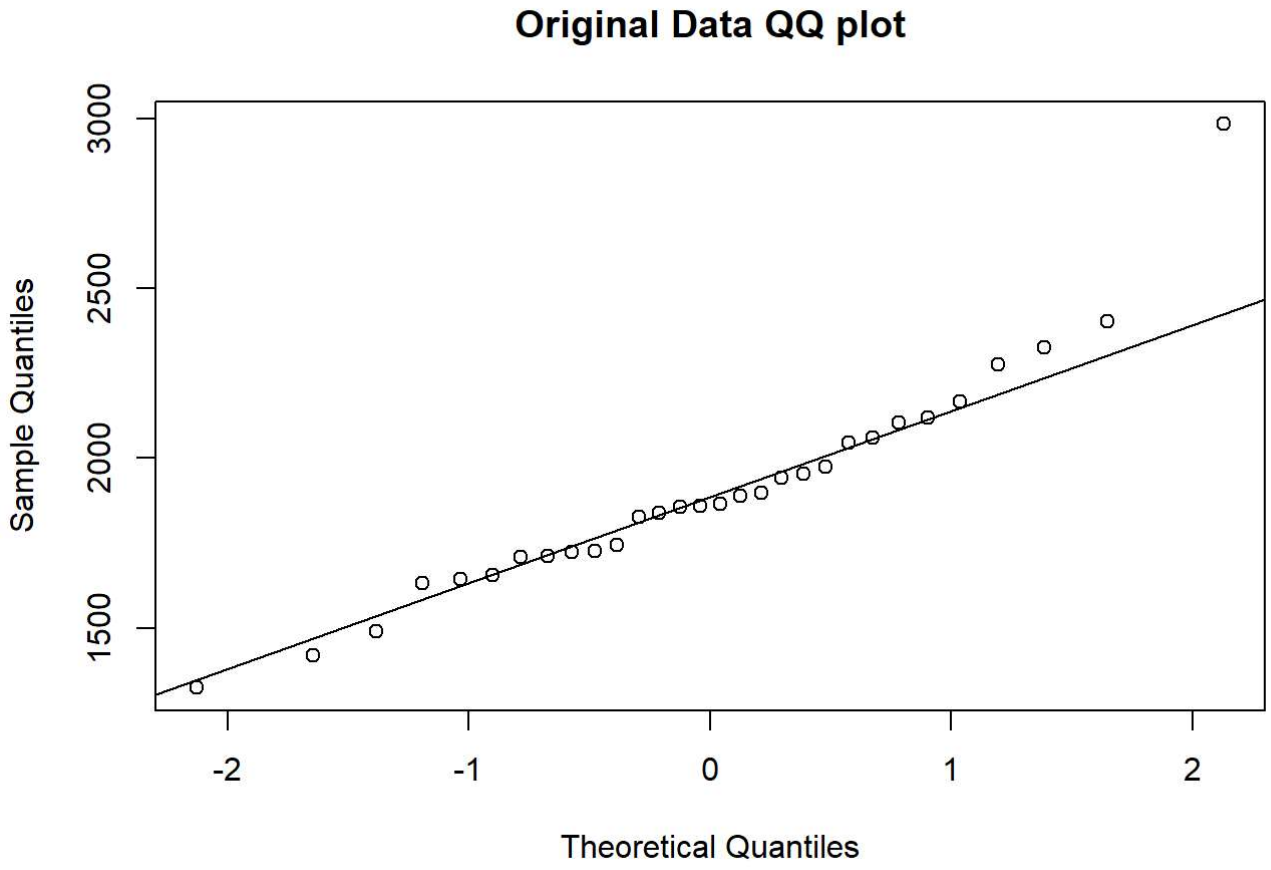
significance: 0.05:
- x1: NOT Normal
- x2: NOT Normal
- x3: NOT normal
- x4: normal
- x5: NOT normal

significance: 0.10:
- x1: NOT Normal
- x2: NOT Normal
- x3: NOT normal
- x4: normal
- x5: NOT normal

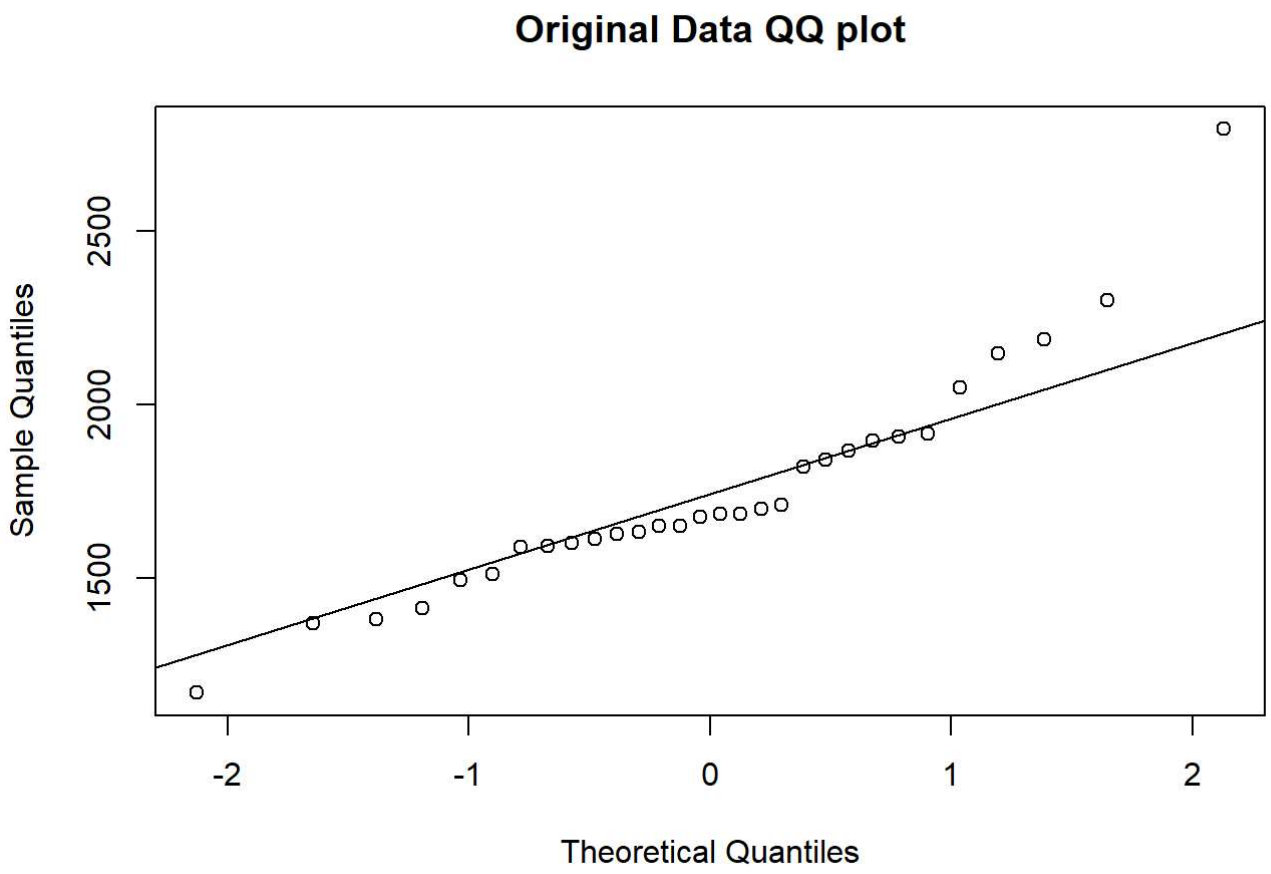
No matter what significance level there is always at least one attribute which is not normally distributed. Maybe we should transform it all? Let's try:

```
for (i in colnames(df)){
  print(i)
  test_norm(df[,i], signigicance = 0.01, transform_data = TRUE)
}
```

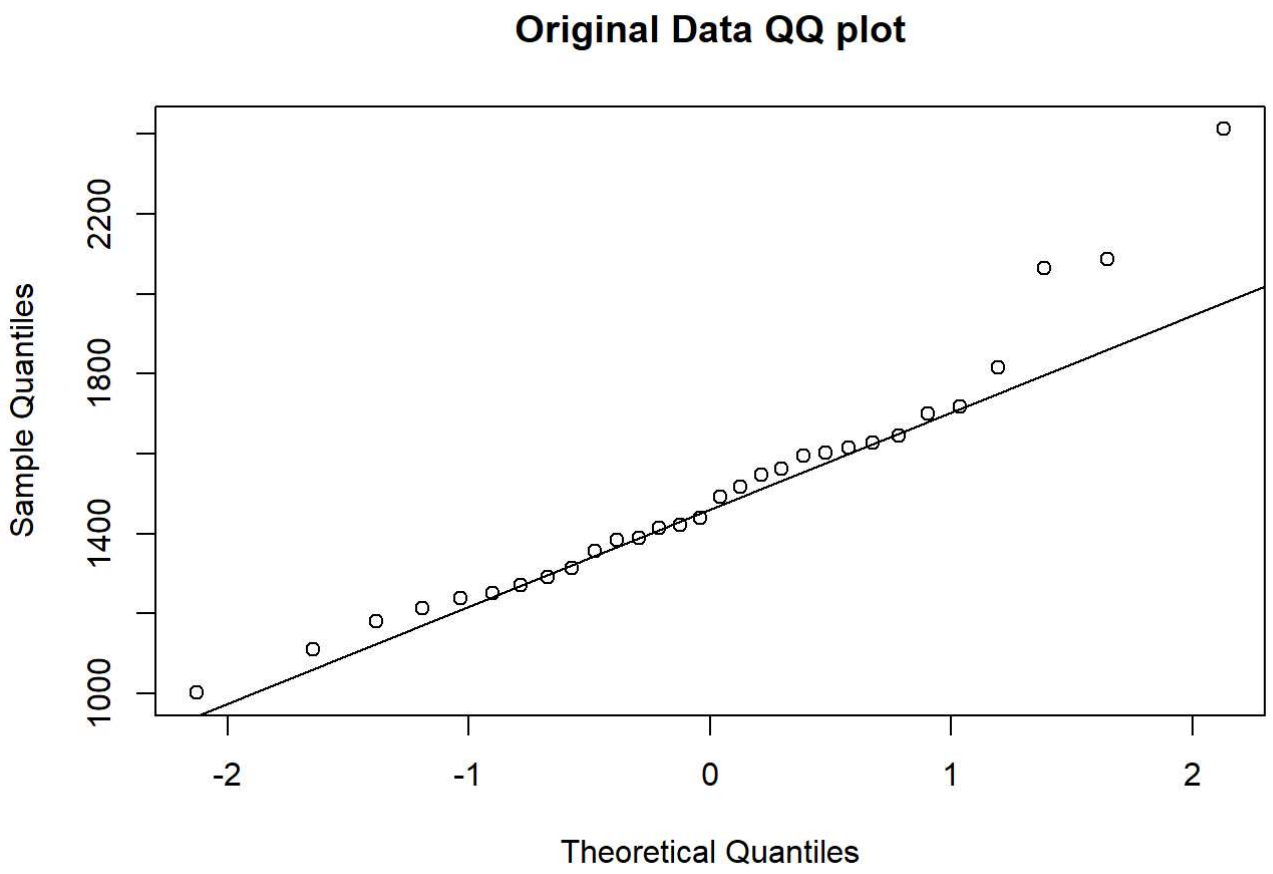
```
## [1] "V1"
```



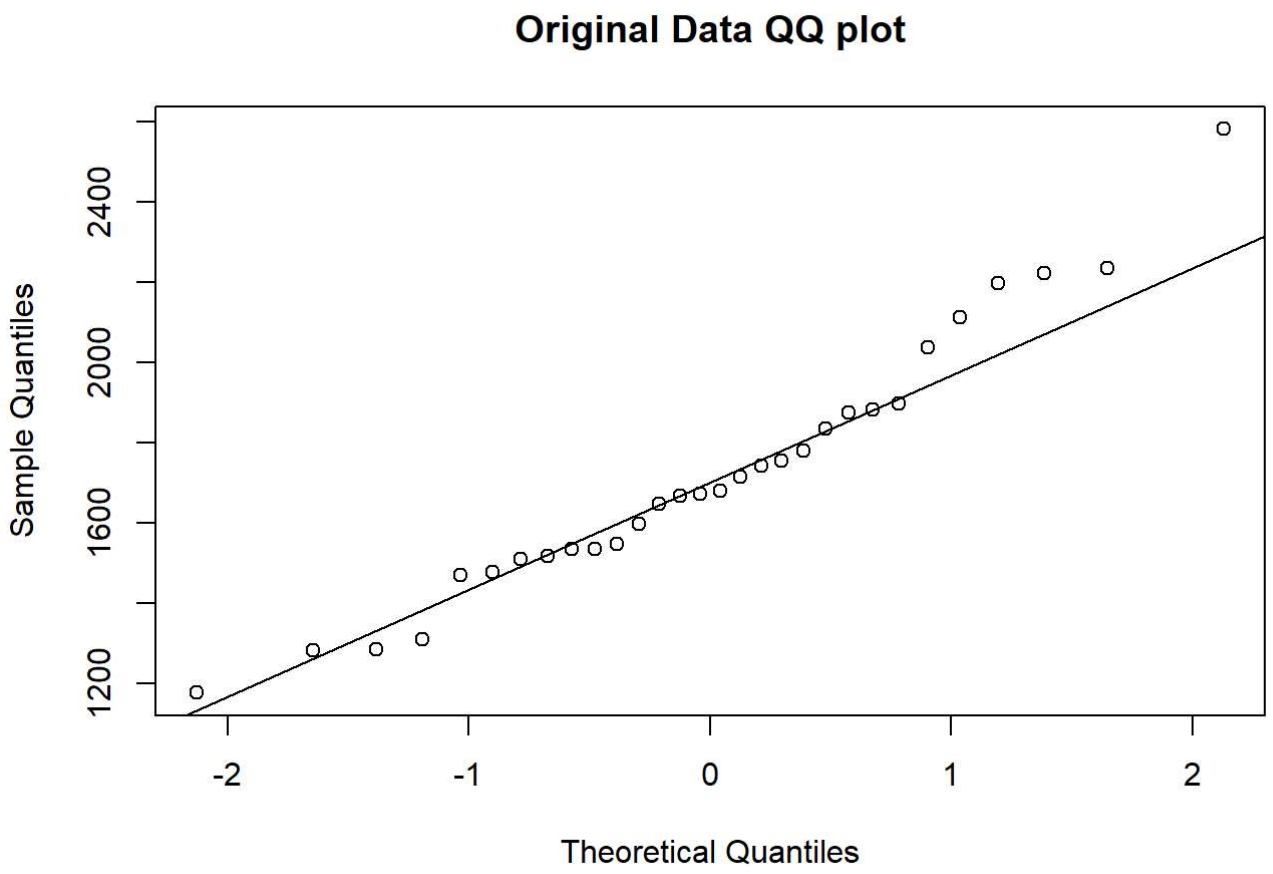

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.959864226926573 The data is normal within signif
icance levels of 0.01"
## [1] "V2"
```



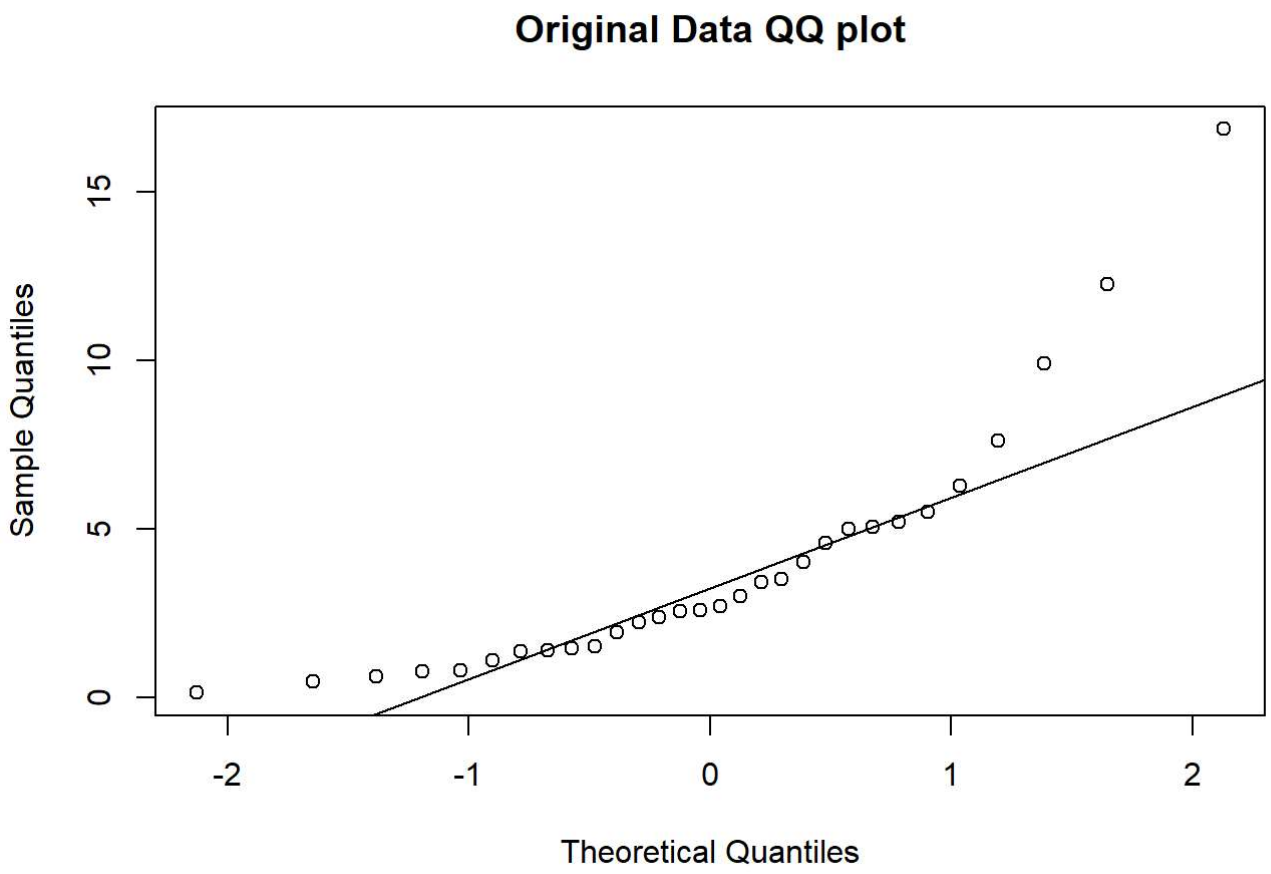

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.950390941215503 The data is normal within signif
icance levels of 0.01"
## [1] "V3"
```



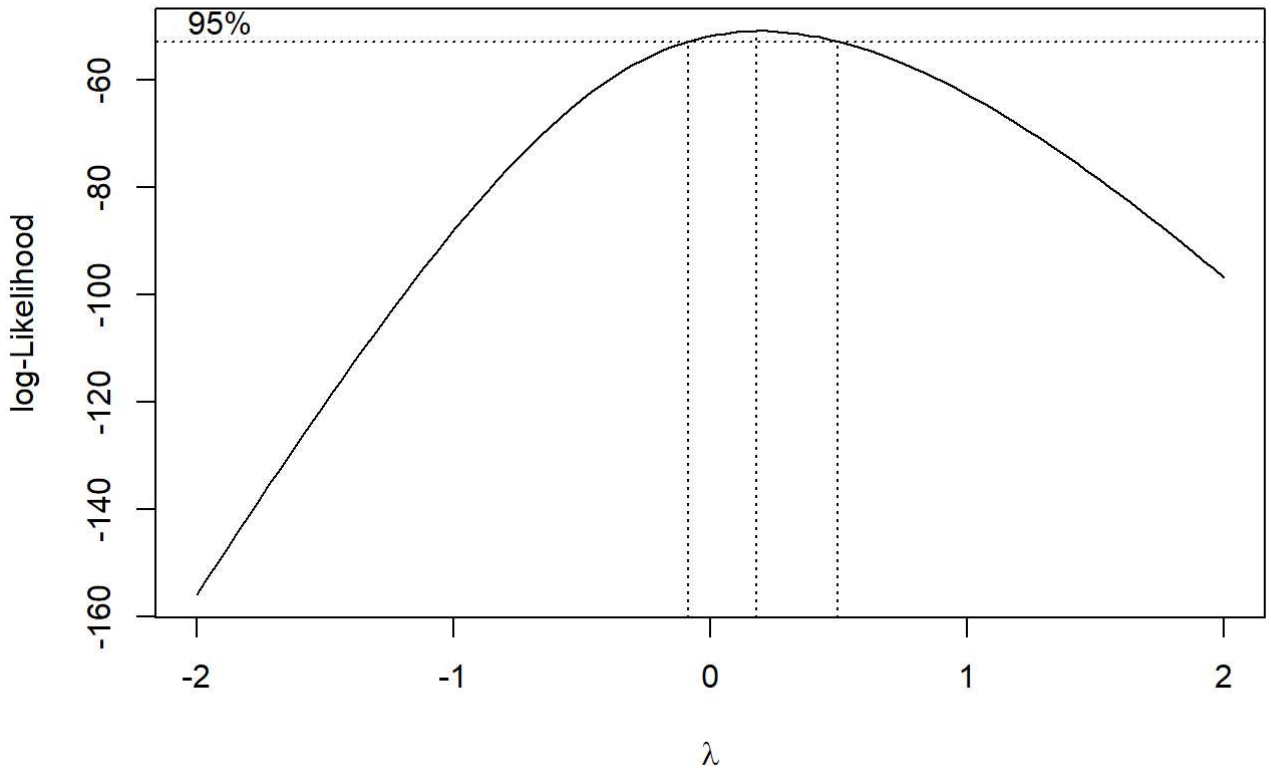

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.963410375639771 The data is normal within signif
icance levels of 0.01"
## [1] "V4"
```



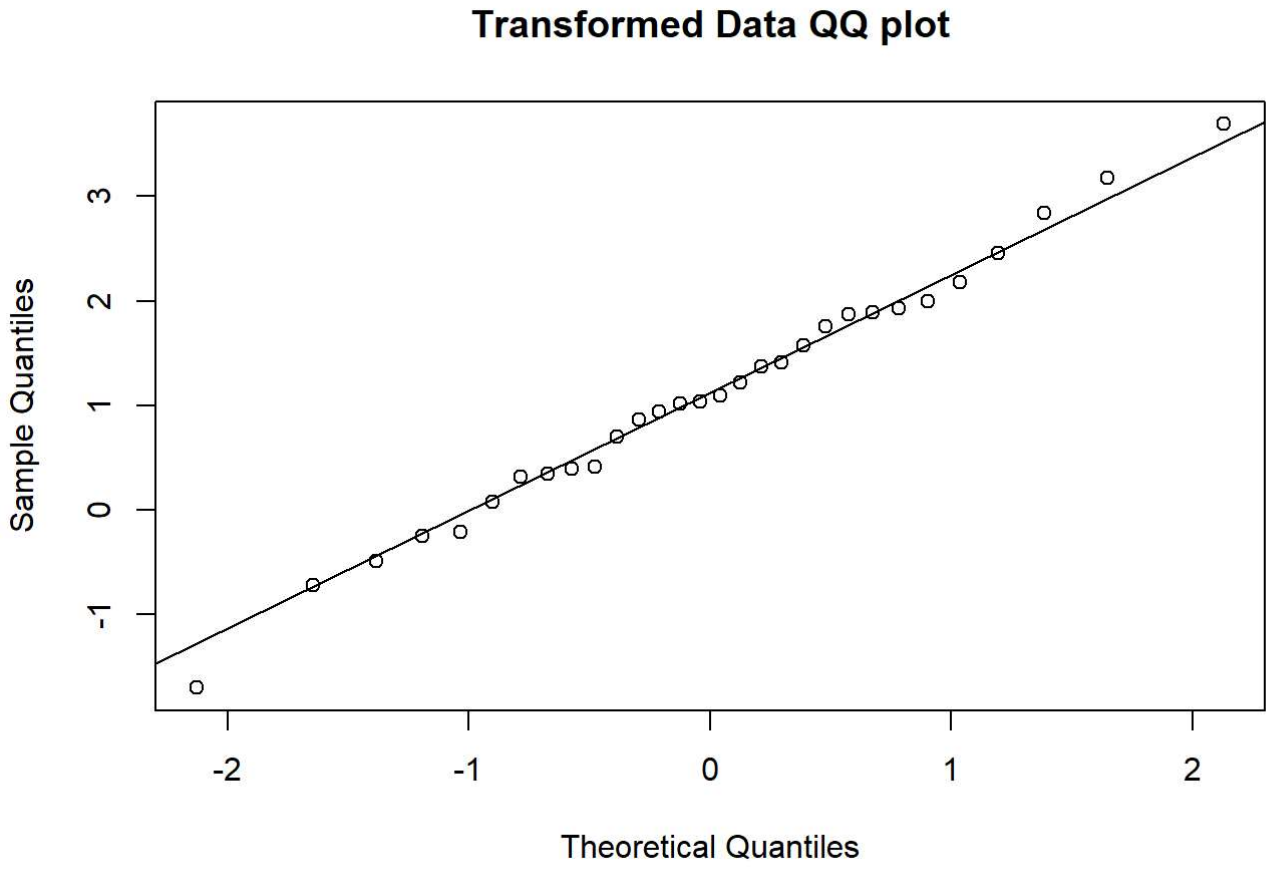

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.980292022100608 The data is normal within signif
icance levels of 0.01"
## [1] "V5"
```




```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.891223192804262 The data is not normal within si
gnificance levels of 0.01 Transforming data ... "
```



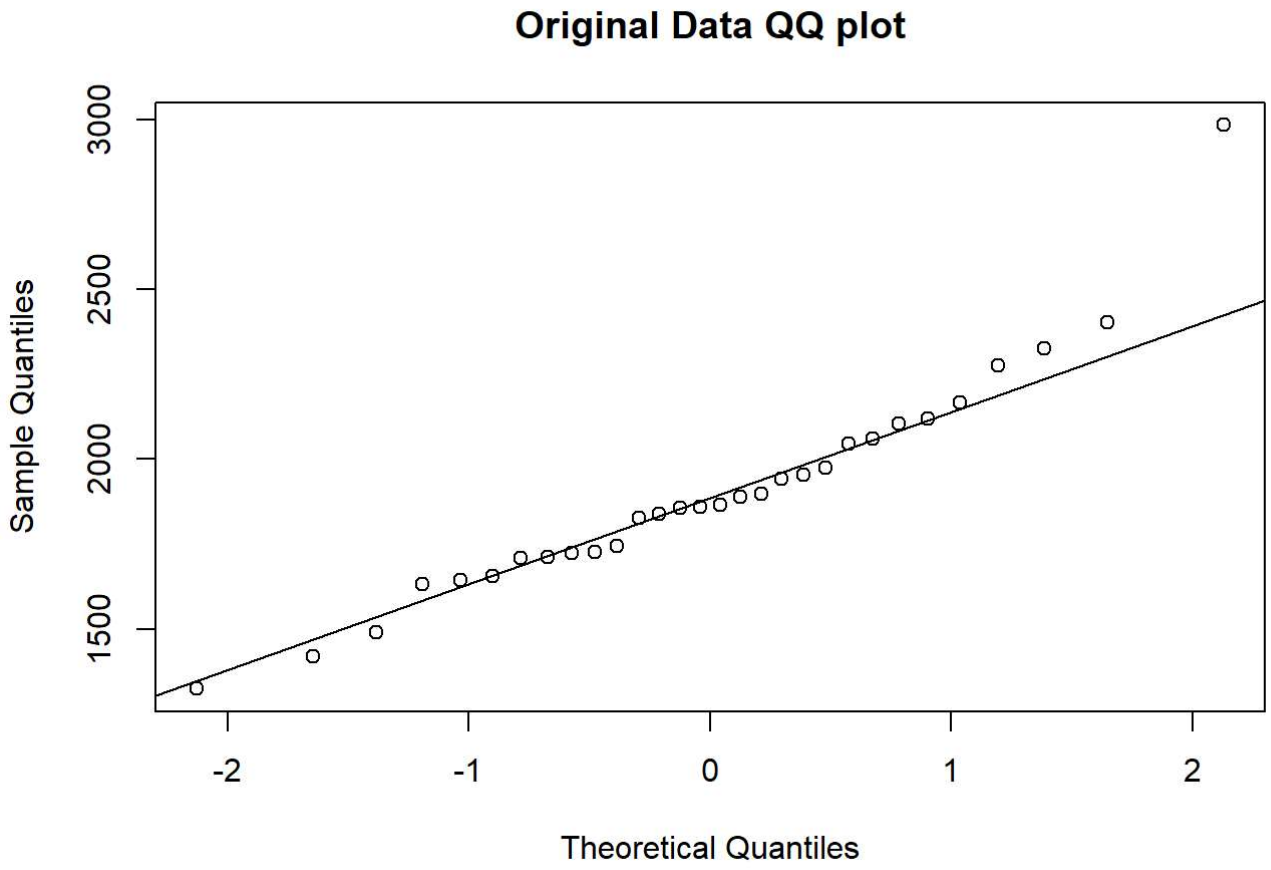

```
## [1] "The best lambda is  0.181818181818182"
```



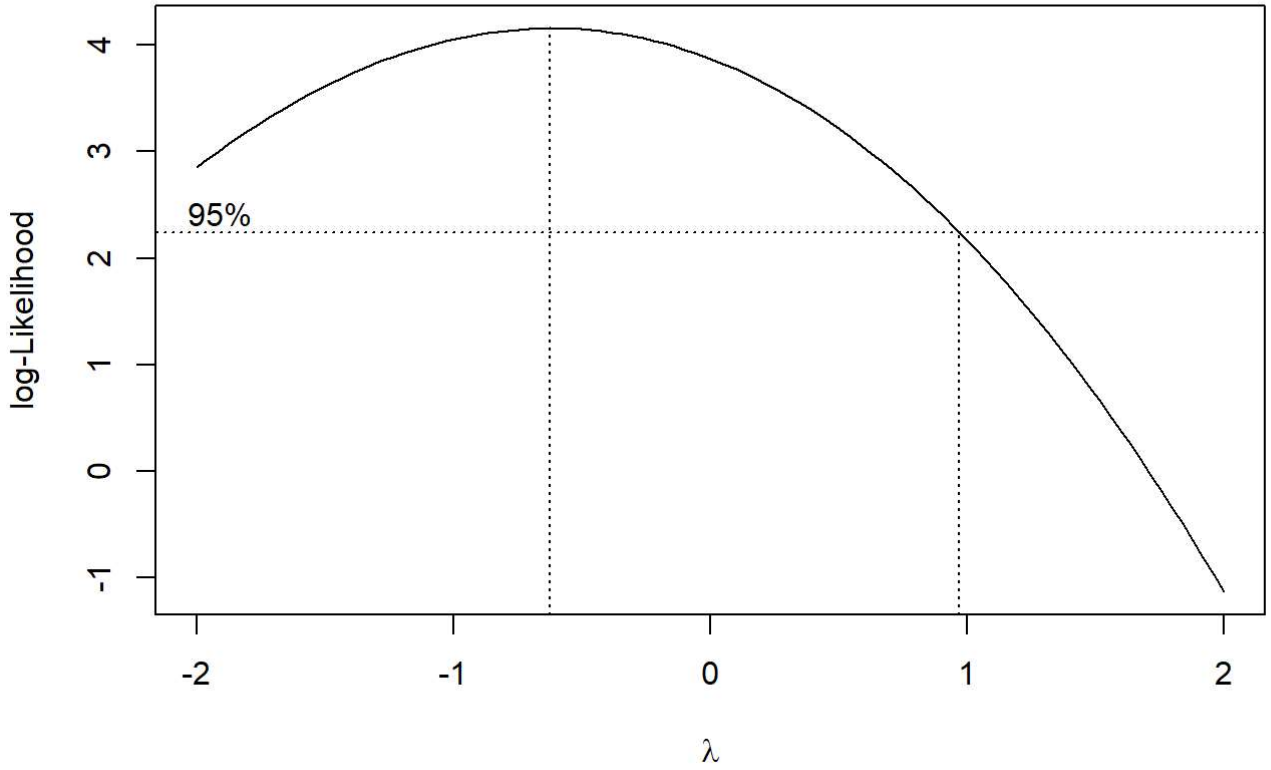
```
## [1] "With a correlation coefficient of  0.996412803498558 The data is normal within signif
icance levels of 0.01 For the box cox transformed data"
```

```
for (i in colnames(df)){
  print(i)
  test_norm(df[[i]], signigicance = 0.05, transform_data = TRUE)
}
```

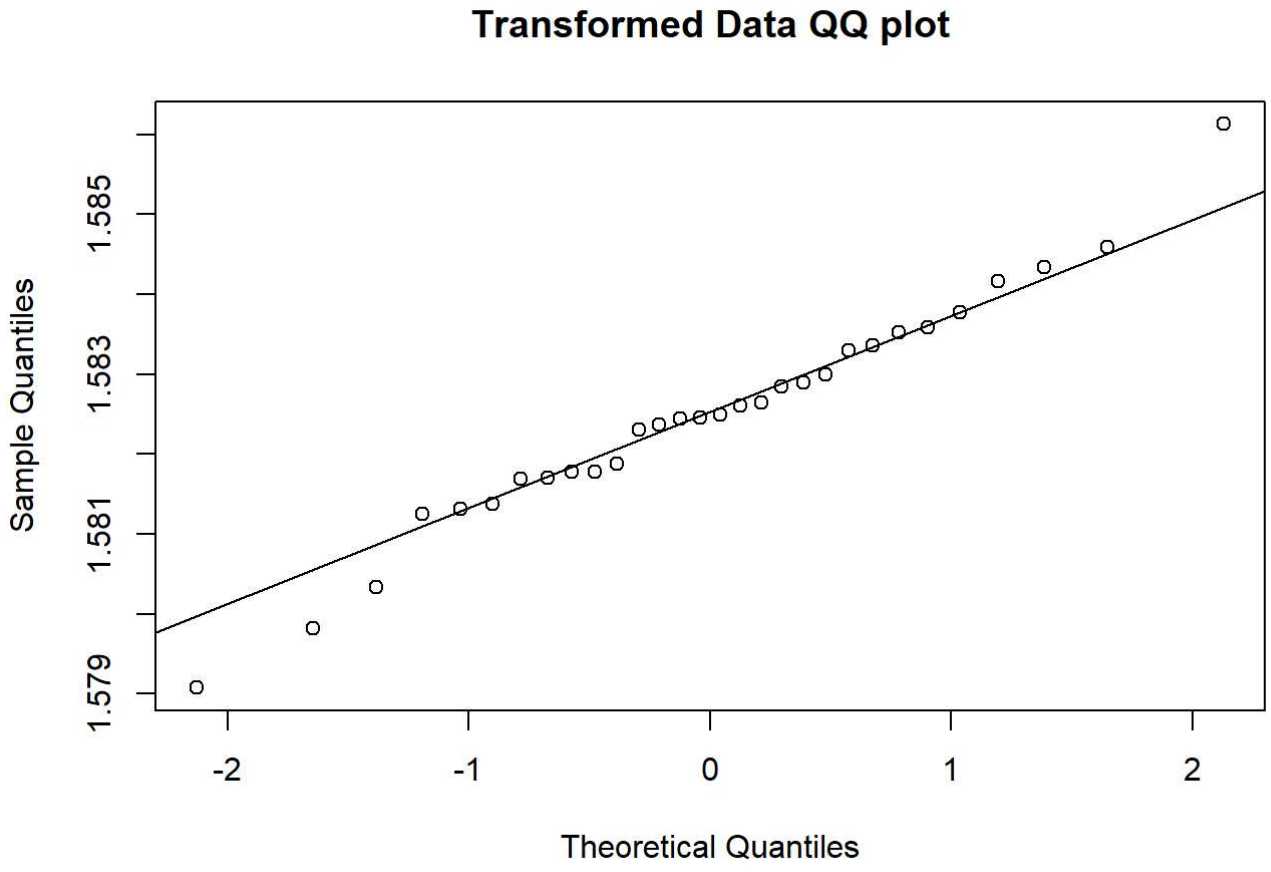
```
## [1] "V1"
```



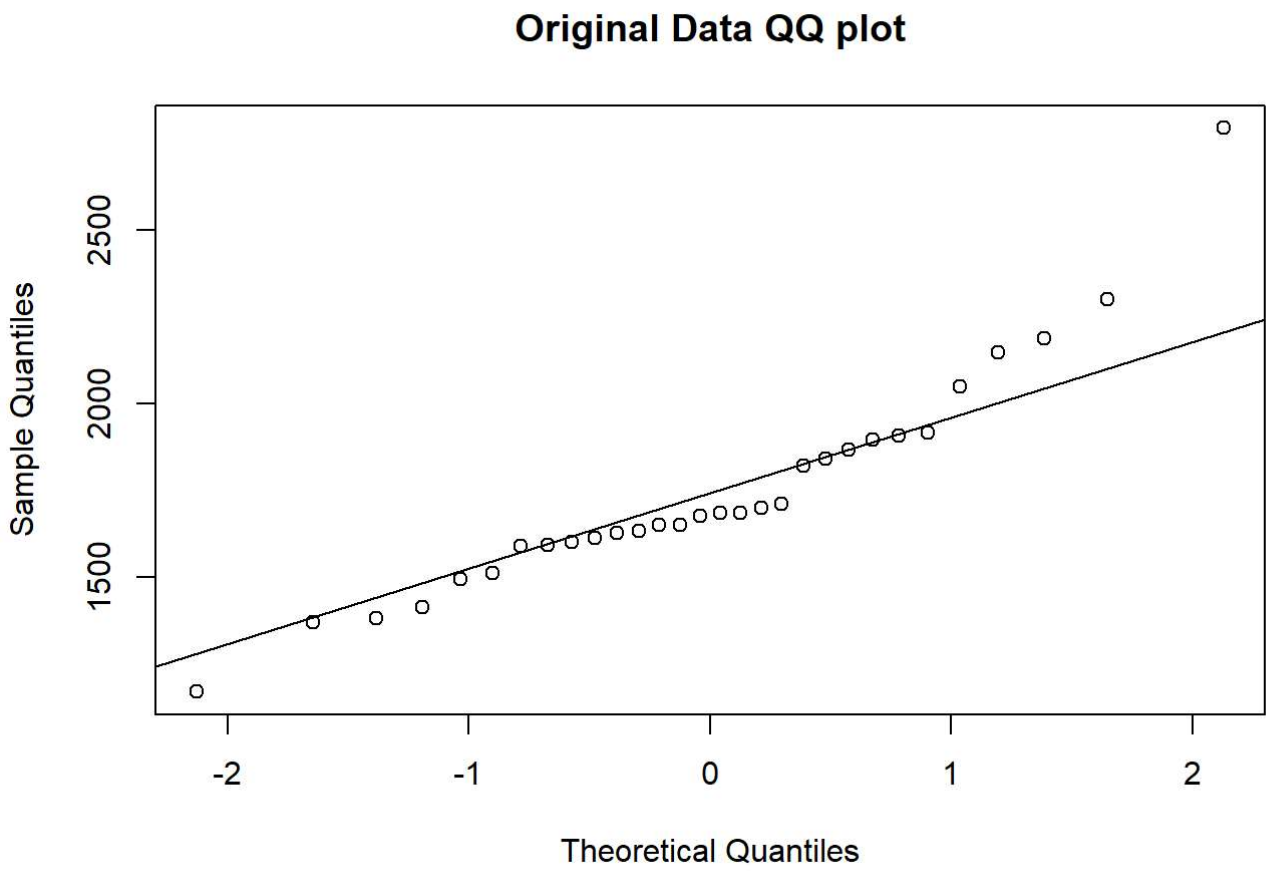

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.959864226926573 The data is not normal within si
gnificance levels of 0.05 Transforming data ... "
```



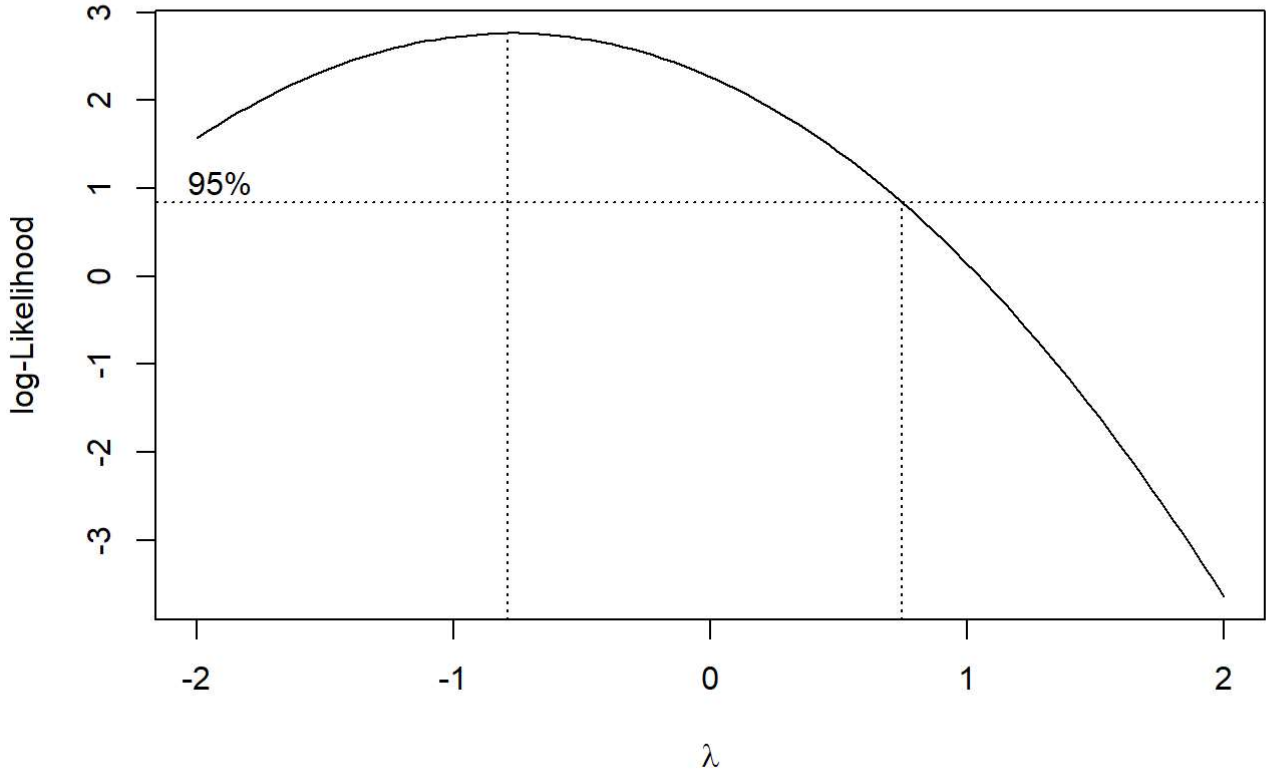

```
## [1] "The best lambda is  -0.626262626262626"
```



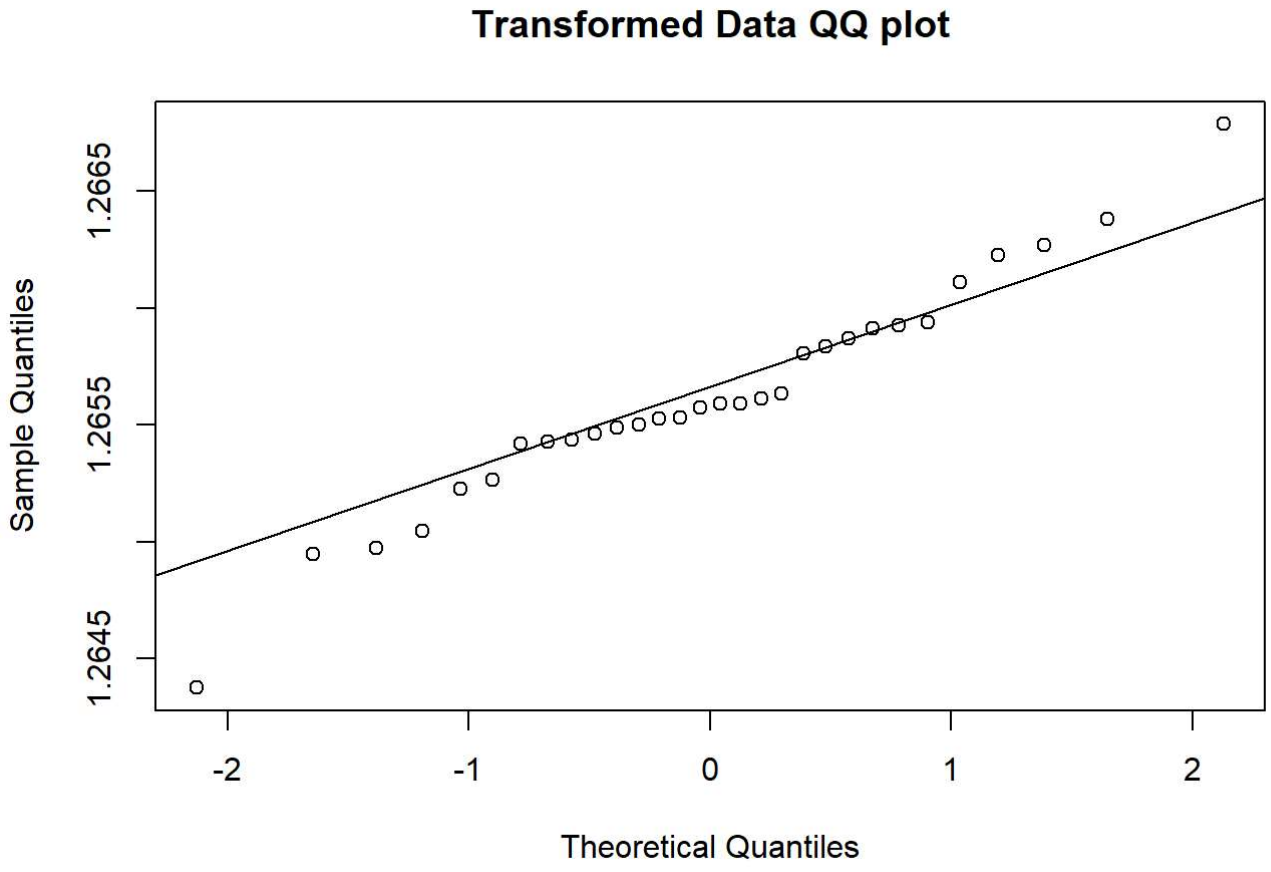
```
## [1] "With a correlation coefficient of  0.987535483360367 The data is normal within signif
icance levels of 0.05 For the box cox transformed data"
## [1] "V2"
```



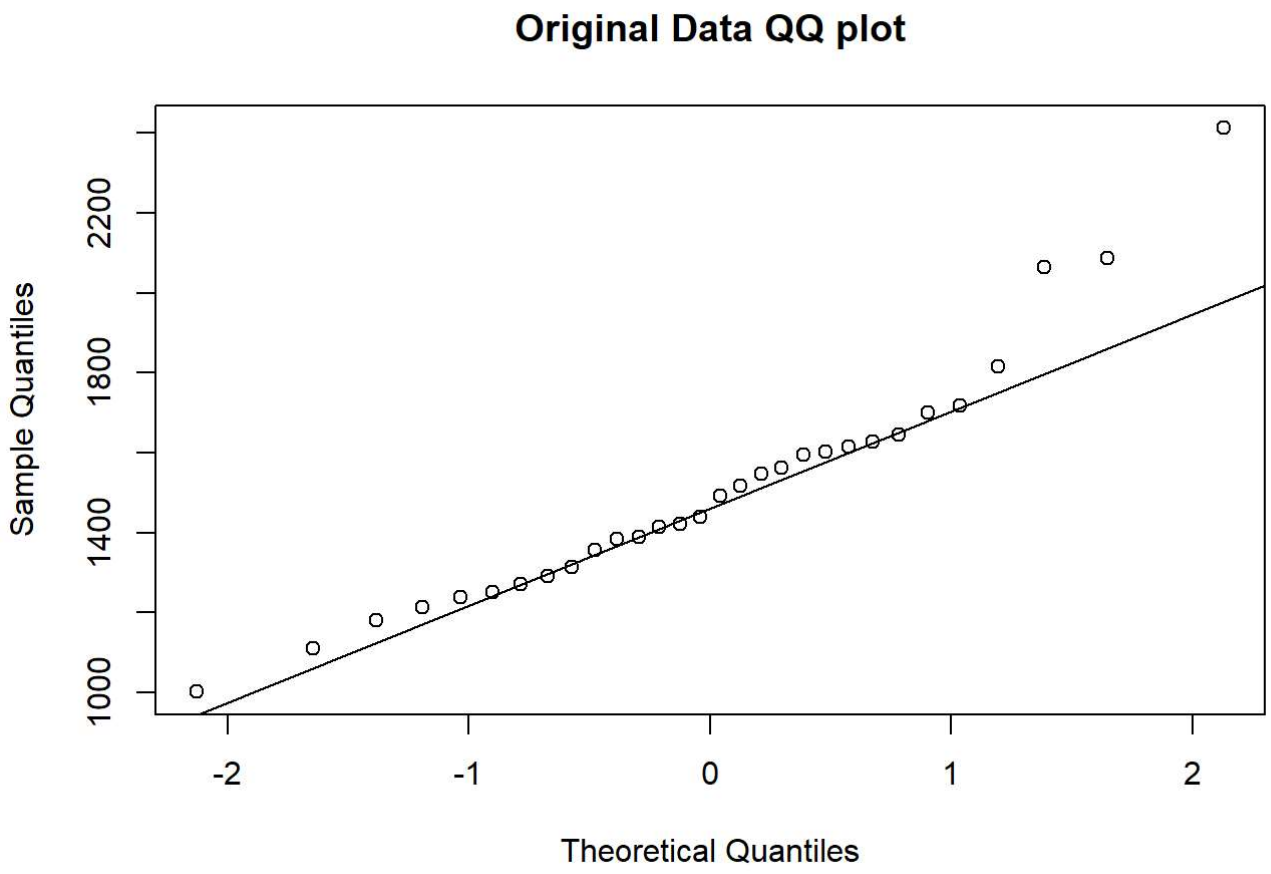

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.950390941215503 The data is not normal within si
gnificance levels of 0.05 Transforming data ... "
```



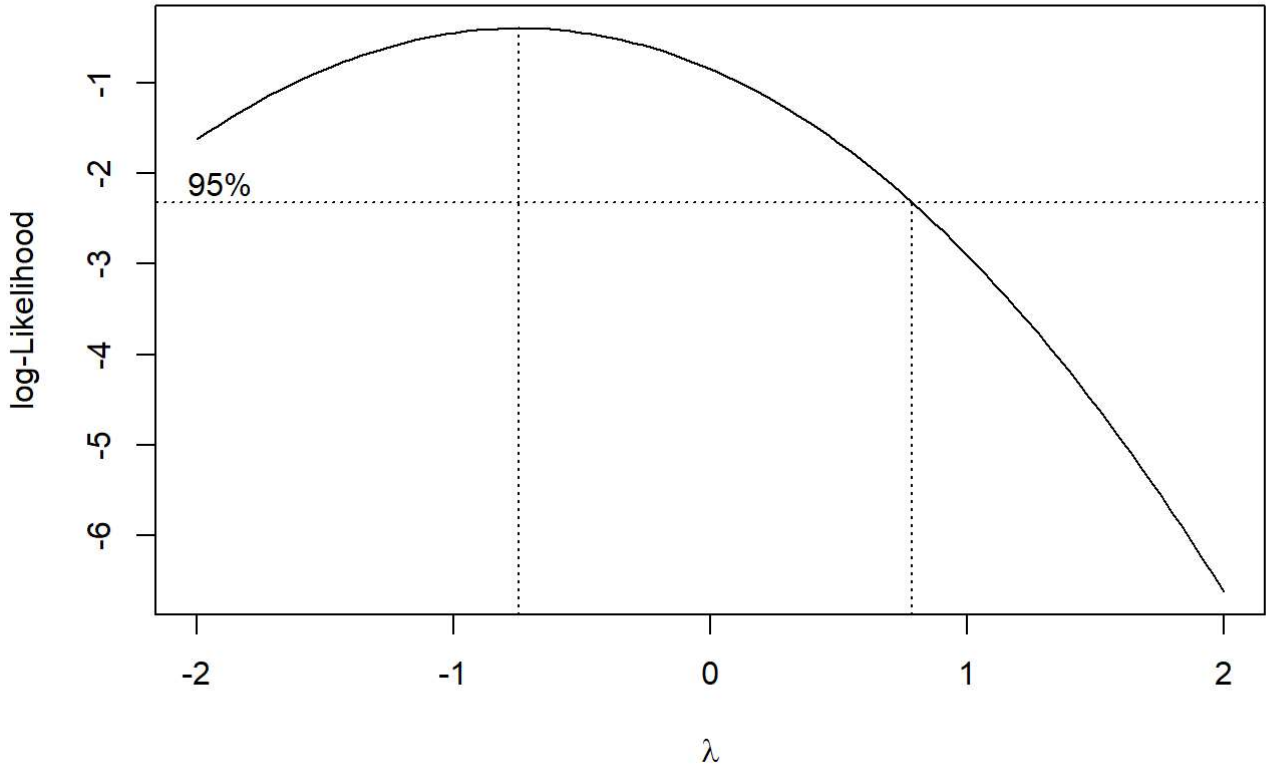

```
## [1] "The best lambda is  -0.787878787878788"
```



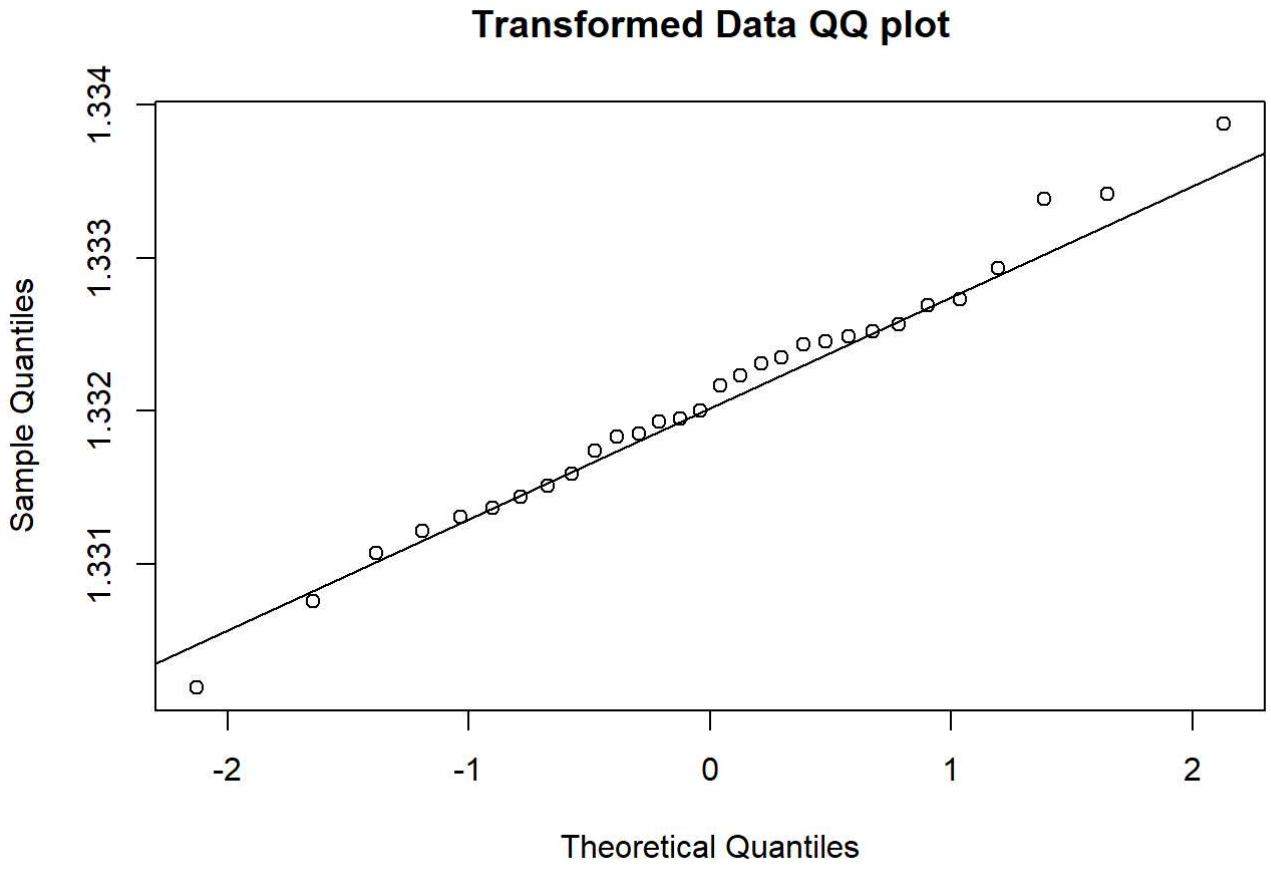
```
## [1] "With a correlation coefficient of  0.982798262680497 The data is normal within signif
icance levels of 0.05 For the box cox transformed data"
## [1] "V3"
```



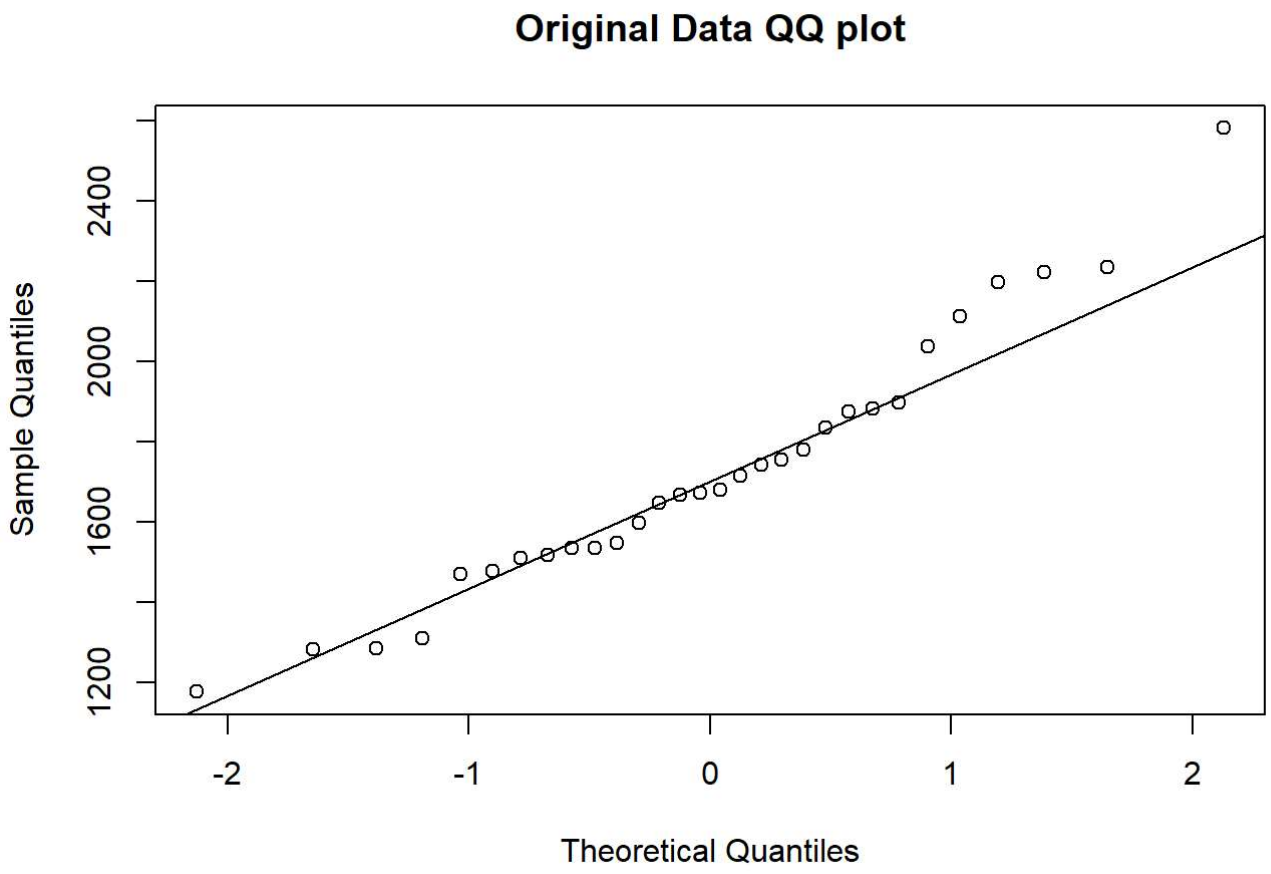

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.963410375639771 The data is not normal within si
gnificance levels of 0.05 Transforming data ... "
```



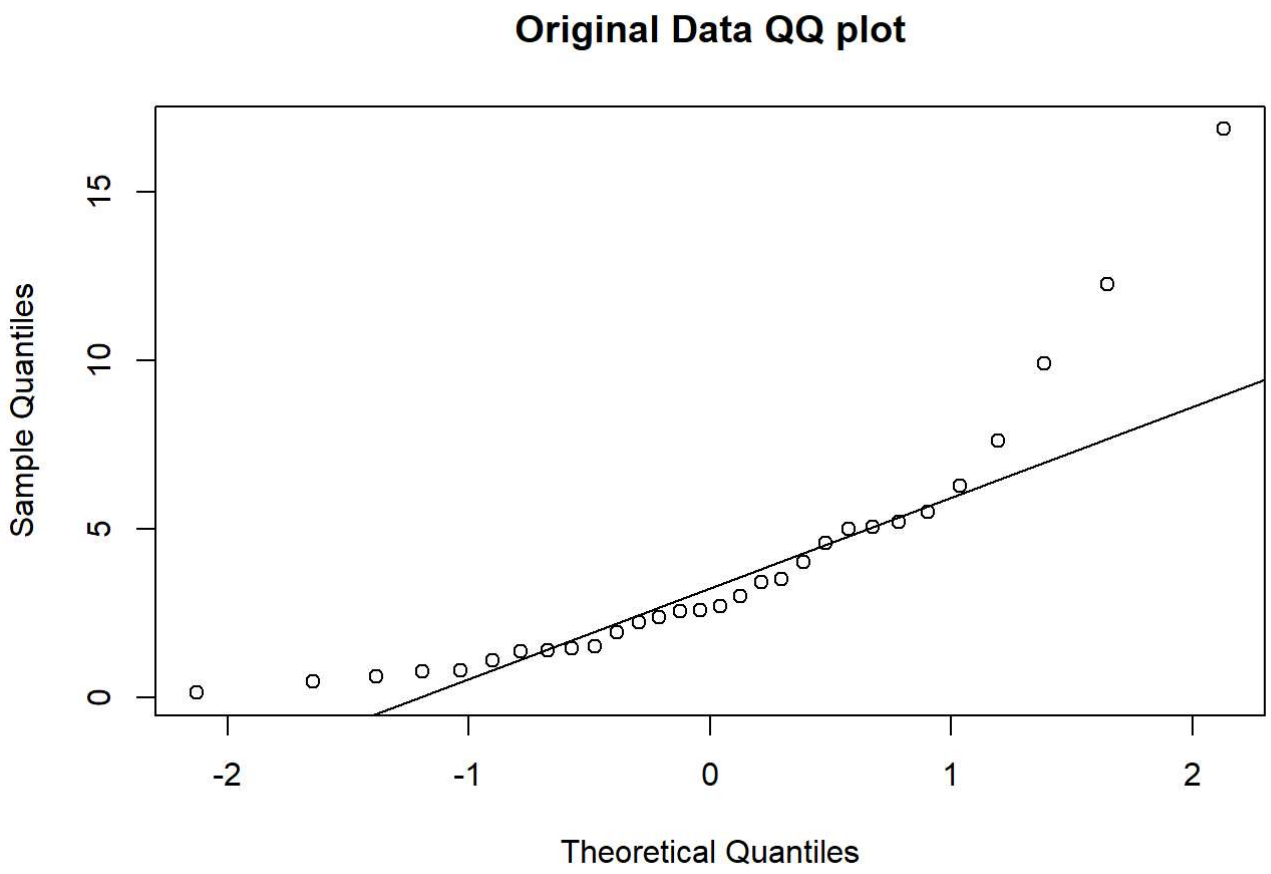

```
## [1] "The best lambda is  -0.747474747474747"
```



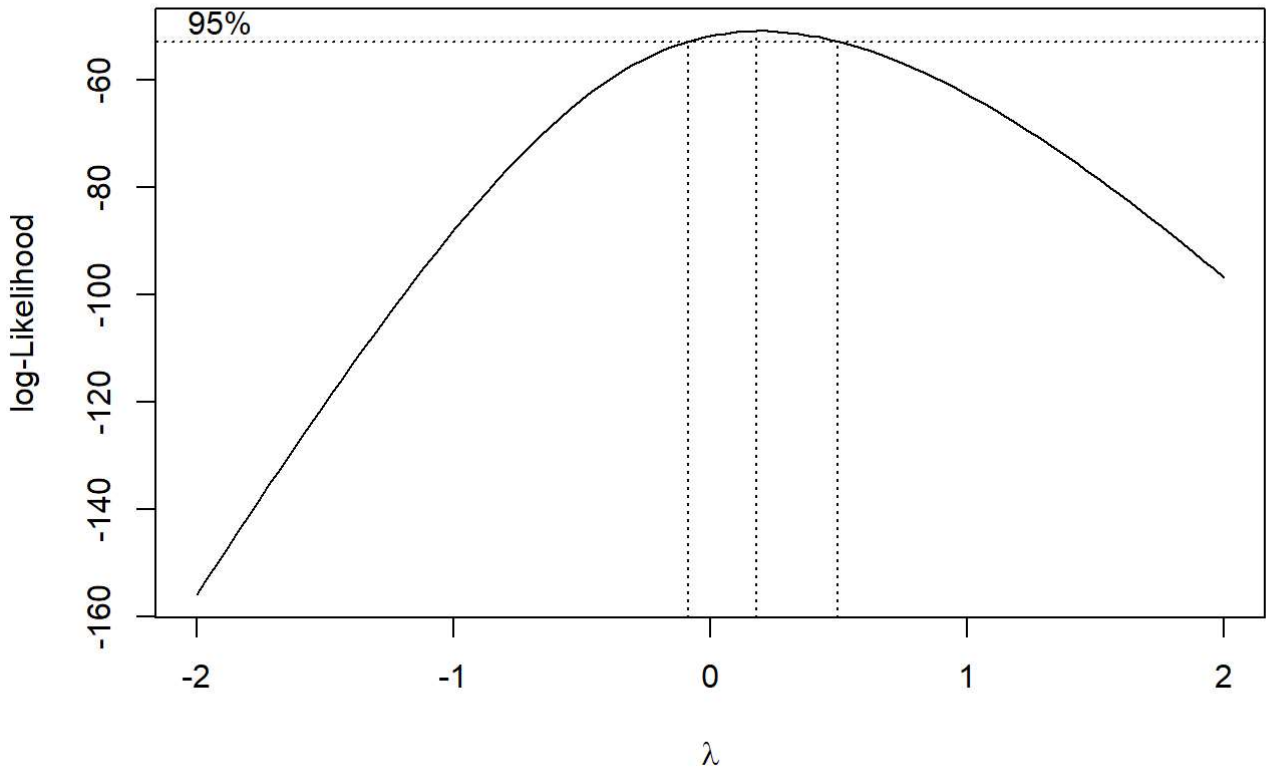
```
## [1] "With a correlation coefficient of  0.994189192711897 The data is normal within signif
icance levels of 0.05 For the box cox transformed data"
## [1] "V4"
```



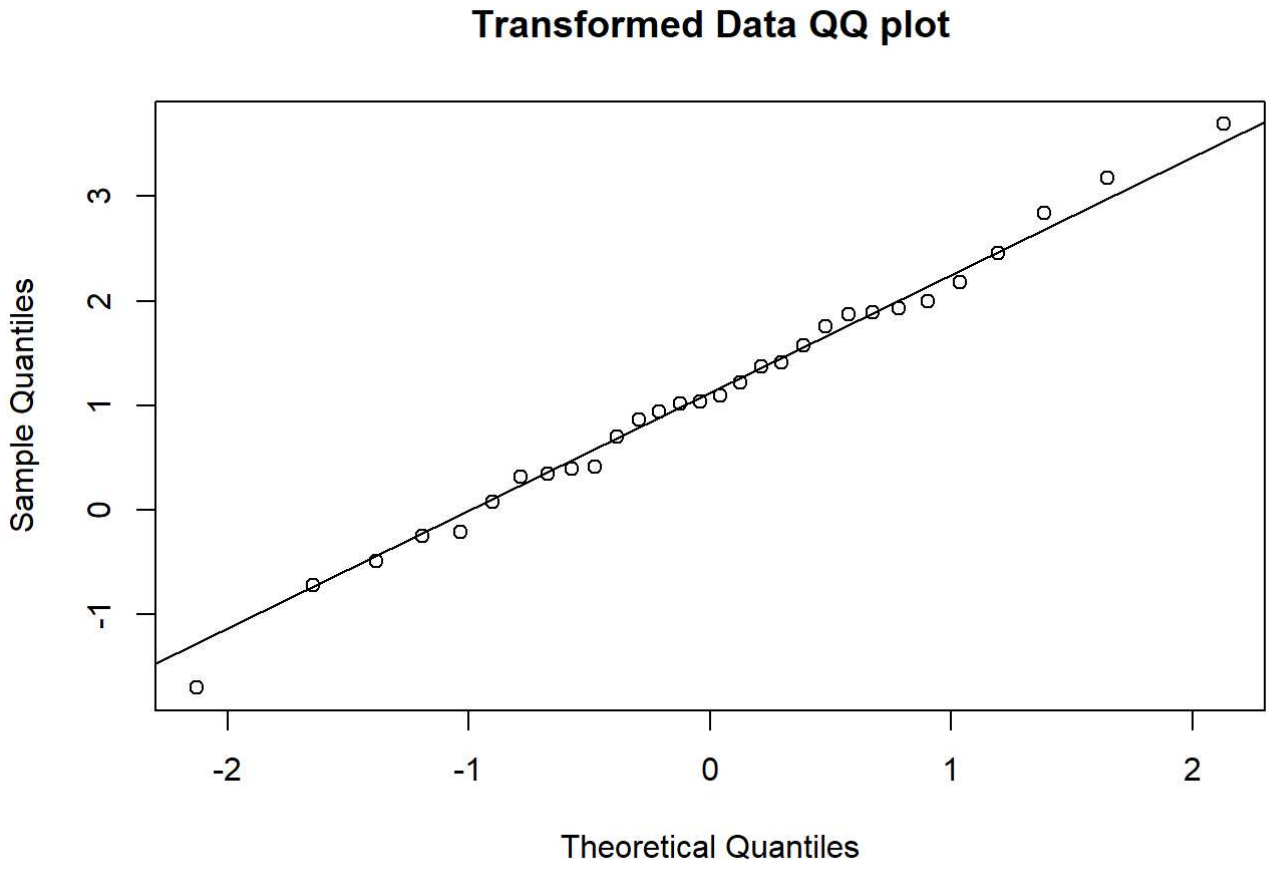

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.980292022100608 The data is normal within signif
icance levels of 0.05"
## [1] "V5"
```




```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.891223192804262 The data is not normal within si
gnificance levels of 0.05 Transforming data ... "
```



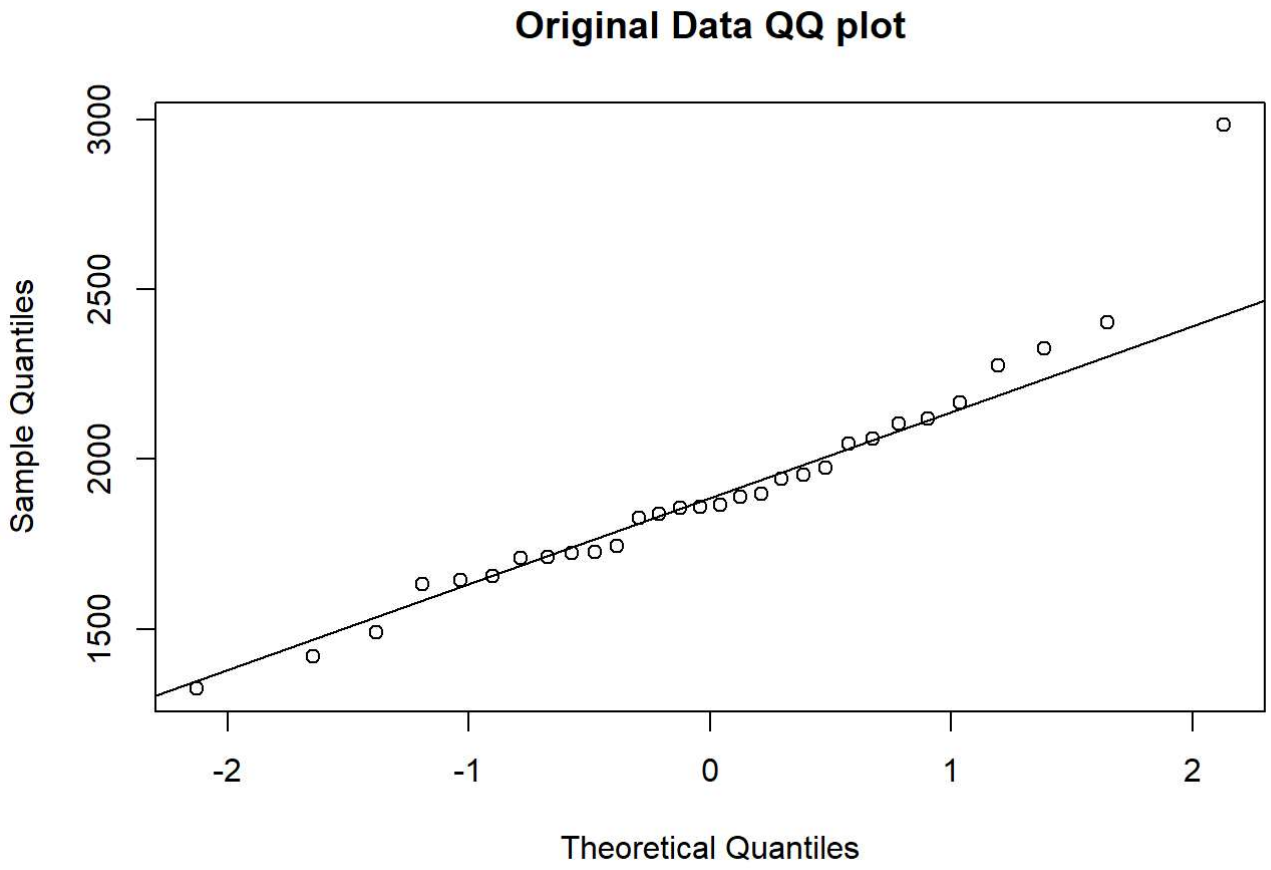

```
## [1] "The best lambda is  0.181818181818182"
```



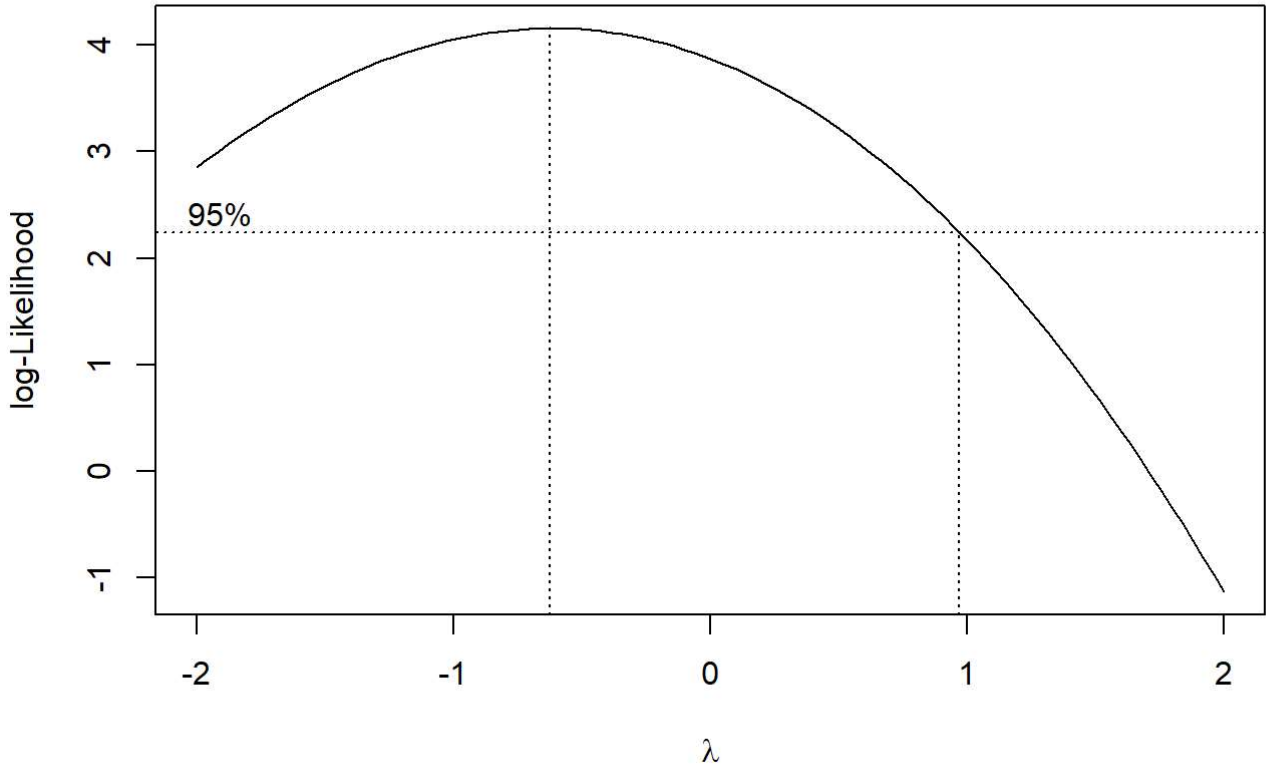
```
## [1] "With a correlation coefficient of  0.996412803498558 The data is normal within signif
icance levels of 0.05 For the box cox transformed data"
```

```
for (i in colnames(df)){
  print(i)
  test_norm(df[[i]], signigicance = 0.10, transform_data = TRUE)
}
```

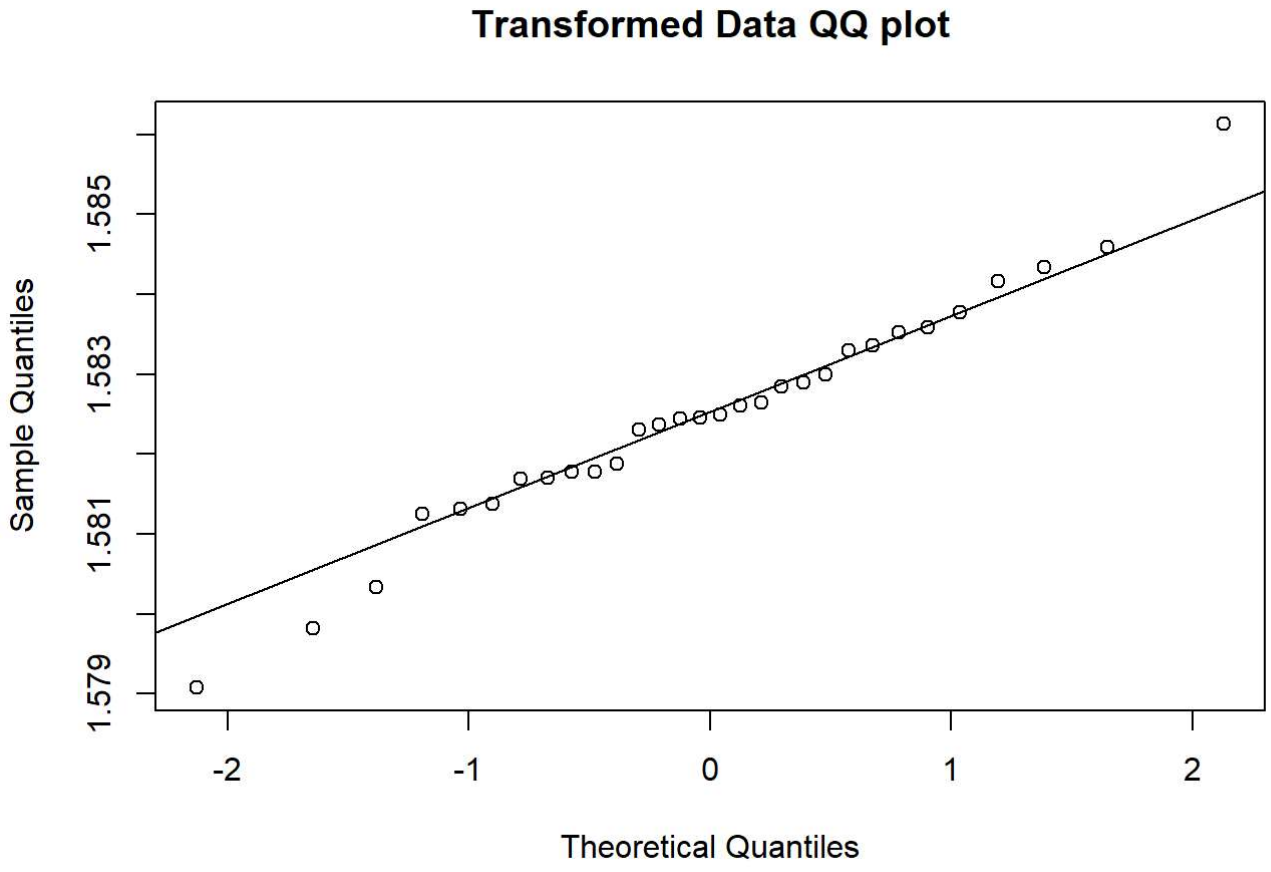
```
## [1] "V1"
```



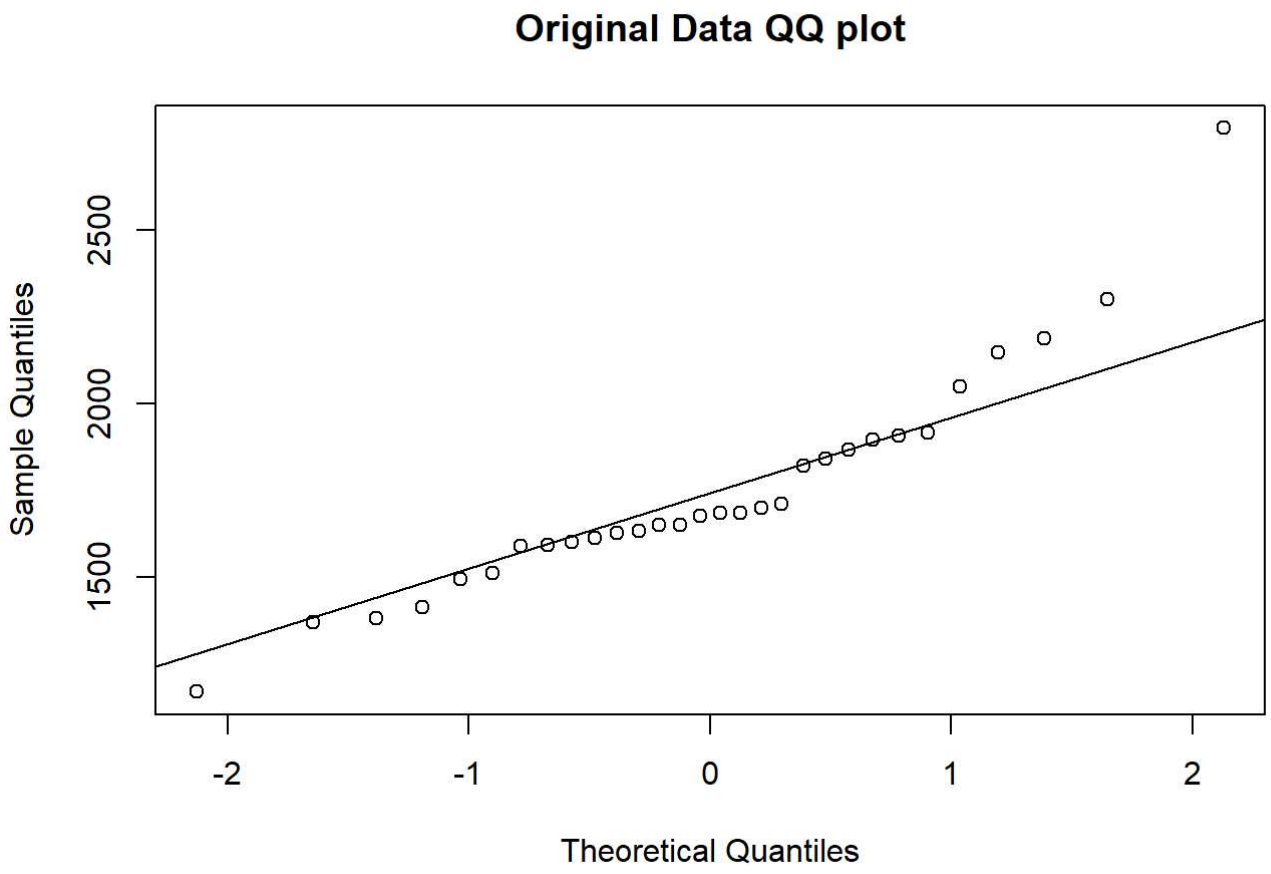

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.959864226926573 The data is not normal within si
gnificance levels of 0.1 Transforming data ... "
```



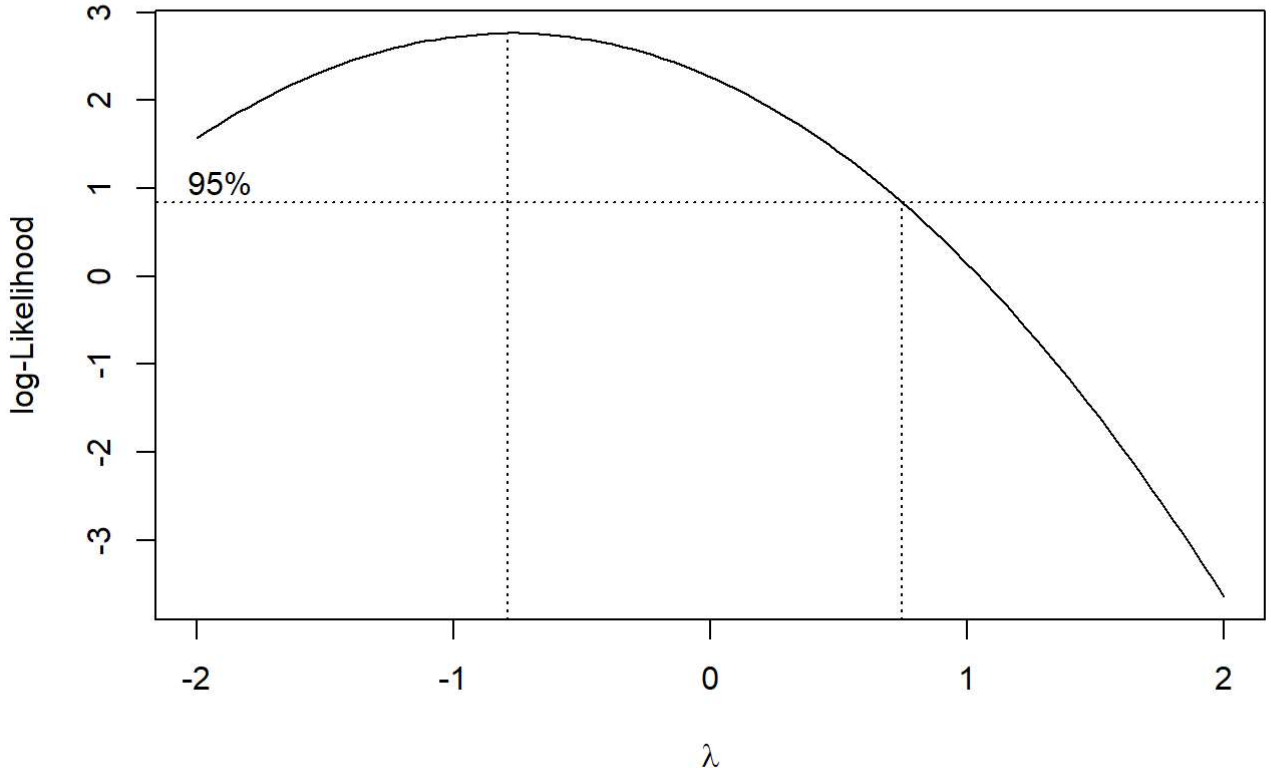

```
## [1] "The best lambda is  -0.626262626262626"
```



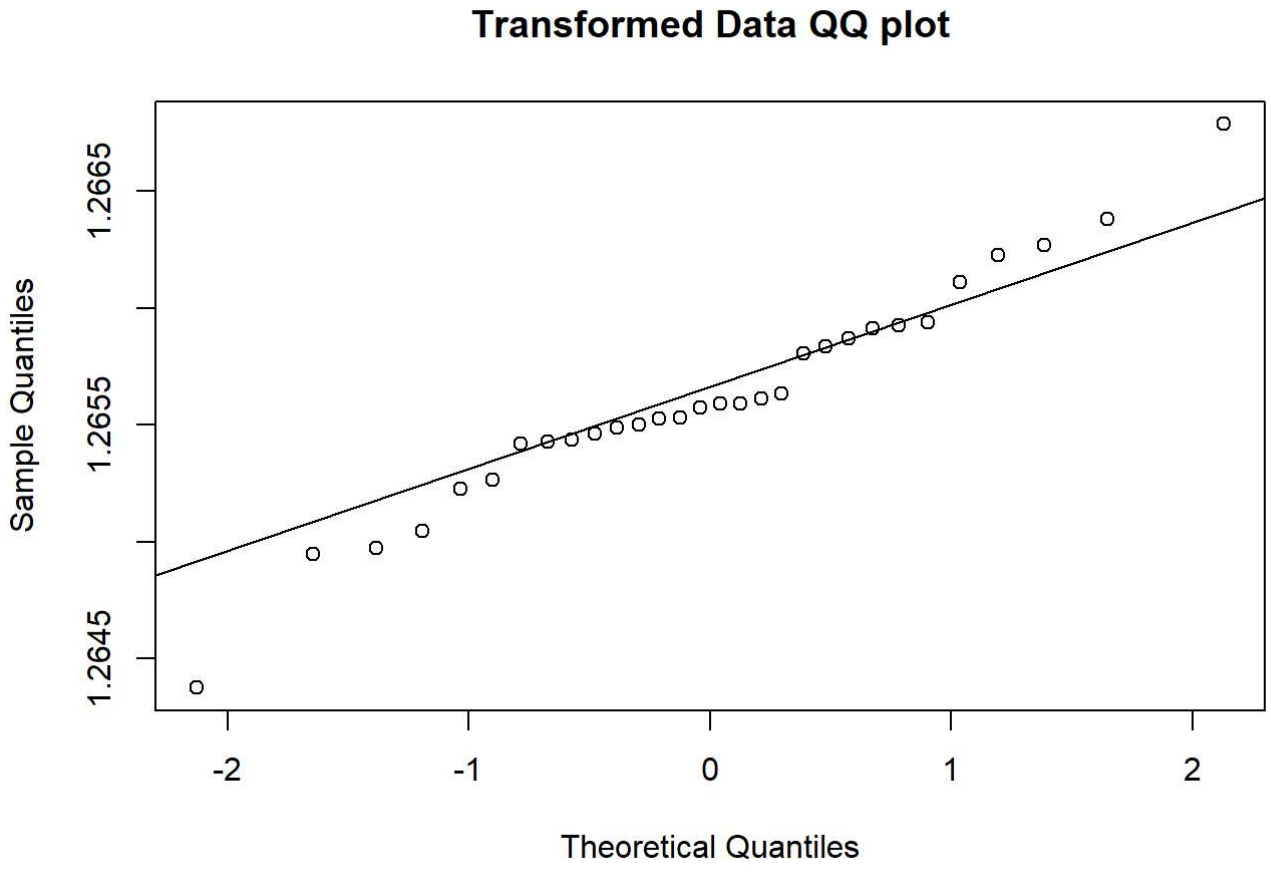
```
## [1] "With a correlation coefficient of  0.987535483360367 The data is normal within signif
icance levels of 0.1 For the box cox transformed data"
## [1] "V2"
```



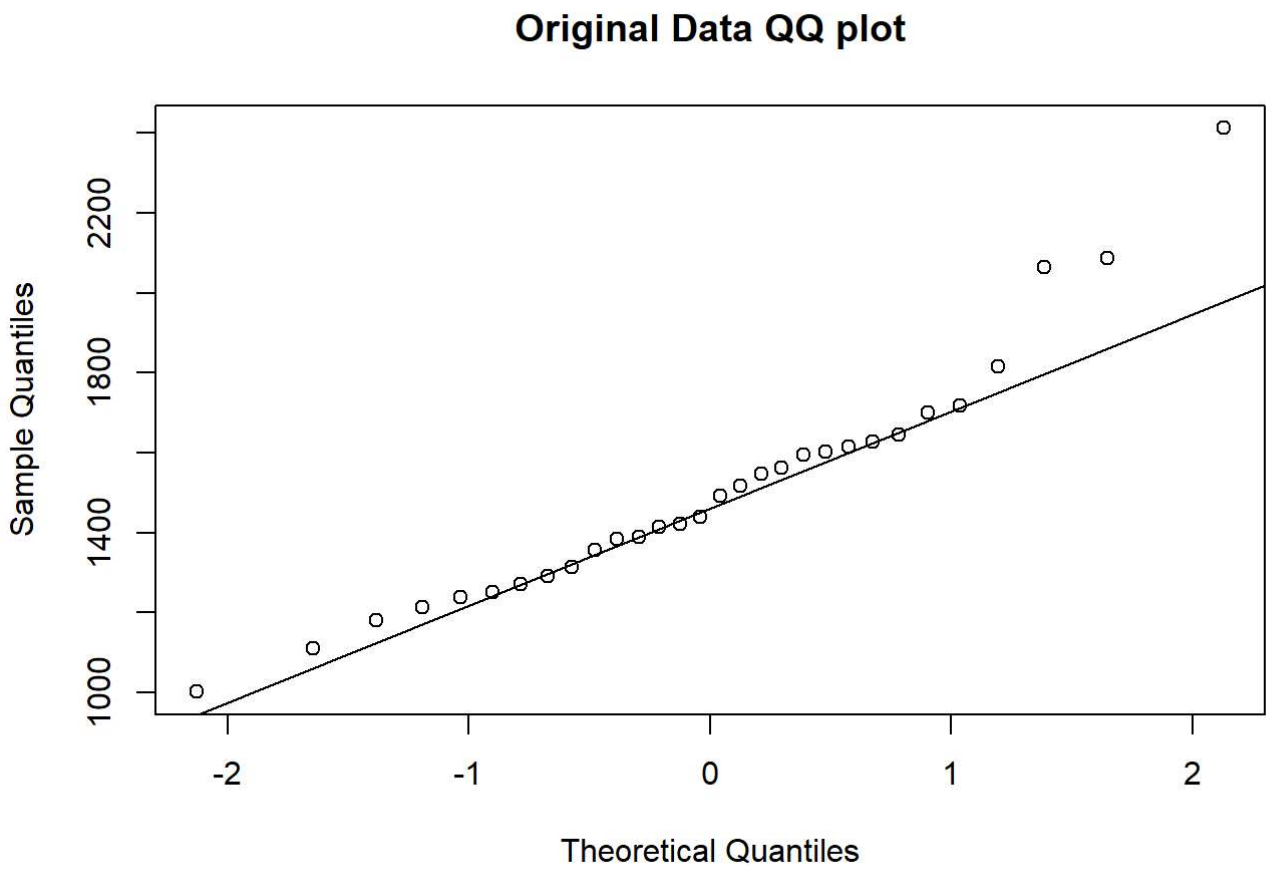

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.950390941215503 The data is not normal within si
gnificance levels of 0.1 Transforming data ... "
```



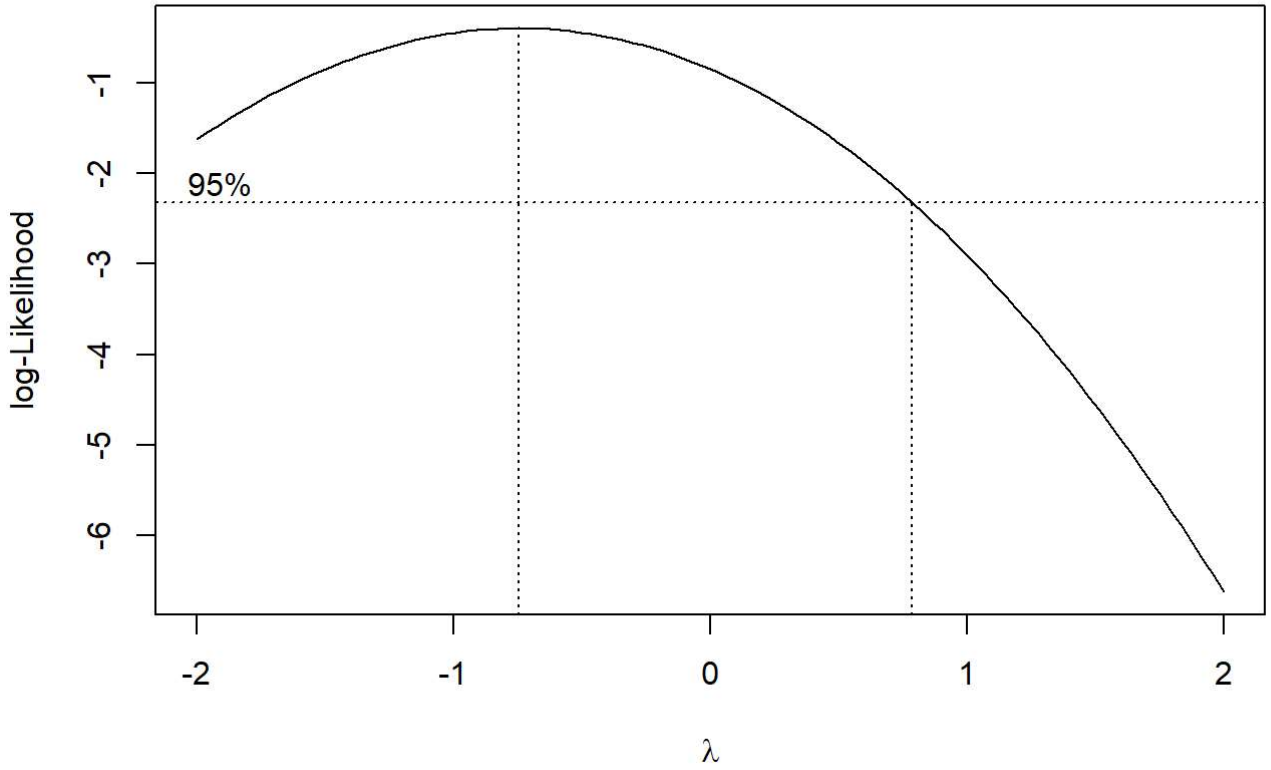

```
## [1] "The best lambda is  -0.787878787878788"
```



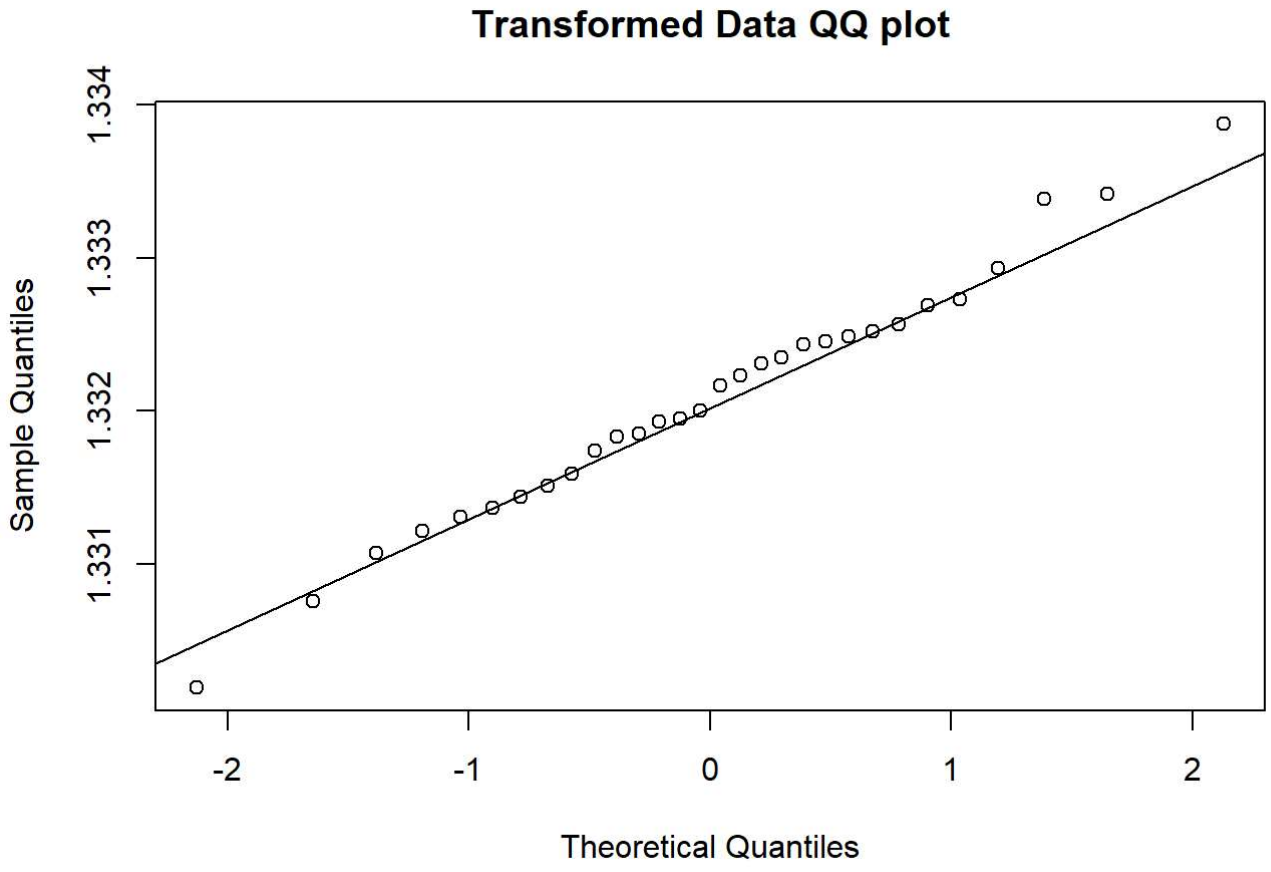
```
## [1] "With a correlation coefficient of  0.982798262680497 The data is normal within signif
icance levels of 0.1 For the box cox transformed data"
## [1] "V3"
```



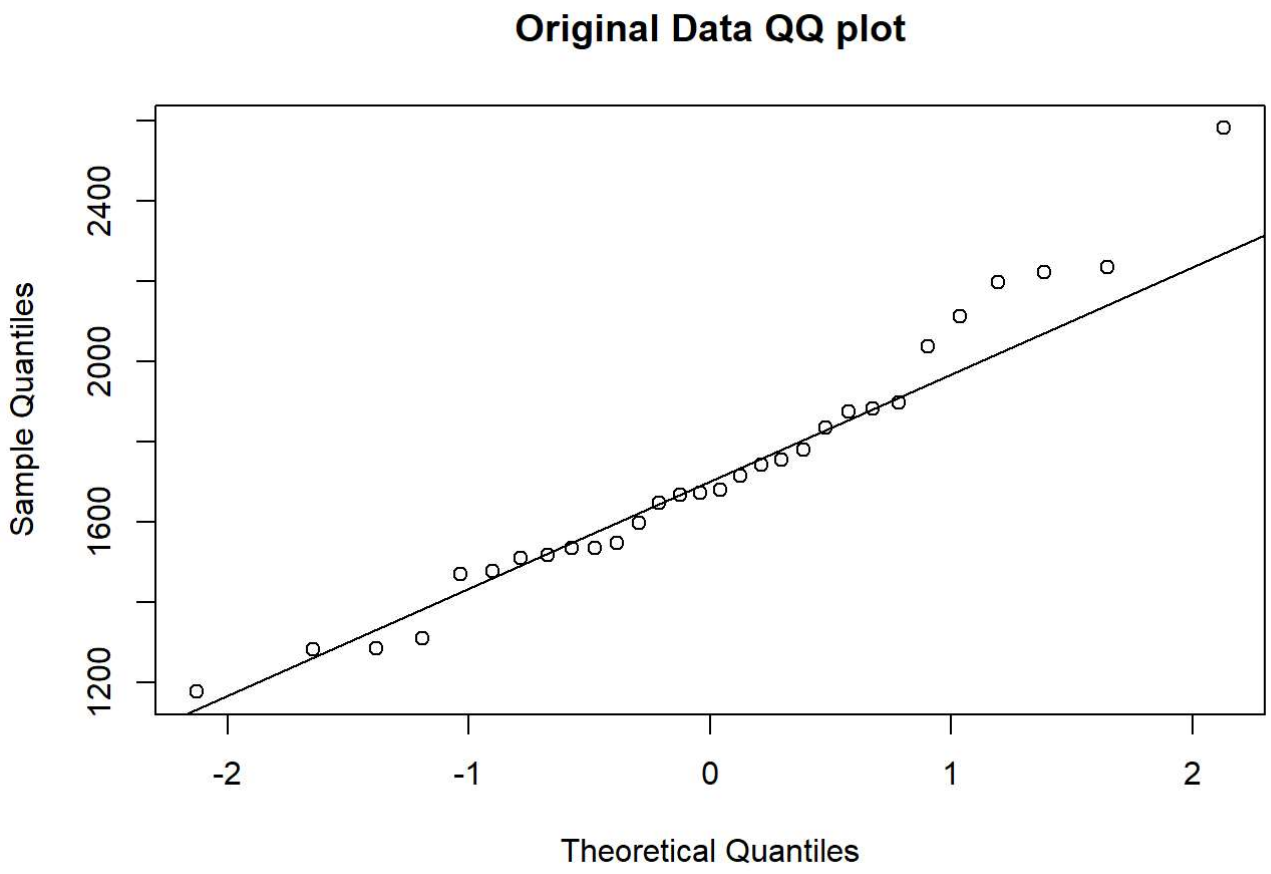

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.963410375639771 The data is not normal within si
gnificance levels of 0.1 Transforming data ... "
```



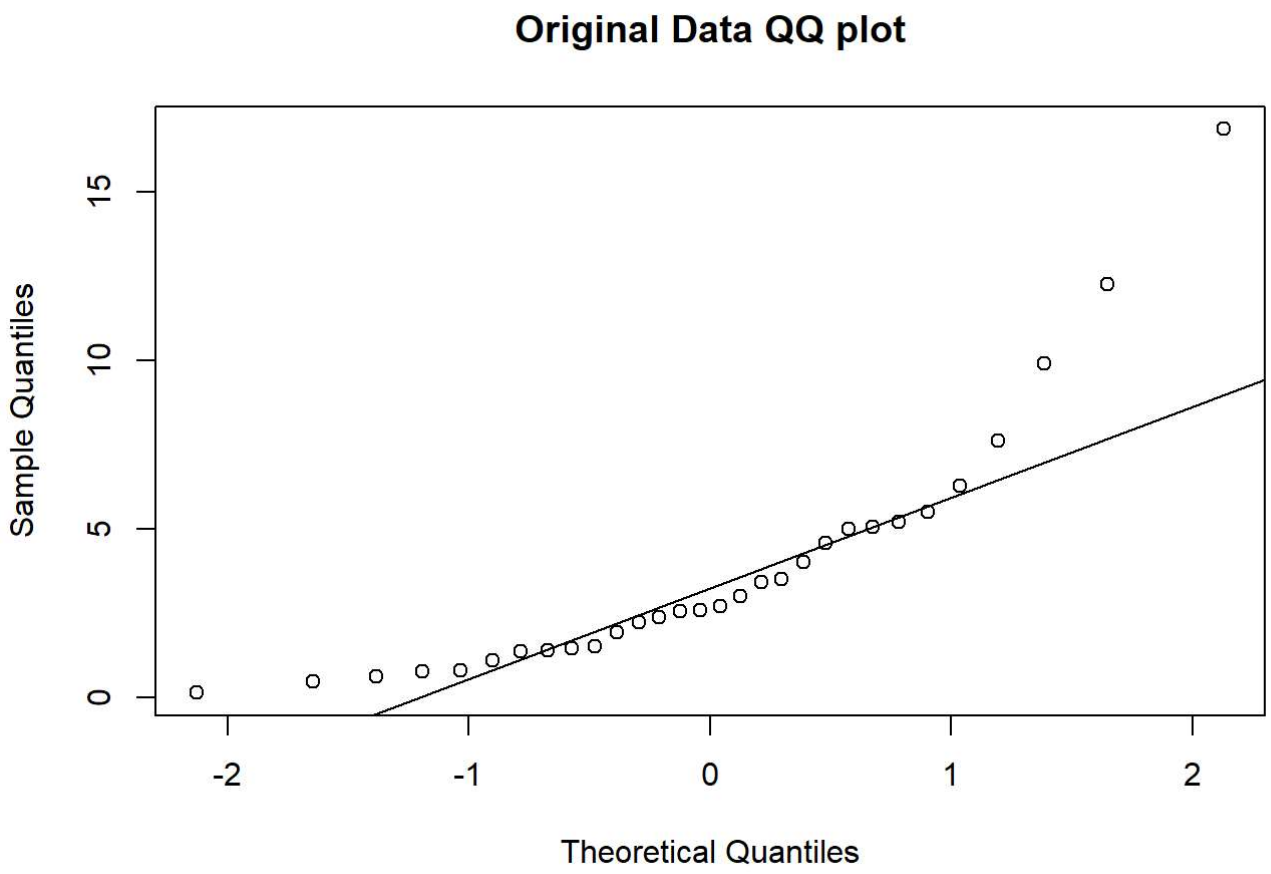

```
## [1] "The best lambda is  -0.747474747474747"
```



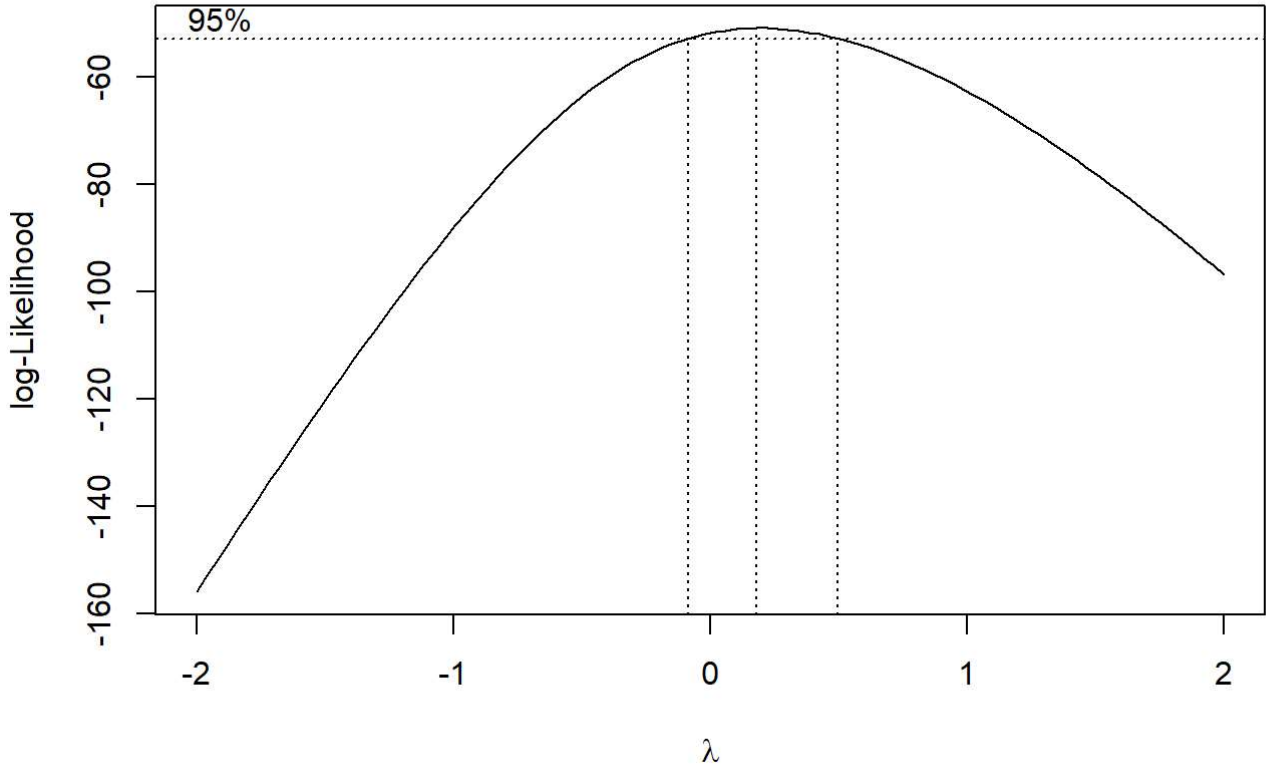
```
## [1] "With a correlation coefficient of  0.994189192711897 The data is normal within signif
icance levels of 0.1 For the box cox transformed data"
## [1] "V4"
```



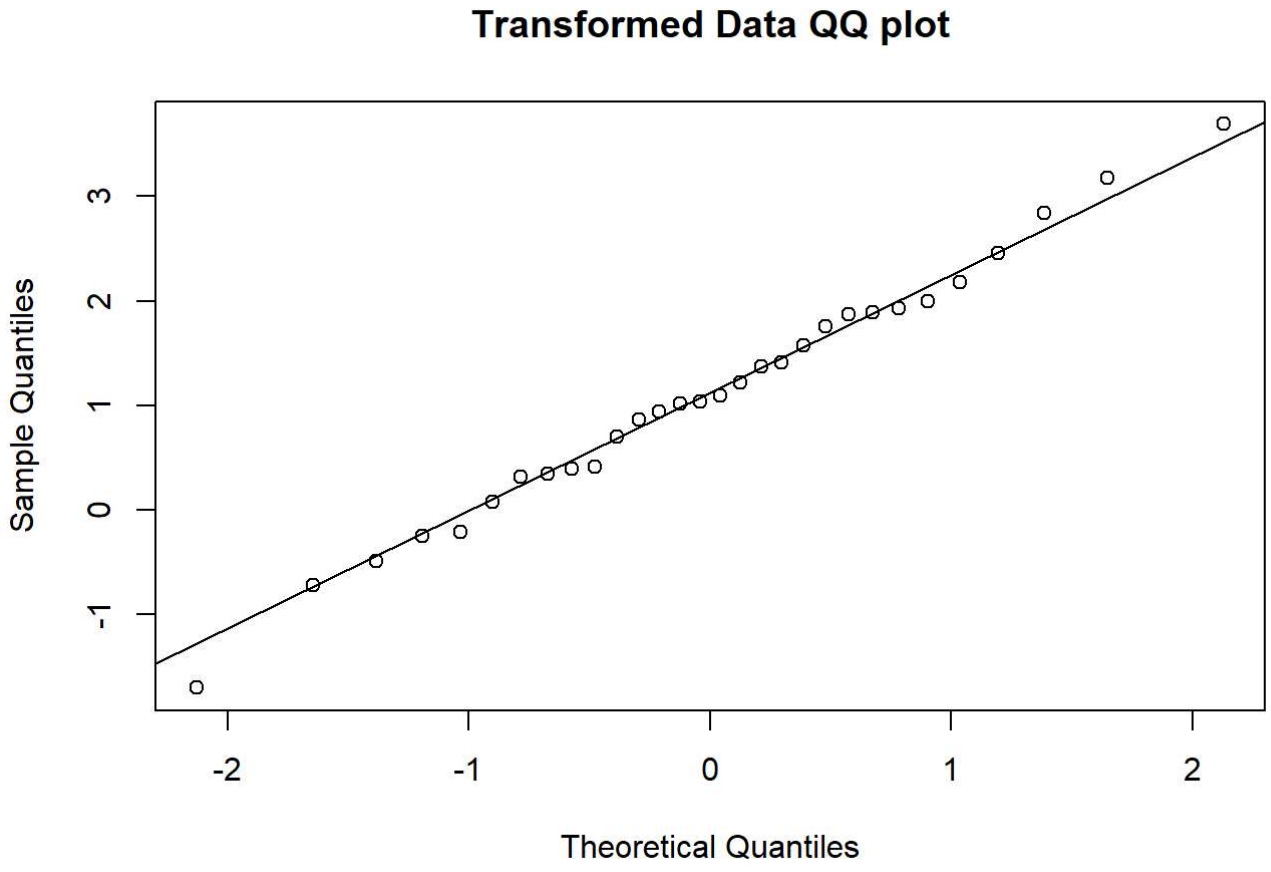

```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.980292022100608 The data is normal within signif
icance levels of 0.1"
## [1] "V5"
```




```
## [1] "Length of data is  30"
##    n    one   five    ten
## 5 25 0.9410 0.9591 0.9665
## 6 30 0.9479 0.9652 0.9715
## [1] "With a correlation coefficient of  0.891223192804262 The data is not normal within si
gnificance levels of 0.1 Transforming data ... "
```




```
## [1] "The best lambda is  0.181818181818182"
```



```
## [1] "With a correlation coefficient of  0.996412803498558 The data is normal within signif
icance levels of 0.1 For the box cox transformed data"
```

Overview of transformed data:
significance 0.01:
- x1: Normal
- x2: Normal
- x3: normal
- x4: normal
- x5: normal (when transformed)

significance: 0.05:
- x1: Normal (when transformed)
- x2: Normal (when transformed)
- x3: Normal (when transformed)
- x4: Normal
- x5: Normal (when transformed)

significance: 0.10:
- x1: Normal (when transformed)
- x2: Normal (when transformed)
- x3: Normal (when transformed)
- x4: Normal
- x5: Normal (when transformed)

Check normality for each pair of the two attributes

To do this we first have a function which simulates a chi squared n points on a distribution n_simulations times (often 10000), and compares it with the theoretical distribution and then calculates the correlation coeeficiant. It then Takes the highest correlation coefficient and returns it. This will then be the critical value that our correlation coeefficient needs to be higher than in order to follow a chi-sqauared distribution, which indicates that we have a bivariate normal distribution.

```
FindcrikChi <- function(n=30, p=2, alpha= 0.05, n_simulations=10000){

    cricvec <- rep(0, n_simulations)  #vector for the rQ result collection#

    for(i in 1:n_simulations){
        #iteration to estimate rQ#
        numvec <- rchisq(n, p)  #generate a data set of size n, degree of freedom=p#
        d <- sort(numvec)
        q <- qchisq((1:n-0.5)/n, p)
        cricvec[i] <- cor(d,q)
    }

    scricvec <- sort(cricvec)
    cN <- ceiling(n_simulations* alpha) #to be on the safe side I use ceiling instead of floo
r(), take the 'worst' alpha*N cor as rQ, everything lower than that is deemed as rejection#
    cricvalue <- scricvec[cN]
    result <- list(cN, cricvalue, scricvec)
    return(result[[2]])
}
```

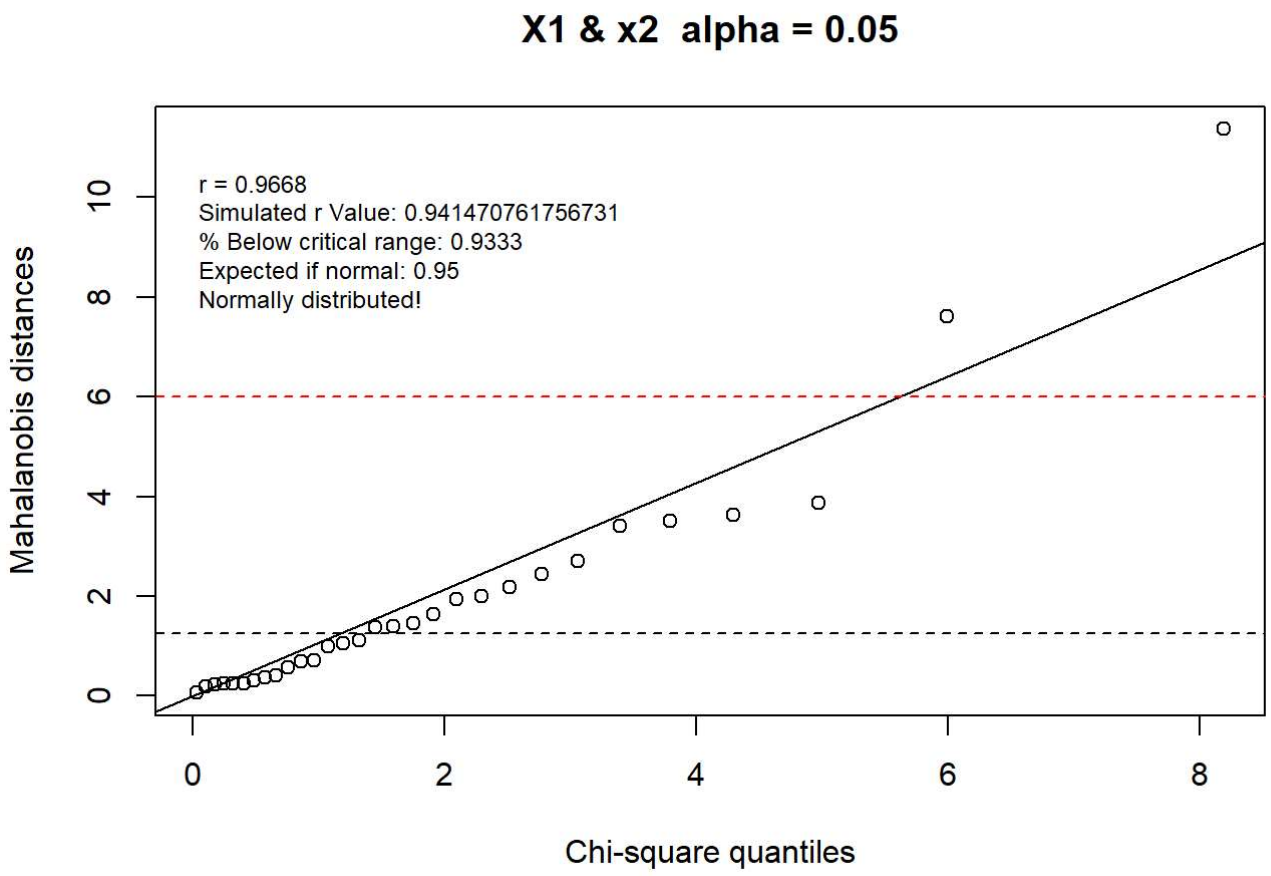
This function takes two variables, the simulated correlation coefficient from preivouslky and a aplha/significance value and creates a qqplot. It then calculates mahalanobis distance for the data points and the correlation coefficient between the mahalanobis distances and a theoretical chi squared distribution. It then compares this correlation coefficient with the simulated coefficient and checks if it is higher. If it is higher, this data follows a chi-squared distribution, which indicates, that it is normally distributed. It also calculate how large a percent of the points that are within in alpha, that is how many percent of the points are within the 1 - alpha contours, and shows what would be expected. If it is not crazy far from the expected, the correlation coeefficient is a usefull measure.

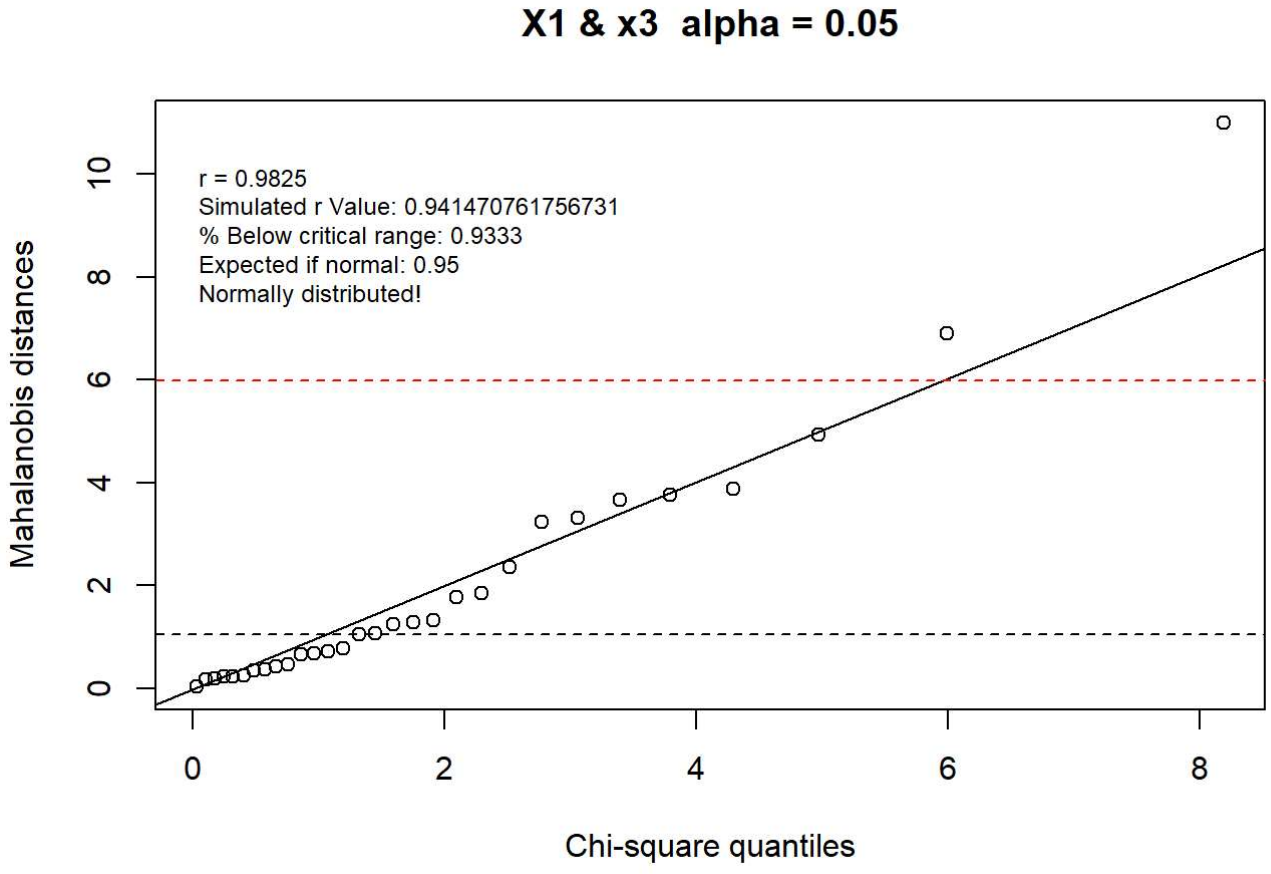
```
multi_var_norm <- function(df, sim_cor, alpha, name, remove_outlier = FALSE, n_outliers = 1)
{

  # Data and parameters
  n <- nrow(df) # observations
  p <- ncol(df) # number of variables
  D2 <- mahalanobis(df,
                    center  = colMeans(df),
                    cov = cov(df)) # generalized squared distance

  # Removes outliers if necessary
  if(remove_outlier == TRUE){

    i = 0
    while (i < n_outliers){
    #This is where we remove outliers. This will most likely change the correlation value and
the % number of points in the contour.
    D2 <- D2[-which.max(D2)]
    i = i + 1
    }
  }


  # Chi square qq plot
  chi_plot <- qqplot(qchisq(ppoints(n, a = .5), df = p), D2,
                     plot.it = F) # chi square plot values.

  my_cor <- cor(chi_plot$x, chi_plot$y) # correlation value
  critical_value <- qchisq(p = alpha,
                           df = p,
                           lower.tail = F) # calculate critical value

  # Proportion of points below alpha value
  prop_within_contour <- round(length(D2[D2 <= critical_value]) / length(D2),4) #


  quantiles <- quantile(D2)
  quantile_25 <- quantiles[2]
  quantile_50 <- quantiles[3]
  quantile_75 <- quantiles[4]

  plot(chi_plot, #From here and downwards it is only how you want the plot to look.
       ylab = 'Mahalanobis distances',
       xlab = 'Chi-square quantiles',
       main = paste0(name, ' alpha = ',alpha)) # plot chi square plot

  # Q line
  y <- rchisq(500, df = p)
  qqline(y, distribution = function(n) qchisq(n, df = p), prob = c(0.1, 0.6))

  # Lines for quantiles
  abline(h = quantile_50, lty = 2) # 50% quantiles becayse book p. 187
  abline(h = critical_value, lty = 2, col = "red") # Below Critical value

  if(my_cor > sim_cor){
```

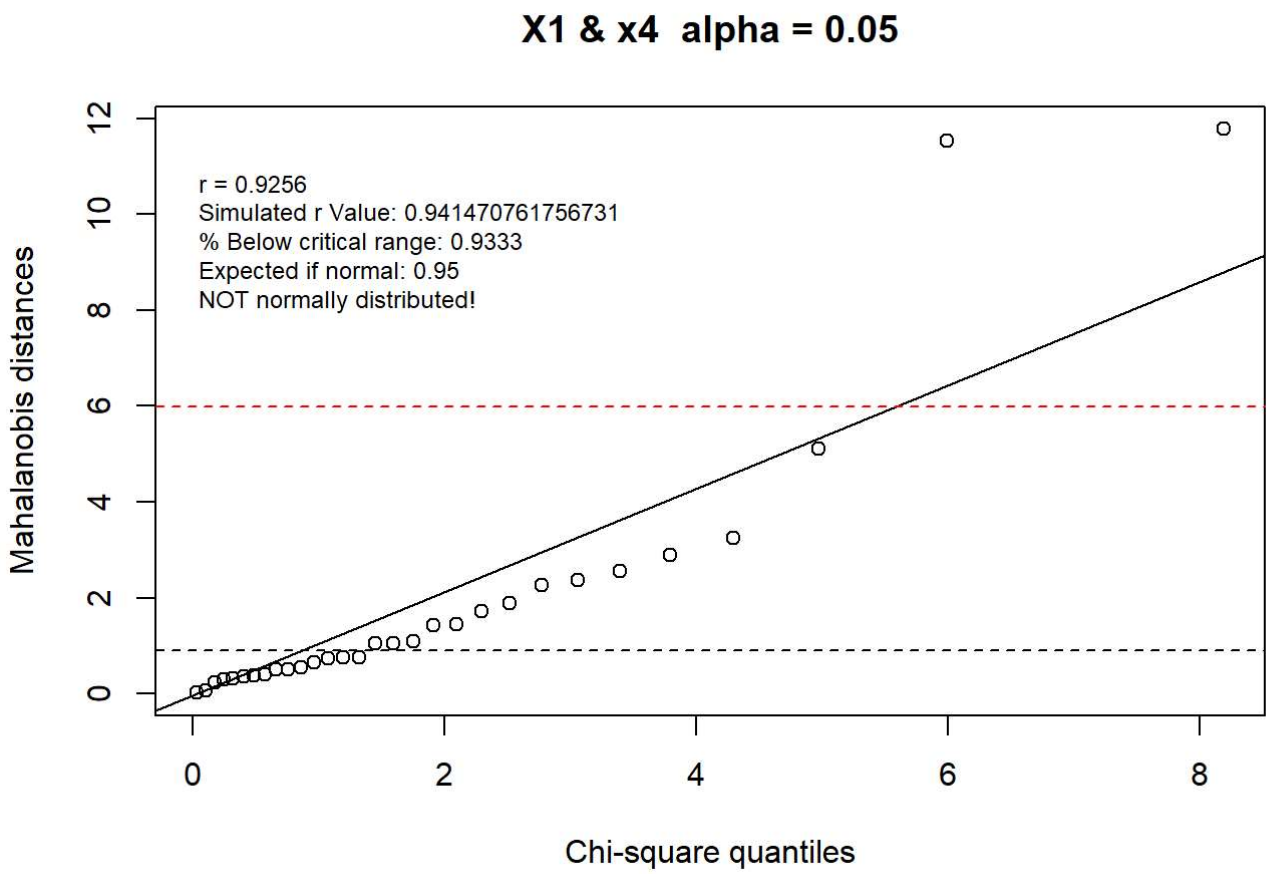
```
  conclusion <- "Normally distributed!"
}

else {
  conclusion <- "NOT normally distributed!"
}
legend("topleft",
       paste0("r = ", round(my_cor,4), "\n",
              "Simulated r Value: ", sim_cor, "\n",
              "% Below critical range: ", prop_within_contour, "\n",
              "Expected if normal: ", 1-alpha, "\n",
              conclusion),
       cex = 0.75,
       bty = "n") # add legend to plot
}
```

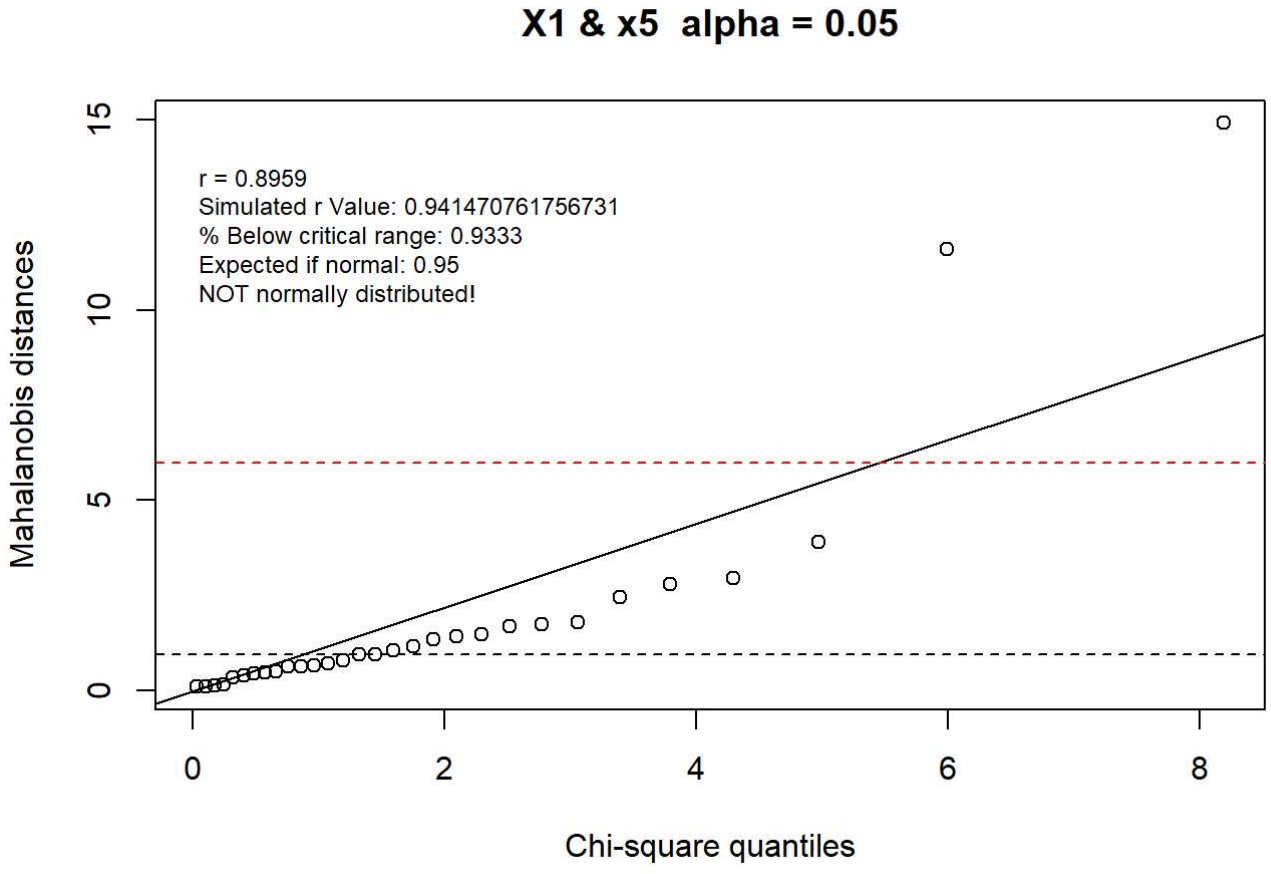
We then create a function which test each pair of variables for normality

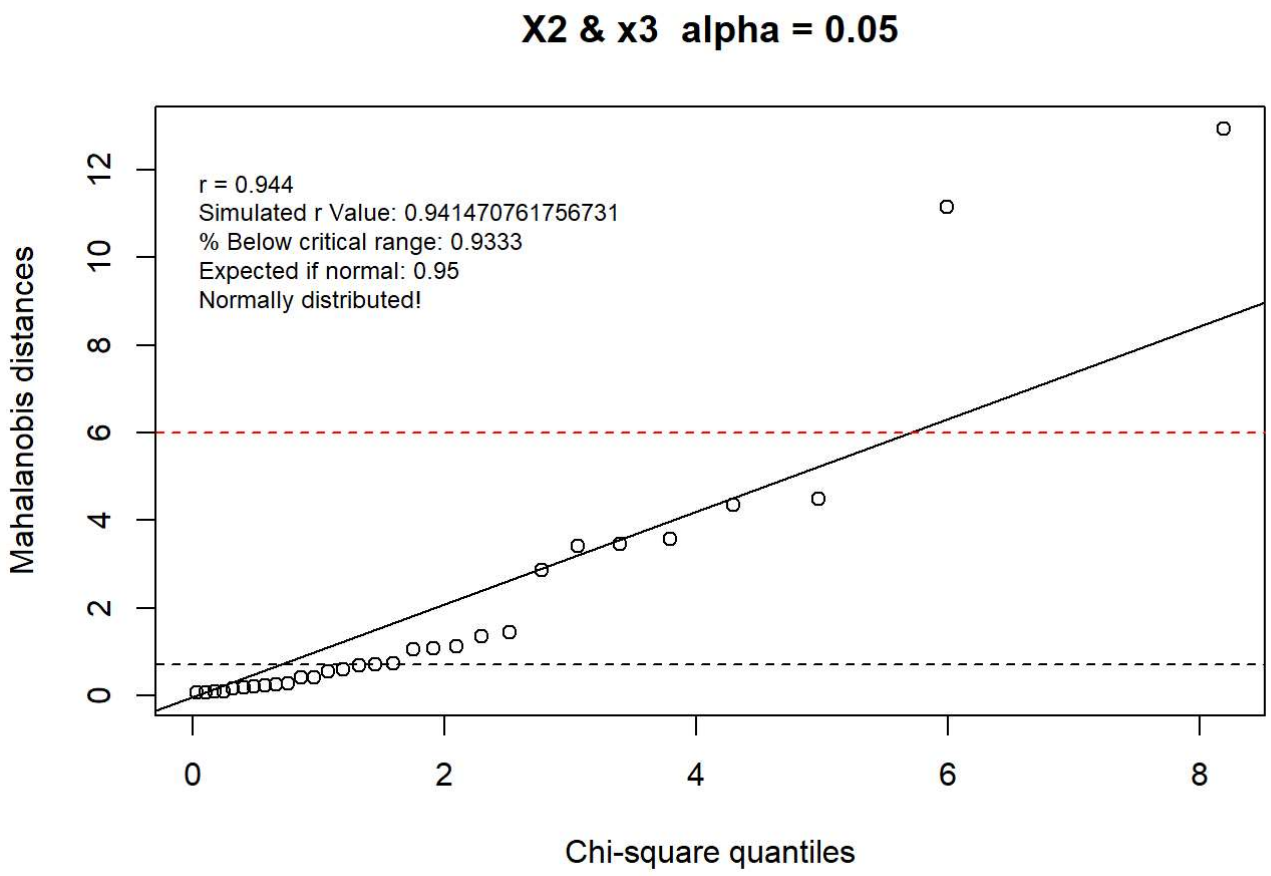
```
bivar_pairs <- function(data_frame, significance = 0.05, remove_outliers = FALSE, n_outliers
= 1, n_simulations = 10000) {

sim_cor <- FindcrikChi(n = length(data_frame[,1]),
                       p = 2,
                       alpha = significance,
                       n_simulations)

prev <- c()
i = 1
for (col_name in colnames(data_frame)){

  j = 1
  for (col_name_inner in colnames(data_frame)){

    if (i == j){
      j = j + 1
      next
    }

    else if (j %in% prev){
      j = j + 1
      next
    }

    else{
      multi_var_norm(df = data_frame[, c(i, j)],
                 sim_cor = sim_cor,
                 alpha = significance,
                 name = paste0("X", i, " & ", "x", j, " "),
                 remove_outlier = remove_outliers,
                 n_outliers = n_outliers)

    }
    j = j + 1
  }
  prev <- append(prev, i)
  i = i + 1
  }
}

bivar_pairs(df, significance = 0.05)
```

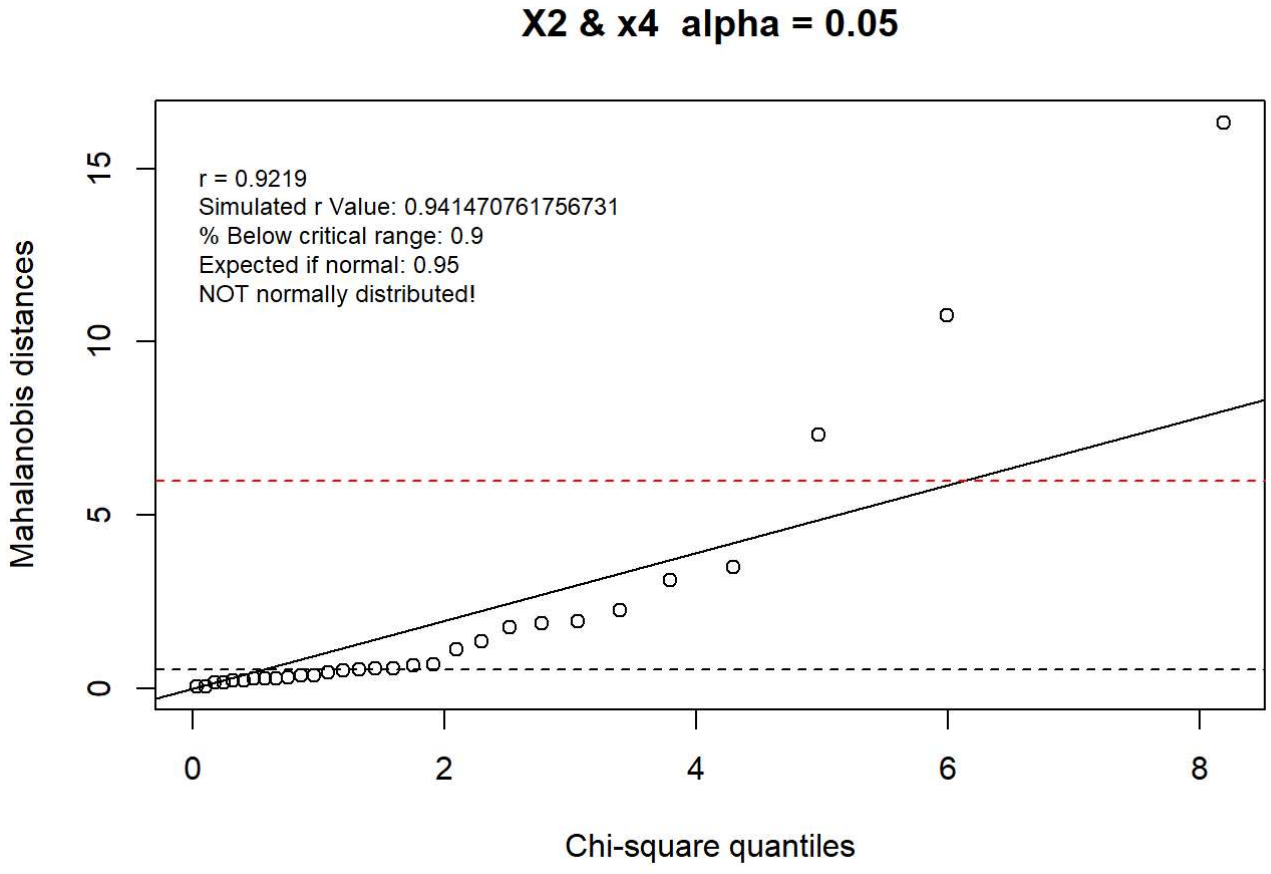

X1 & x2 alpha = 0.05
r = 0.9668
Simulated r Value: 0.941470761756731
% Below critical range: 0.9333
Expected if normal: 0.95
Normally distributed!
Mahalanobis distances
Chi-square quantiles
0
2
4
6
8
0
2
4
6
8
10

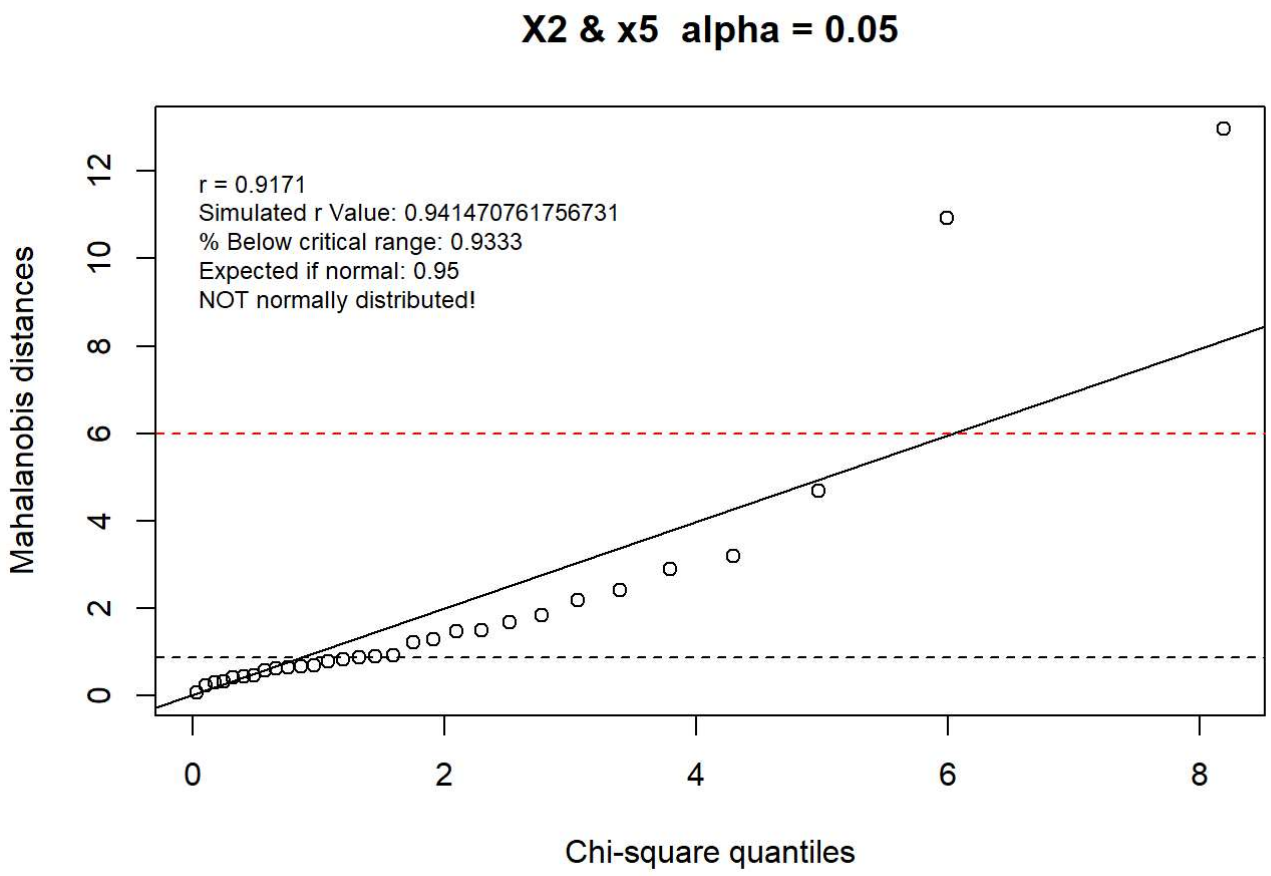

X1 & x3 alpha = 0.05
r = 0.9825
Simulated r Value: 0.941470761756731
% Below critical range: 0.9333
Expected if normal: 0.95
Normally distributed!
Mahalanobis distances
Chi-square quantiles
0
2
4
6
8
0
2
4
6
8
10

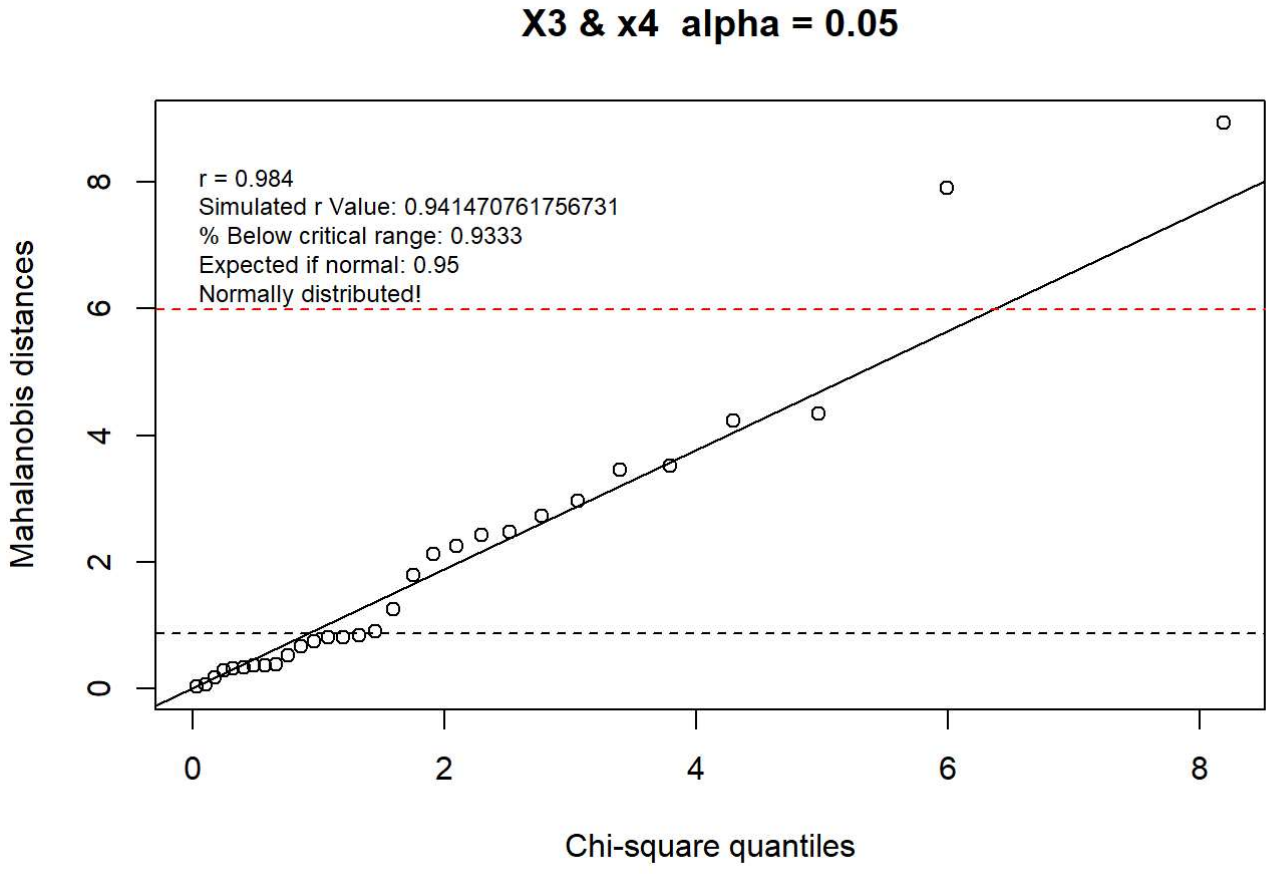

X1 & x4 alpha = 0.05
r = 0.9256
Simulated r Value: 0.941470761756731
% Below critical range: 0.9333
Expected if normal: 0.95
NOT normally distributed!
Mahalanobis distances
Chi-square quantiles
0
2
4
6
8
10
12

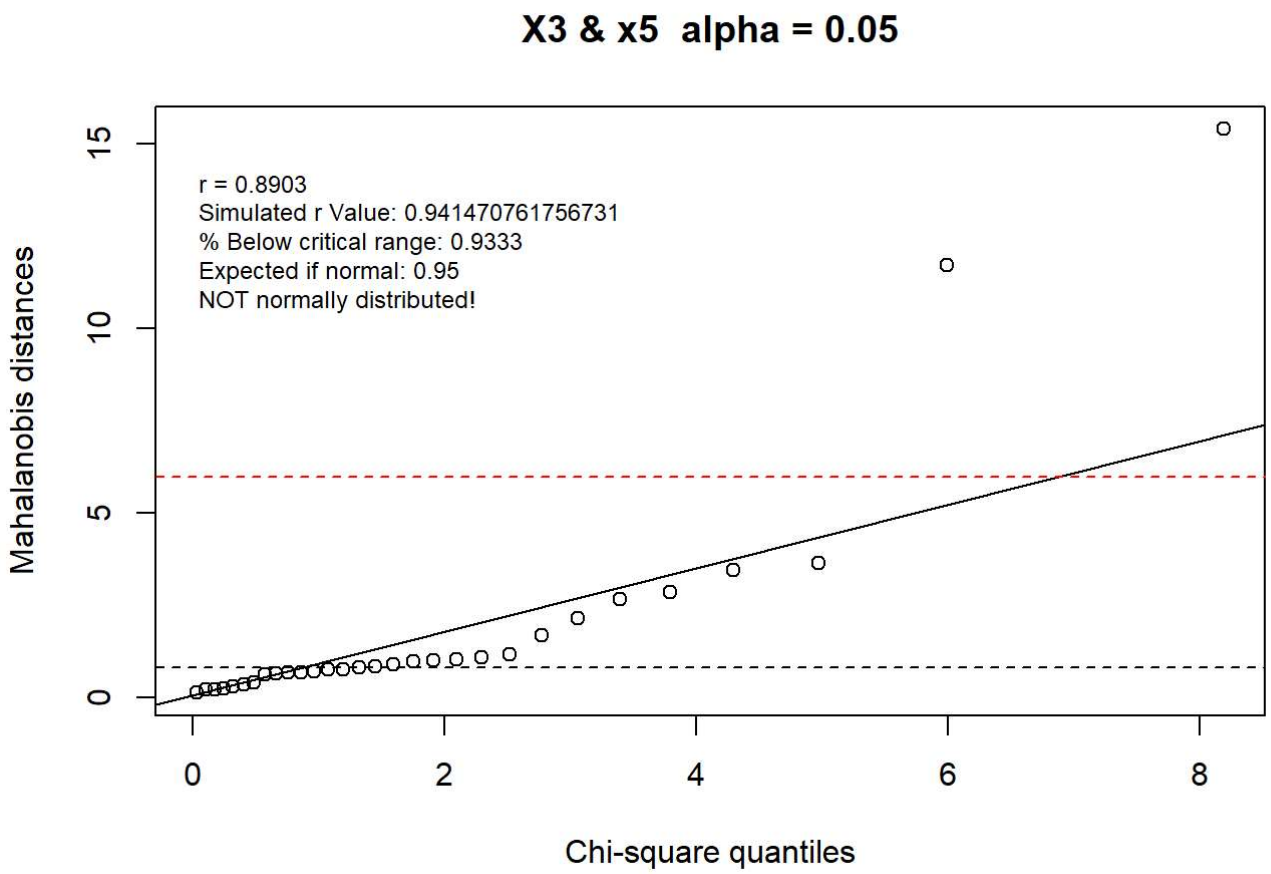

X1 & x5 alpha = 0.05
r = 0.8959
Simulated r Value: 0.941470761756731
% Below critical range: 0.9333
Expected if normal: 0.95
NOT normally distributed!
Mahalanobis distances
Chi-square quantiles
0
2
4
6
8
5
10
15

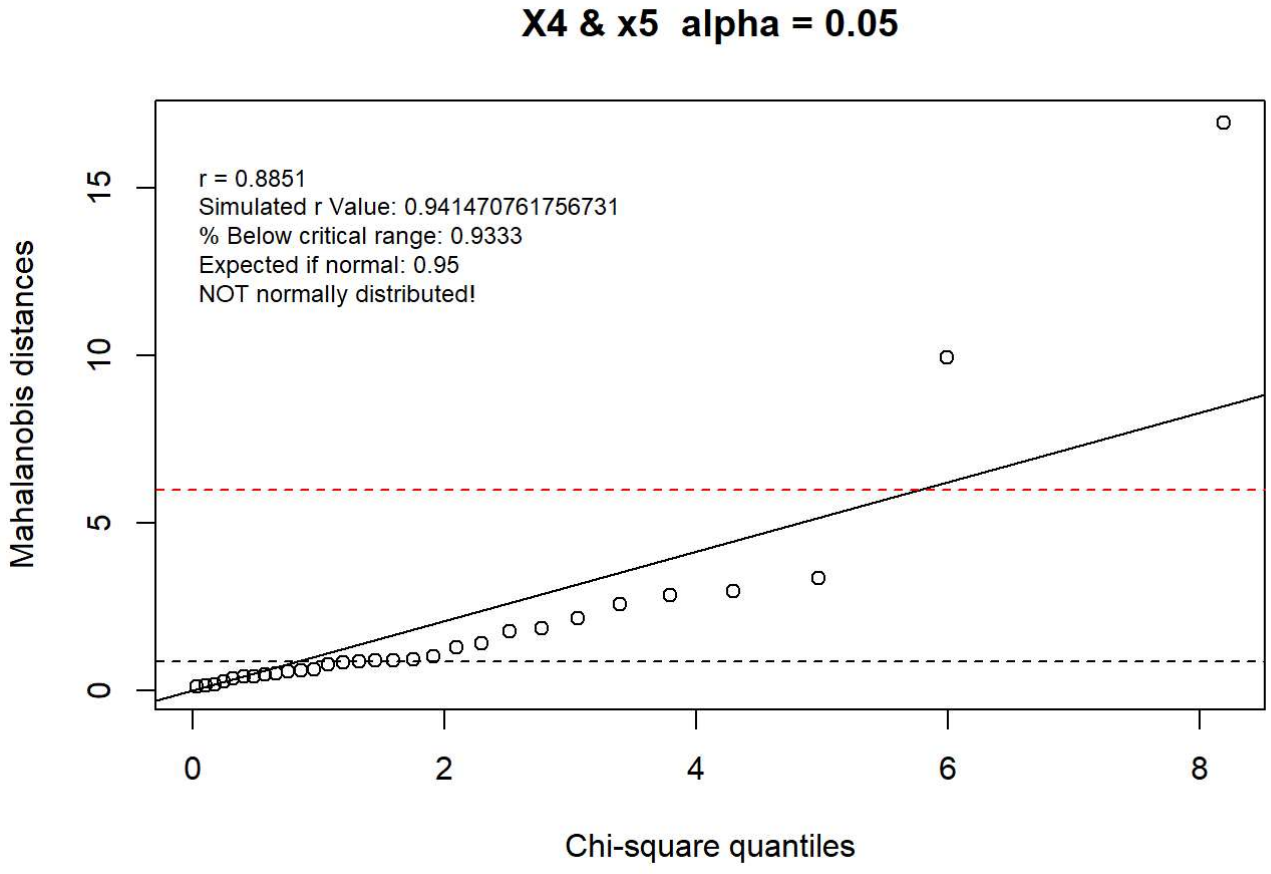

X2 & x3 alpha = 0.05
r = 0.944
Simulated r Value: 0.941470761756731
% Below critical range: 0.9333
Expected if normal: 0.95
Normally distributed!
Mahalanobis distances
Chi-square quantiles
0
2
4
6
8
0
2
4
6
8
10
12


X2 & x4 alpha = 0.05
r = 0.9219
Simulated r Value: 0.941470761756731
% Below critical range: 0.9
Expected if normal: 0.95
NOT normally distributed!
Mahalanobis distances
Chi-square quantiles
0
2
4
6
8
0
5
10
15


X2 & x5 alpha = 0.05
r = 0.9171
Simulated r Value: 0.941470761756731
% Below critical range: 0.9333
Expected if normal: 0.95
NOT normally distributed!
Mahalanobis distances
Chi-square quantiles
0
2
4
6
8
10
12


X3 & x4 alpha = 0.05
r = 0.984
Simulated r Value: 0.941470761756731
% Below critical range: 0.9333
Expected if normal: 0.95
Normally distributed!
Mahalanobis distances
Chi-square quantiles
0
2
4
6
8





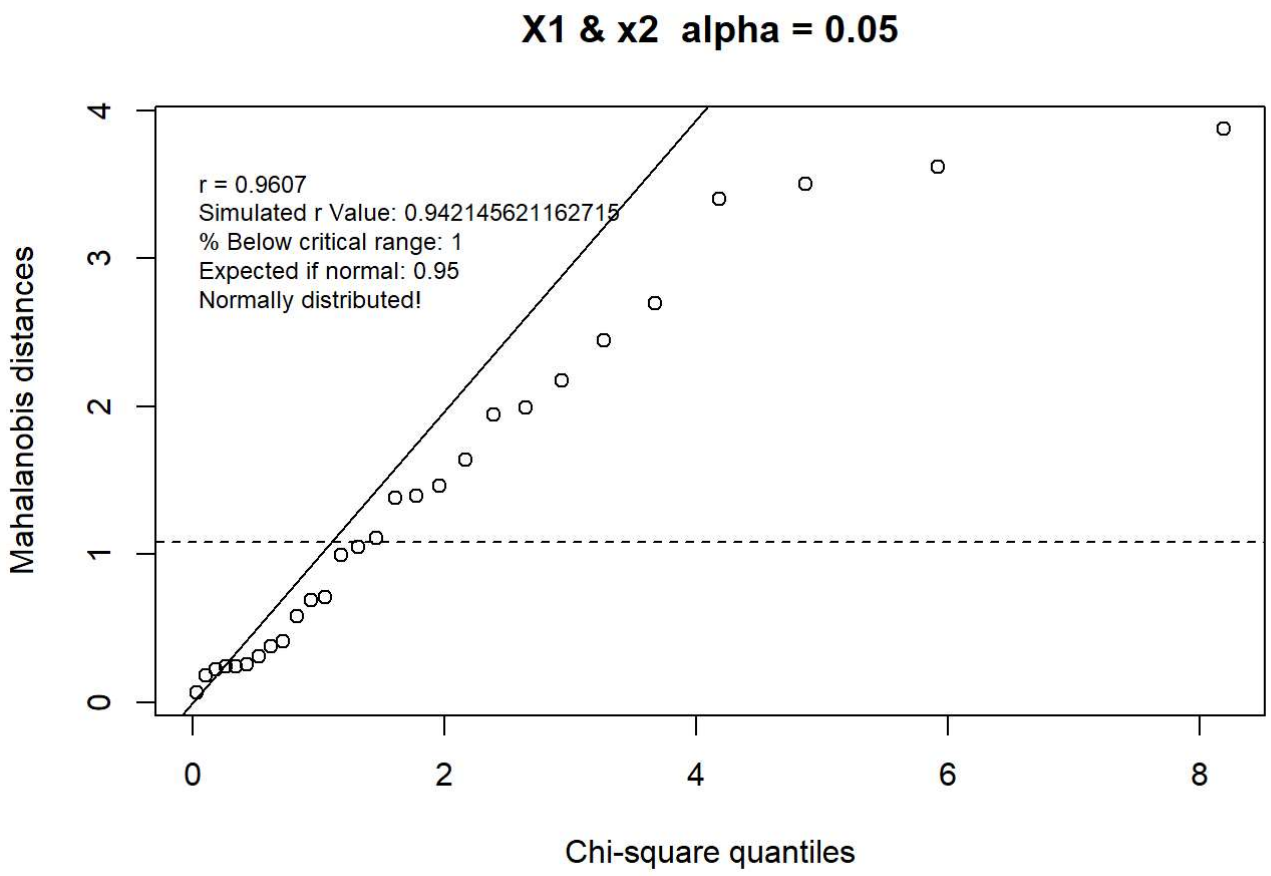
This gives the following result:

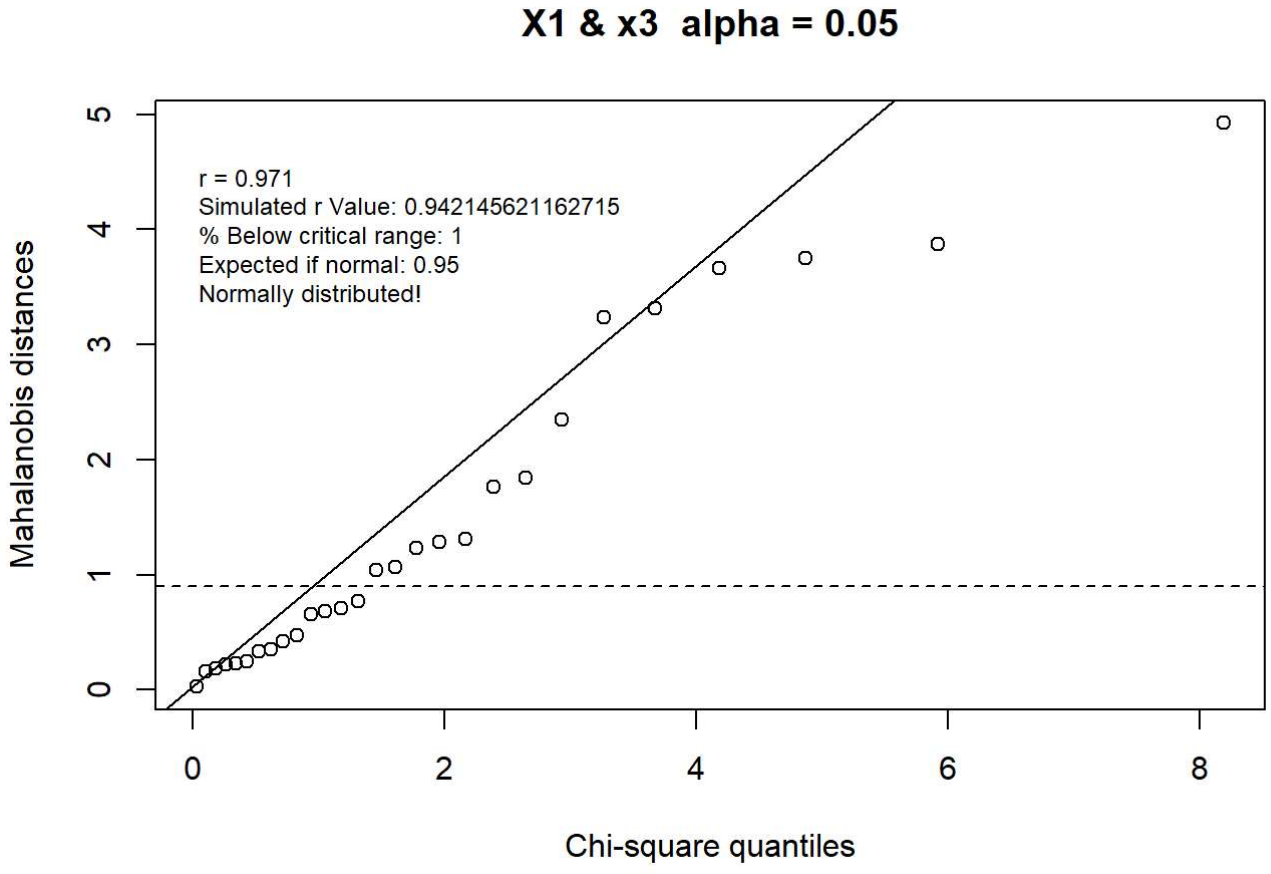
- X1 and X2: Normal
- x1 and x3: Normal
- X1 and x4: Not normal

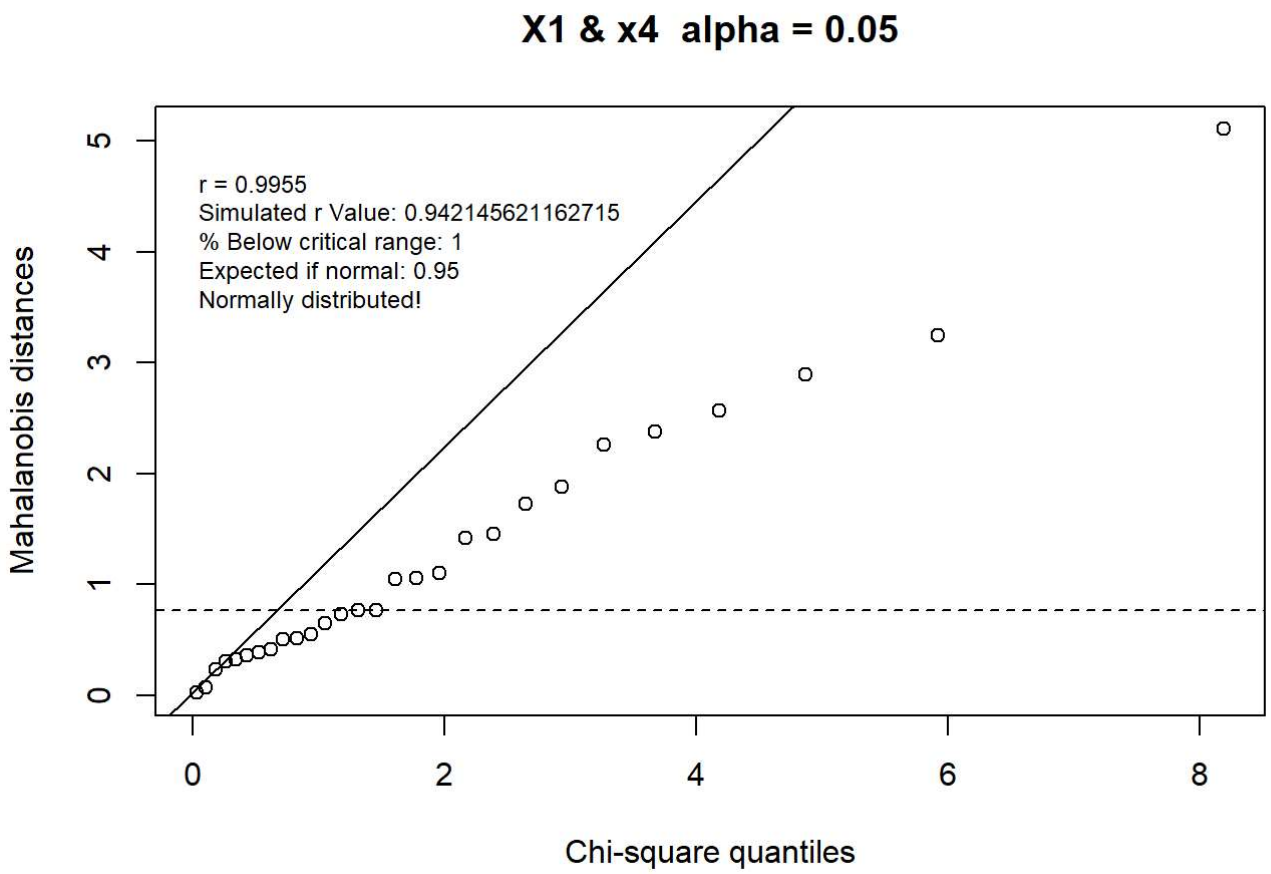
- x1 and x5: Not normal - x2 and x3: Normal
- x2 and x4: Not normal
- x2 and x5: Not normal
- x3 and x4: Normal distributed
- x3 and x5: Not normal distributed
- x4 and x5: Not normal distributed

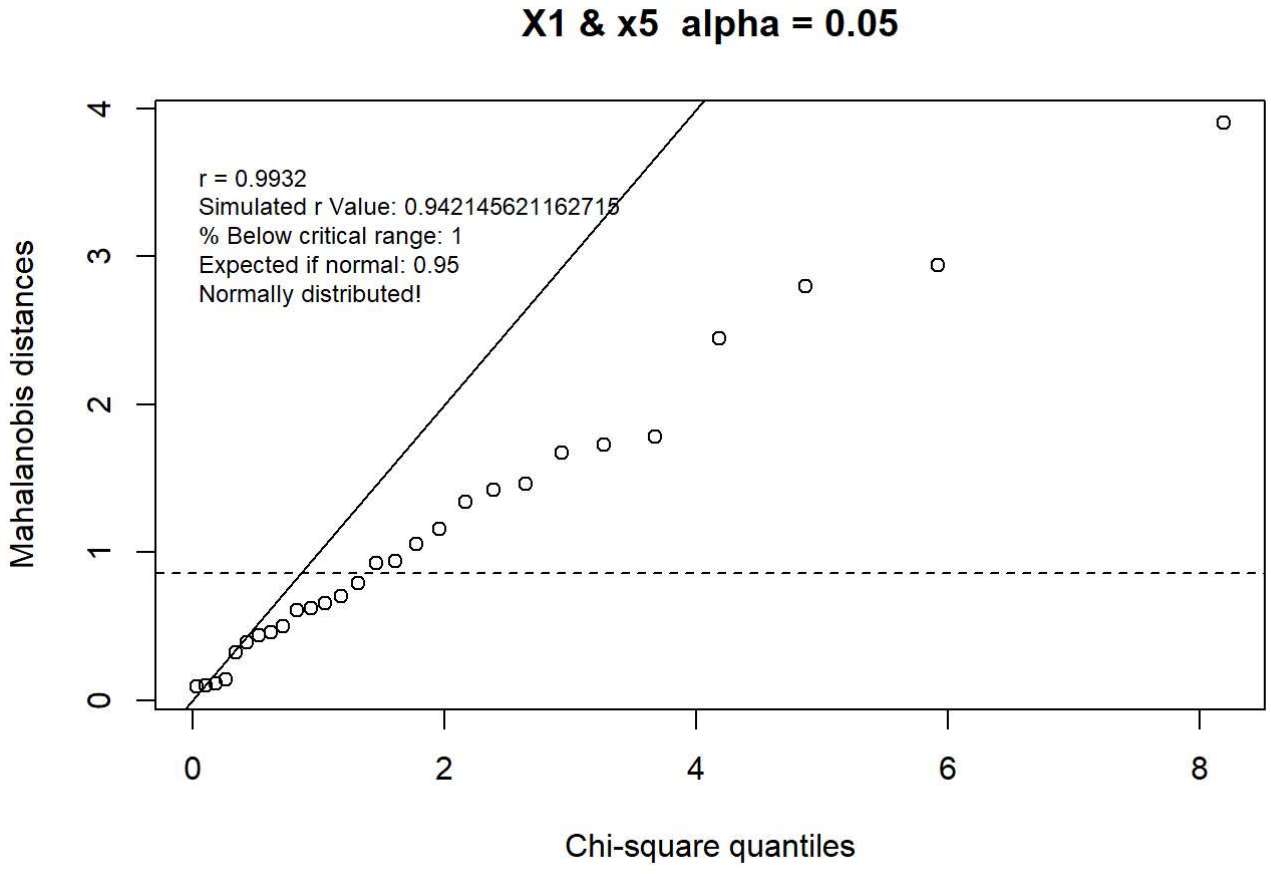
Let us try, by removing outliers and see how this changes it:

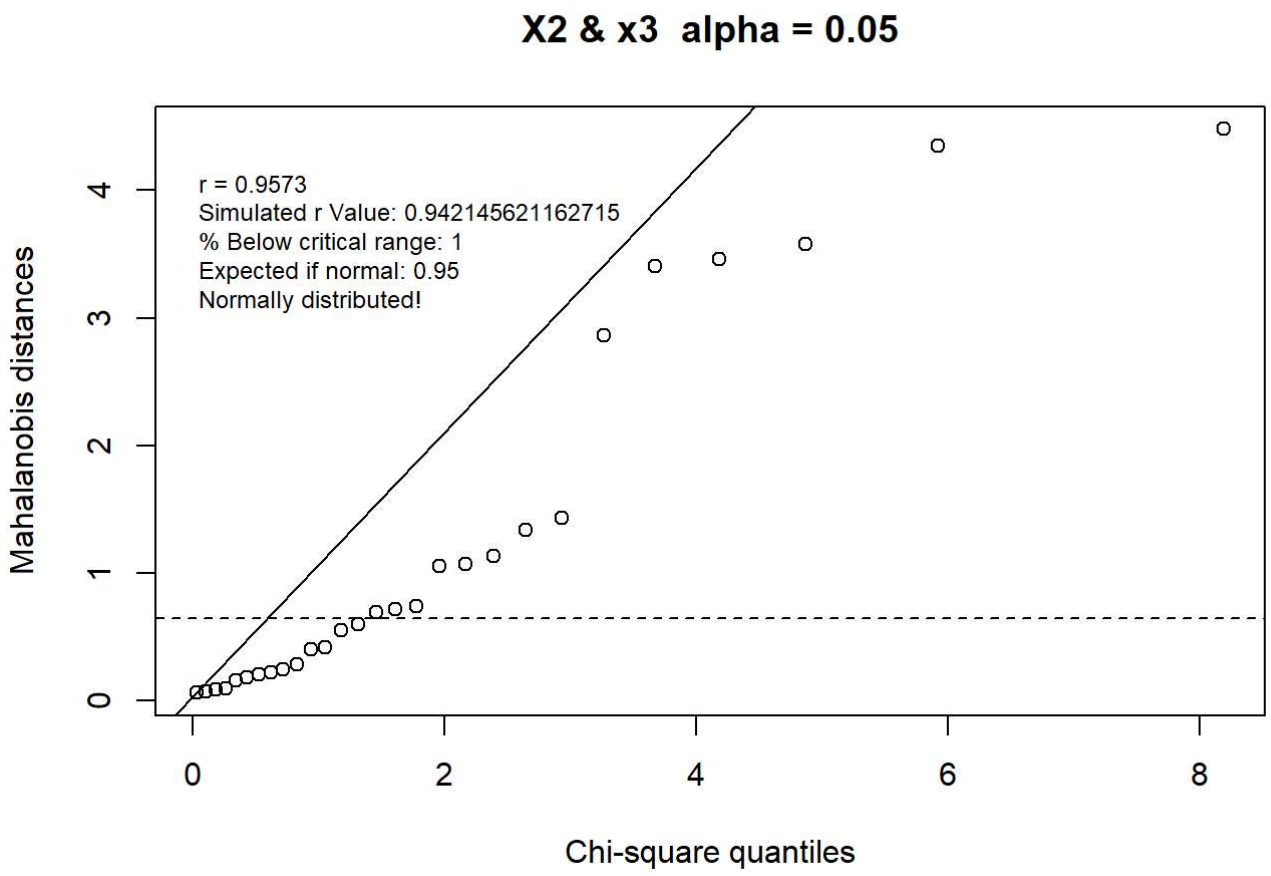
```
bivar_pairs(df, significance = 0.05, remove_outliers = TRUE, n_outliers = 2)
```

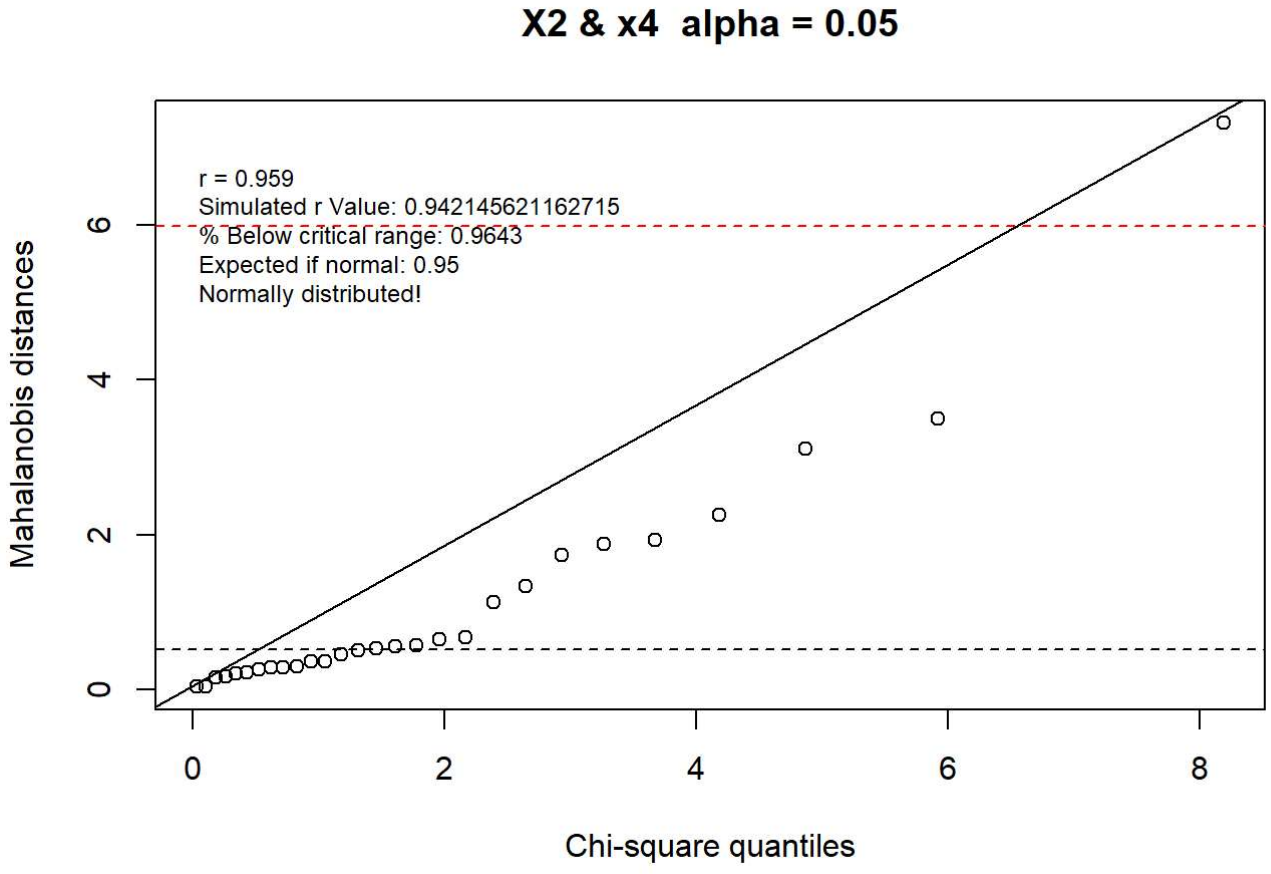

X1 & x2 alpha = 0.05
r = 0.9607
Simulated r Value: 0.942145621162715
% Below critical range: 1
Expected if normal: 0.95
Normally distributed!
Mahalanobis distances
Chi-square quantiles
0
1
2
3
4
0
2
4
6
8

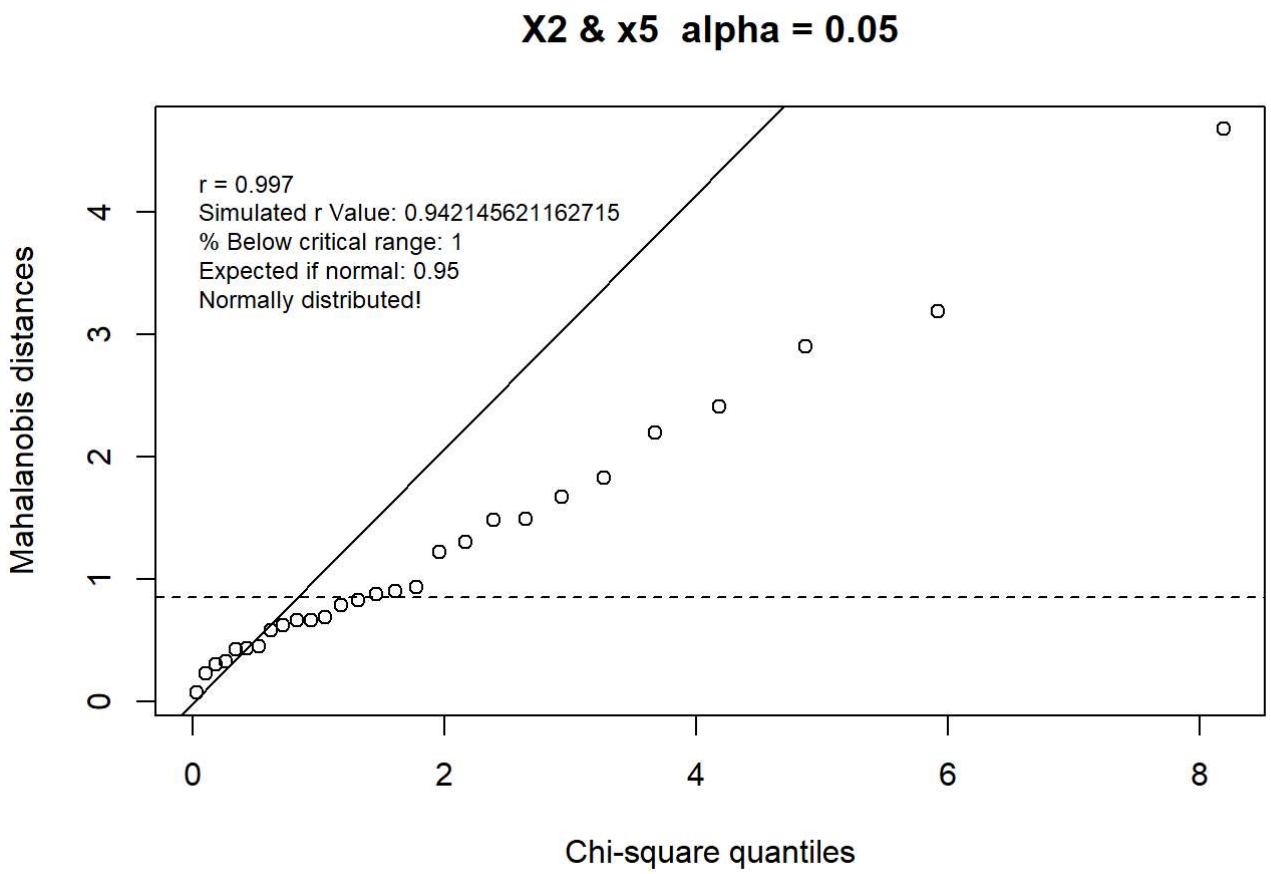

X1 & x3 alpha = 0.05
r = 0.971
Simulated r Value: 0.942145621162715
% Below critical range: 1
Expected if normal: 0.95
Normally distributed!
Mahalanobis distances
Chi-square quantiles
0
1
2
3
4
5
0
2
4
6
8

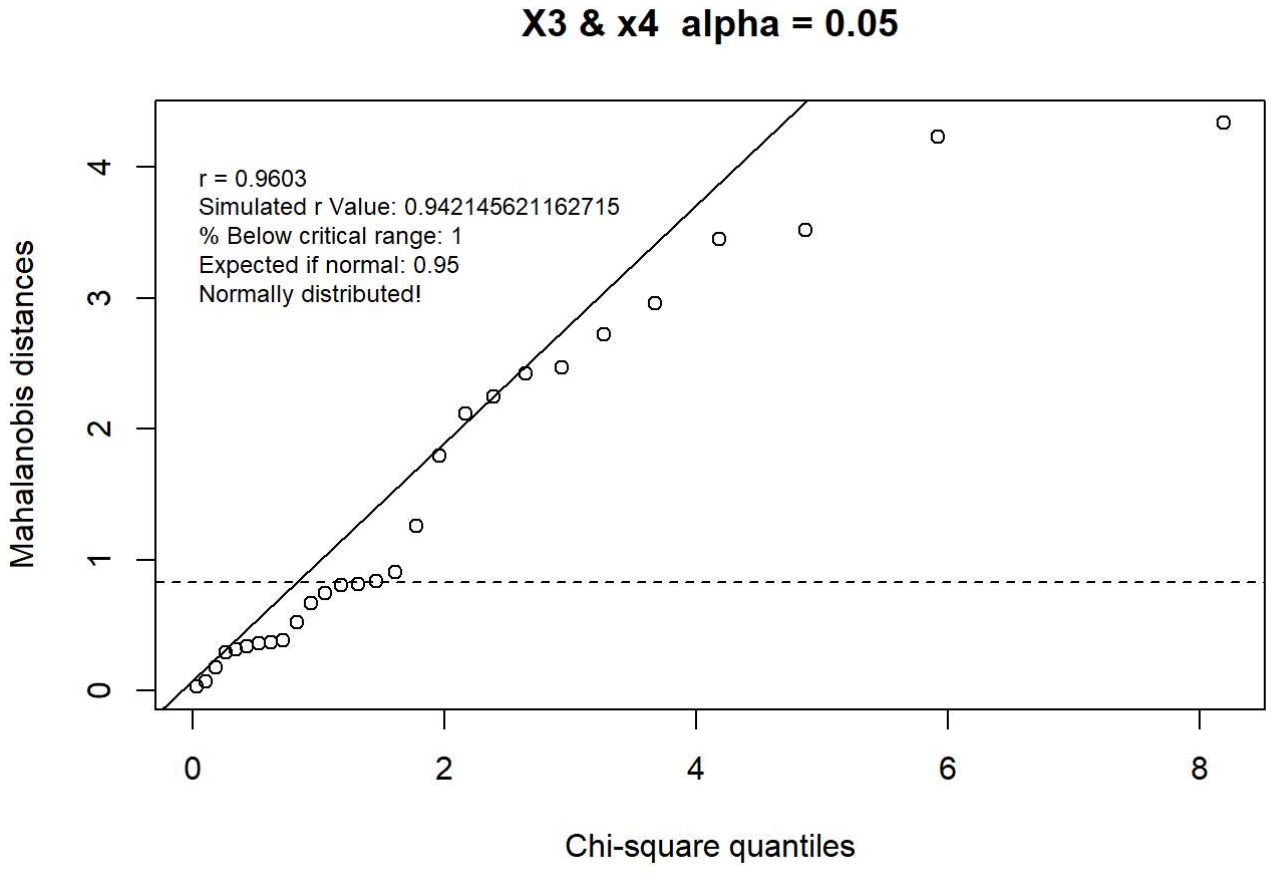

X1 & x4 alpha = 0.05
r = 0.9955
Simulated r Value: 0.942145621162715
% Below critical range: 1
Expected if normal: 0.95
Normally distributed!
Mahalanobis distances
Chi-square quantiles
0
1
2
3
4
5
0
2
4
6
8

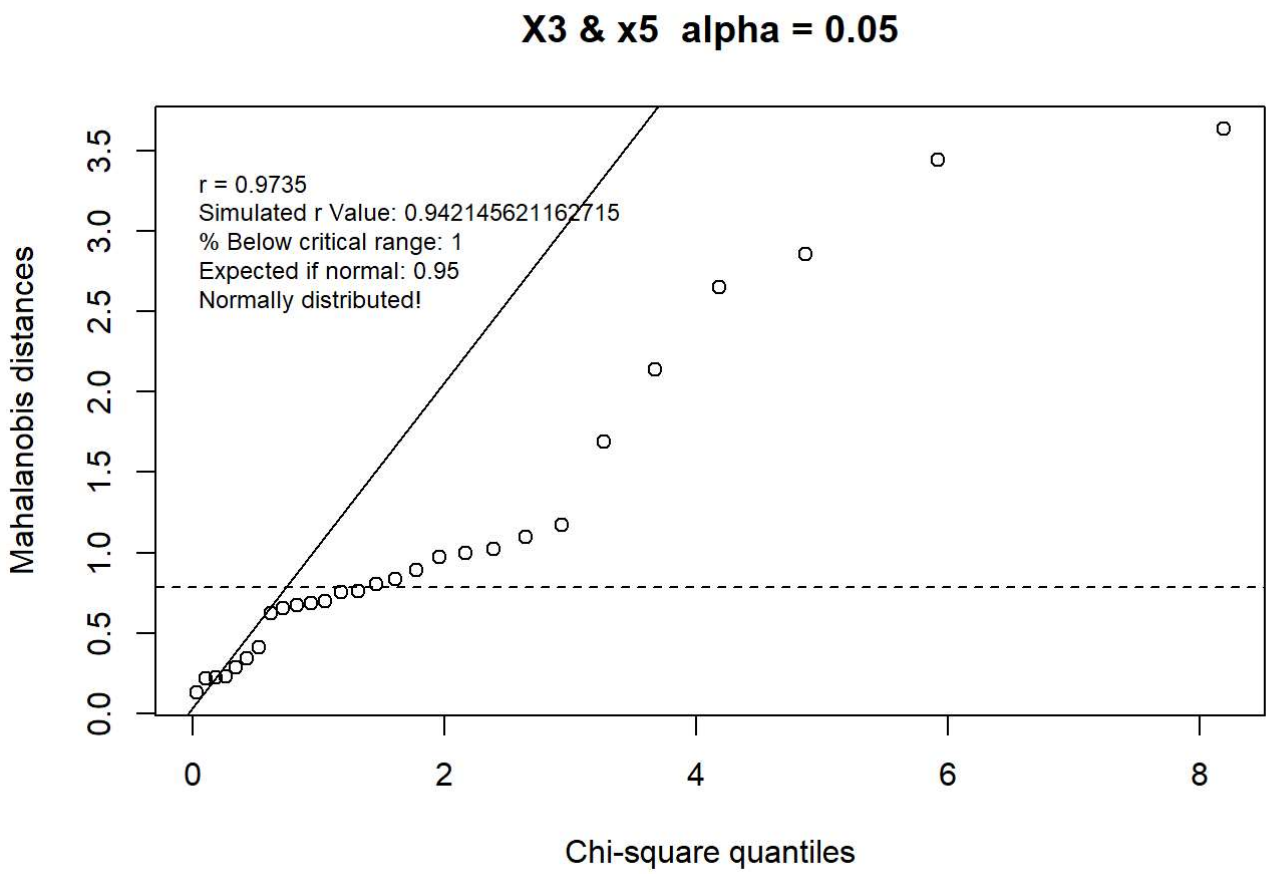

X1 & x5 alpha = 0.05
r = 0.9932
Simulated r Value: 0.942145621162715
% Below critical range: 1
Expected if normal: 0.95
Normally distributed!
Mahalanobis distances
Chi-square quantiles
0
1
2
3
4
0
2
4
6
8

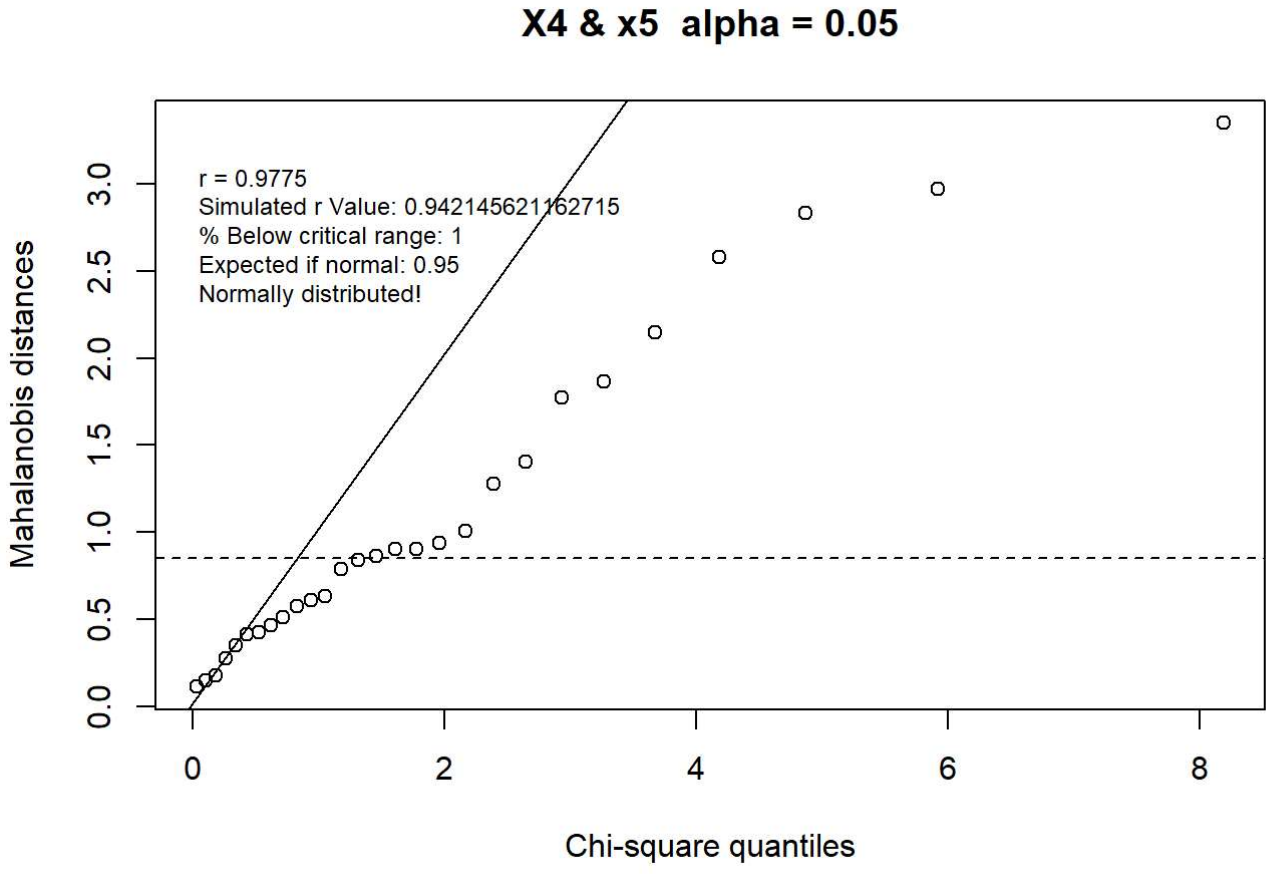

X2 & x3 alpha = 0.05
r = 0.9573
Simulated r Value: 0.942145621162715
% Below critical range: 1
Expected if normal: 0.95
Normally distributed!
Mahalanobis distances
Chi-square quantiles
0
1
2
3
4
0
2
4
6
8


X2 & x4 alpha = 0.05
r = 0.959
Simulated r Value: 0.942145621162715
% Below critical range: 0.9643
Expected if normal: 0.95
Normally distributed!
Mahalanobis distances
Chi-square quantiles
0
2
4
6
0
2
4
6
8


X2 & x5 alpha = 0.05
r = 0.997
Simulated r Value: 0.942145621162715
% Below critical range: 1
Expected if normal: 0.95
Normally distributed!
Mahalanobis distances
Chi-square quantiles
0
1
2
3
4
0
2
4
6
8


X3 & x4 alpha = 0.05
r = 0.9603
Simulated r Value: 0.942145621162715
% Below critical range: 1
Expected if normal: 0.95
Normally distributed!
Mahalanobis distances
Chi-square quantiles
0
1
2
3
4
0
2
4
6
8

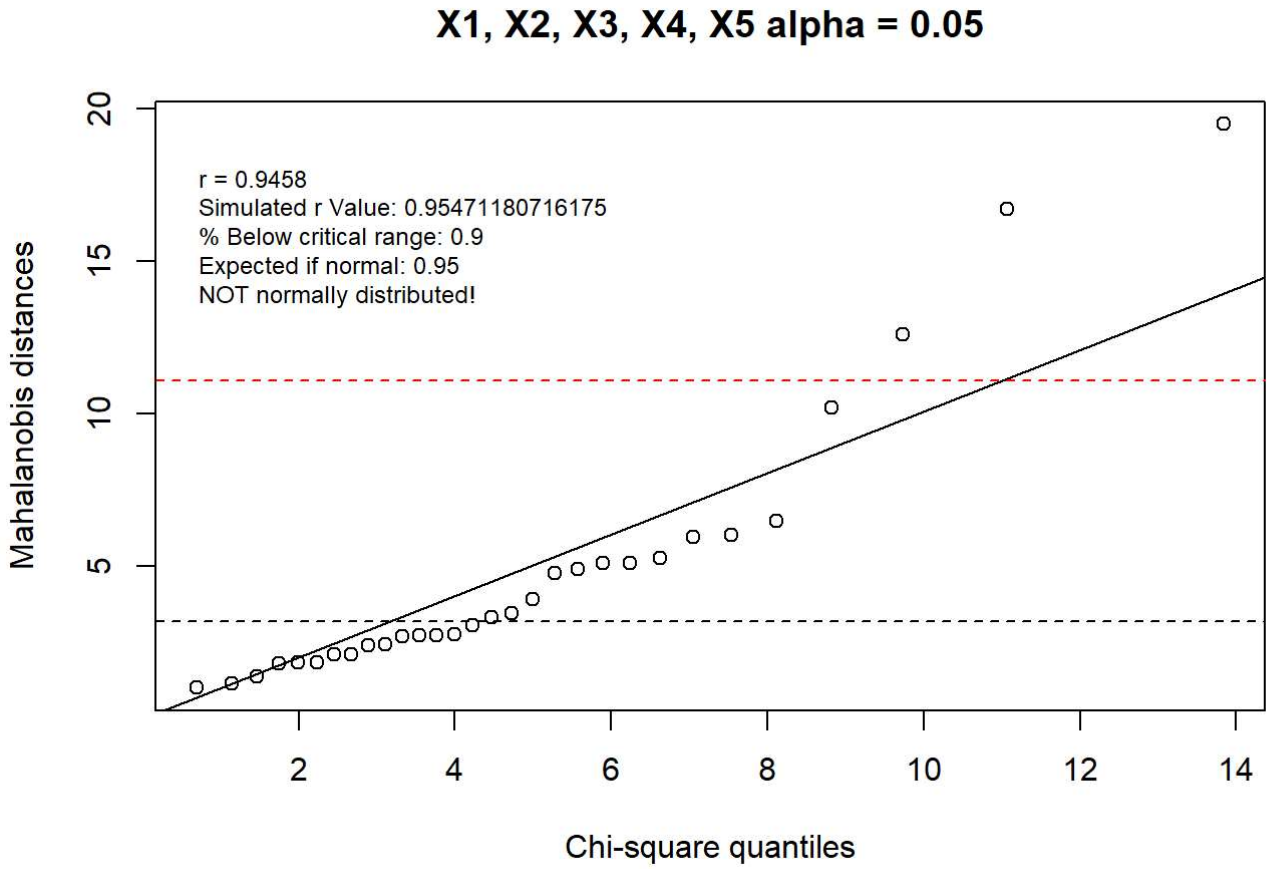




Interestingtly all the data is normally distributed if we just remove the two outliers.
Let us now try to do it with all attributes.

Normality of all attributes

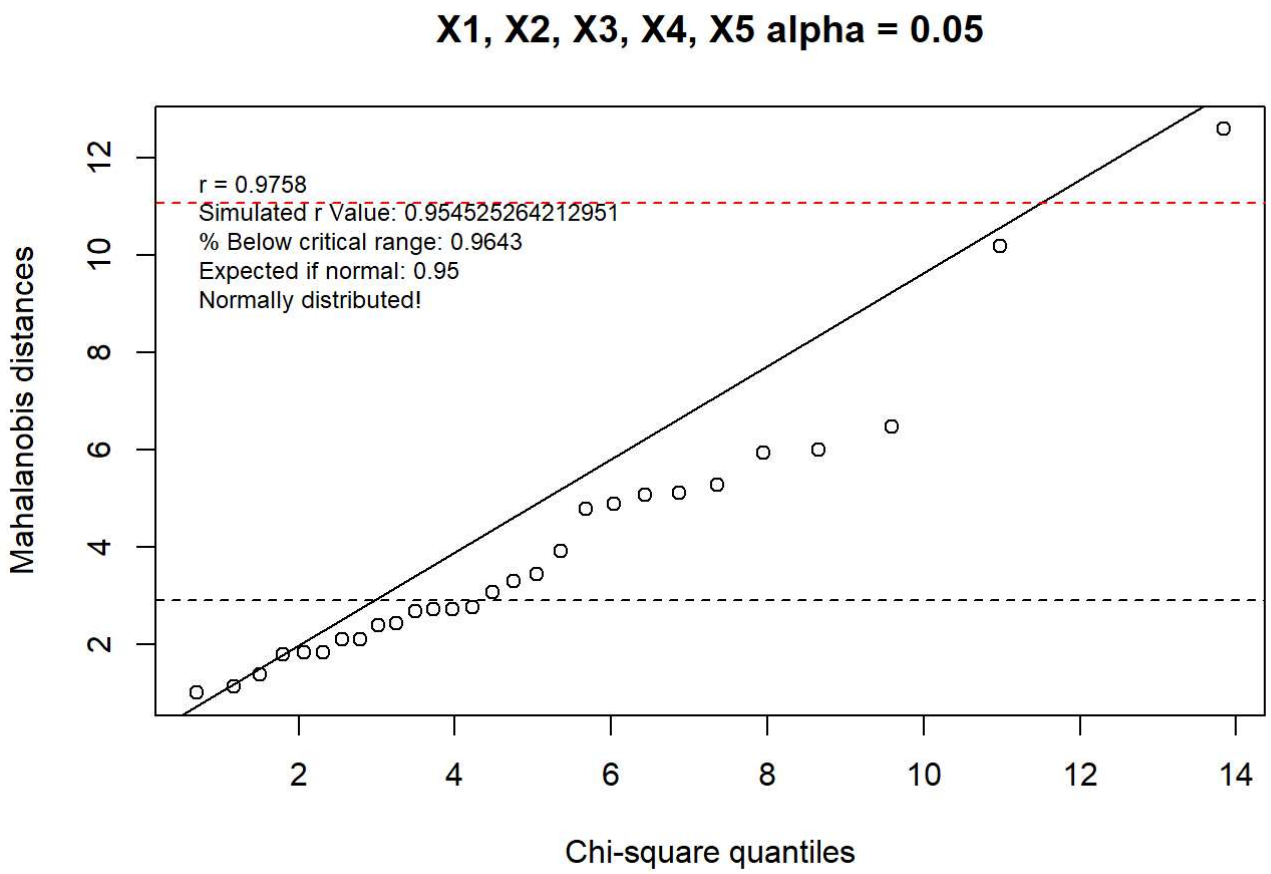
```
sim_cor <- FindcrikChi(n = length(df[,1]),
                       p = ncol(df),
                       alpha = 0.05,
                       n_simulations = 10000)

multi_var_norm(df, sim_cor, 0.05, "X1, X2, X3, X4, X5")
```



The whole data set is not normally distributed, which fits the theorem that states, that if a subset is not normally distributed then the whole data set will not be normally distributed.

What if we remove two outliers again?

```
sim_cor <- FindcrikChi(n = length(df[,1]),
                       p = ncol(df),
                       alpha = 0.05,
                       n_simulations = 10000)

multi_var_norm(df, sim_cor, 0.05, "X1, X2, X3, X4, X5", remove_outlier = TRUE, n_outliers =
2)
```



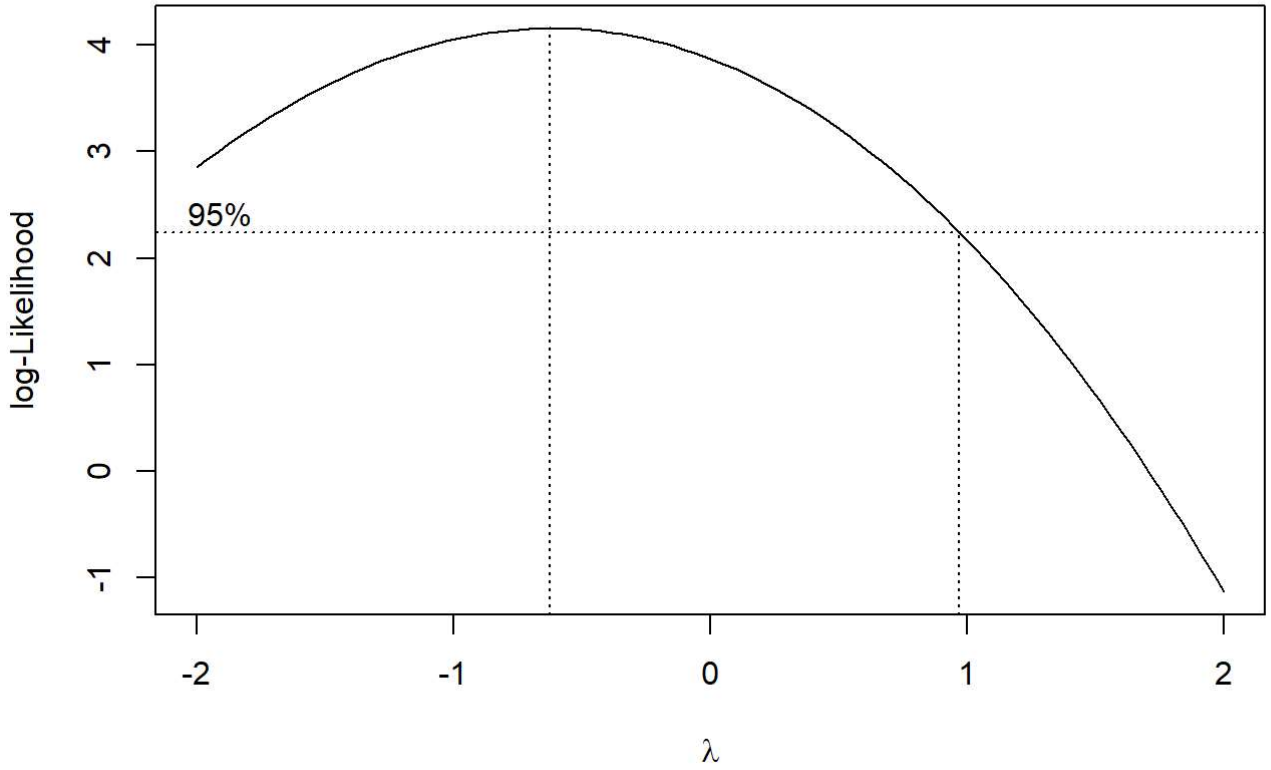
Then the data set is suddenly normally distributed. Let's try and transform the dataset.

Transformed multivariate dataset

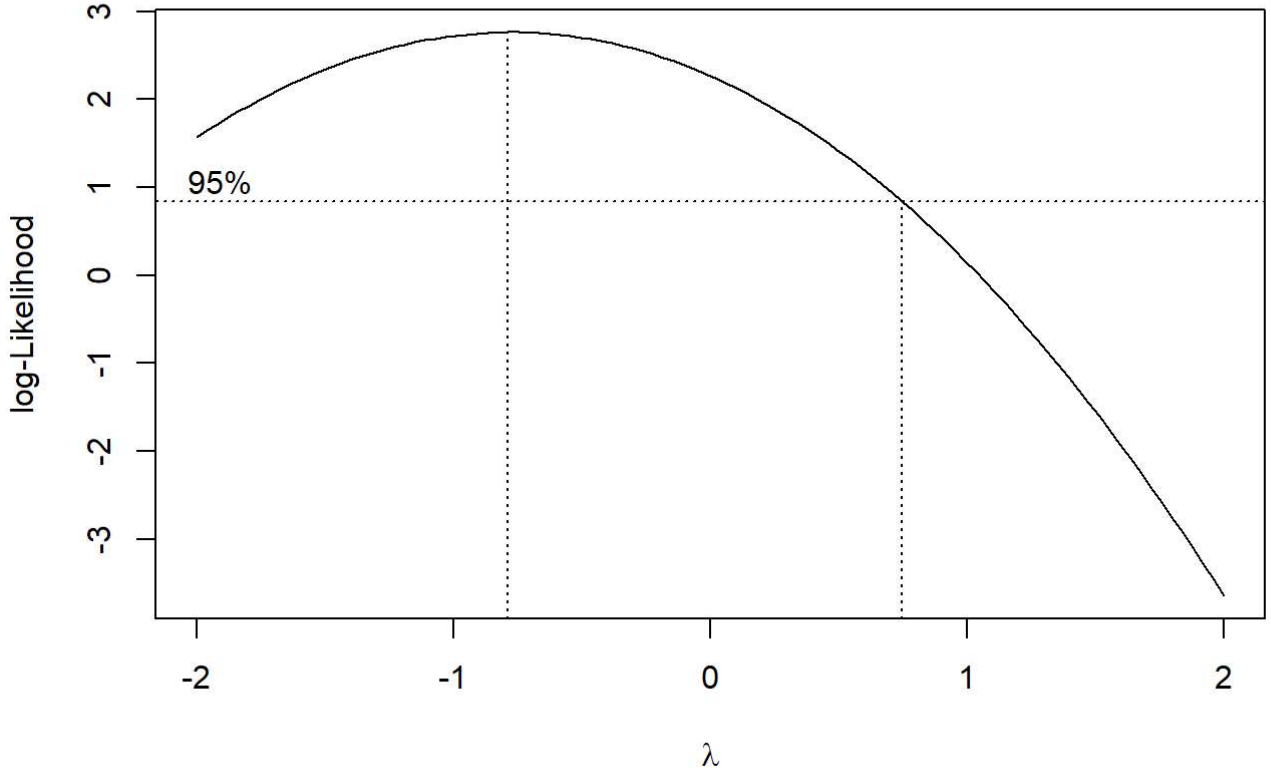
Each column is transformed with box_cox transformation

```
transformed_df <- df
i = 1
for (col_name in colnames(df)) {
  print(paste0("X", i))
  transformed_df[, i] <- box_cox_transformation(df[, i])
  i = i + 1
}
```

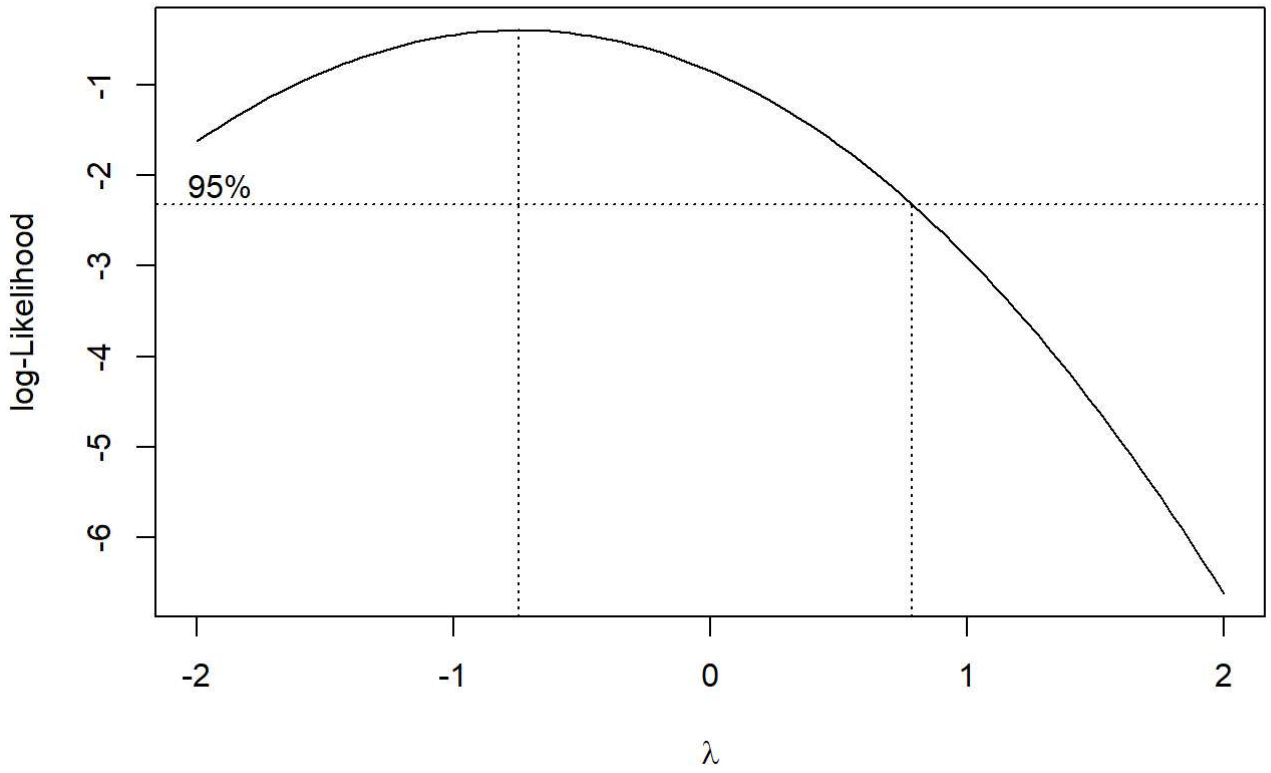
```
## [1] "X1"
```



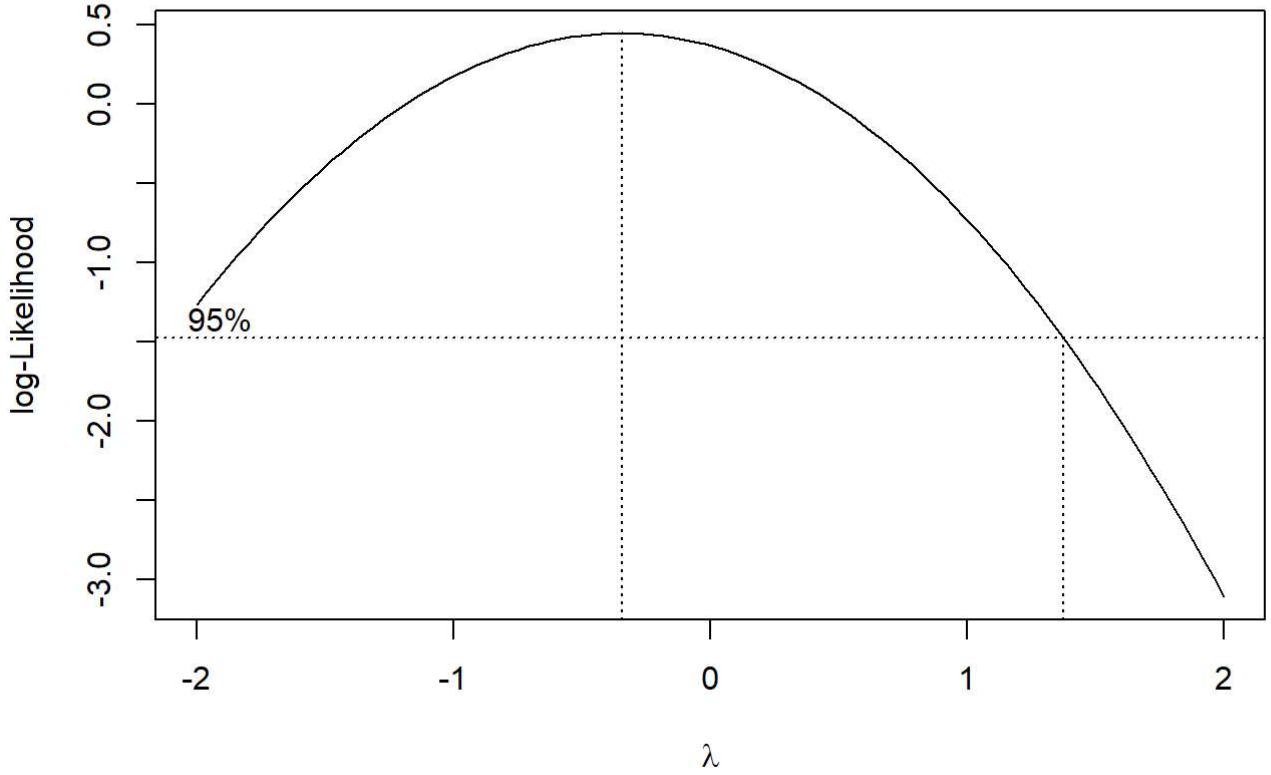

```
## [1] "The best lambda is  -0.626262626262626"
## [1] "X2"
```



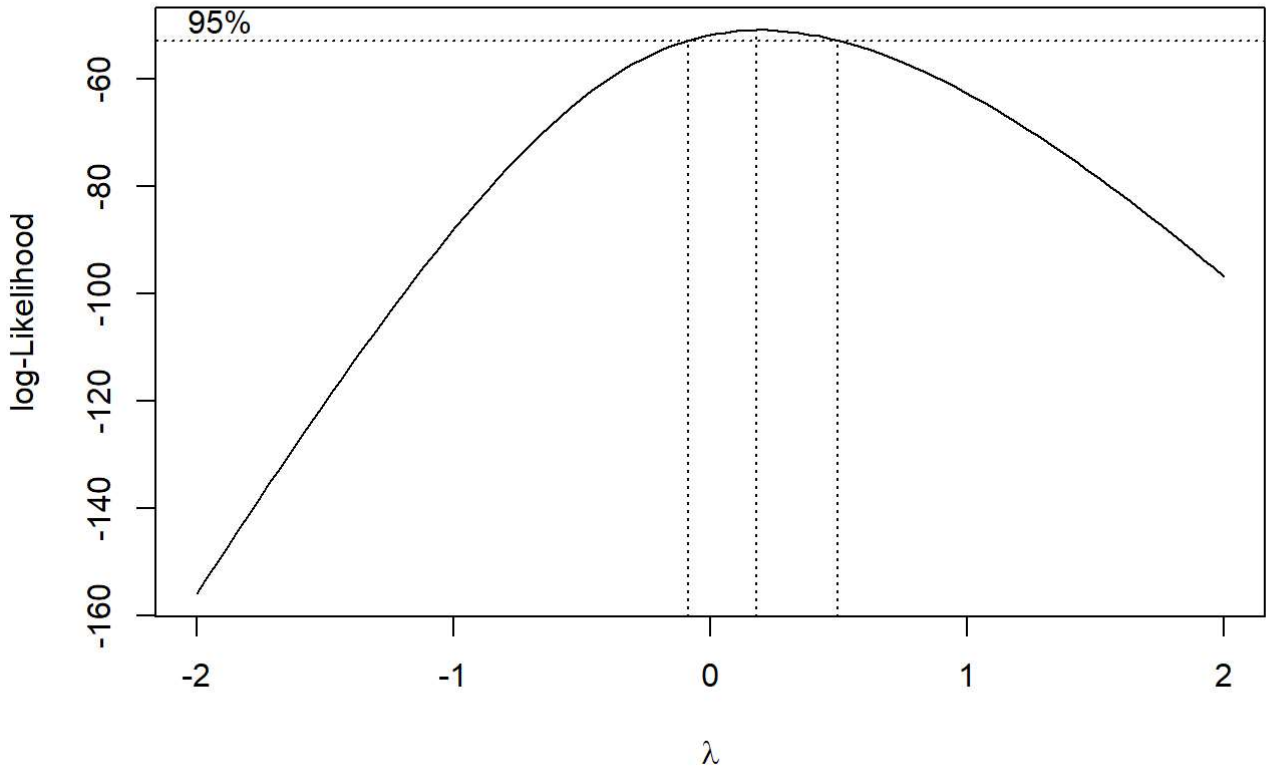
```
## [1] "The best lambda is  -0.787878787878788"
## [1] "X3"
```




```
## [1] "The best lambda is  -0.747474747474747"
## [1] "X4"
```




```
## [1] "The best lambda is  -0.343434343434343"
## [1] "X5"
```



```
## [1] "The best lambda is  0.181818181818182"
```

```
print(transformed_df)
```

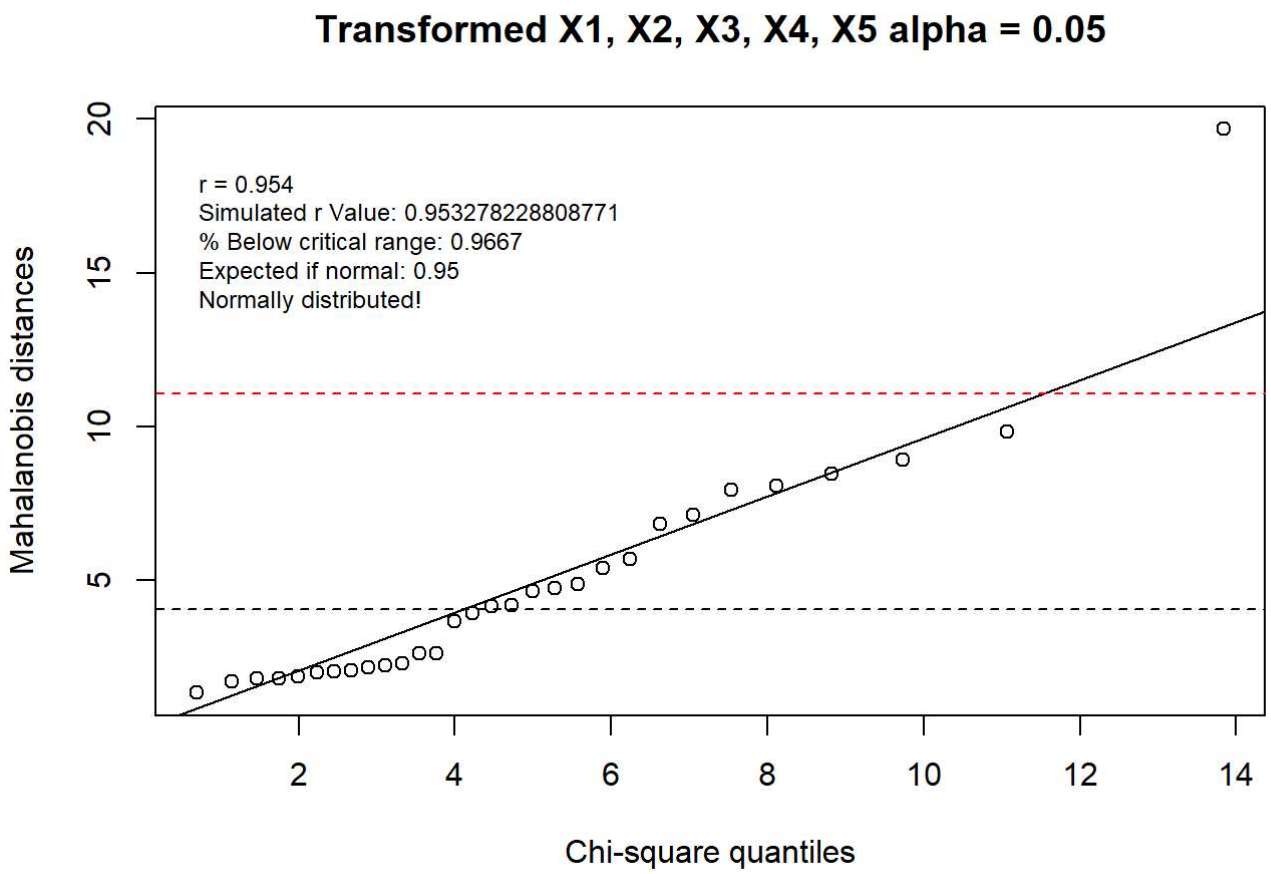
```
##          V1       V2       V3       V4          V5
## 1  1.582601 1.265530 1.332350 2.688912 -0.48782122
## 2  1.584584 1.266107 1.333421 2.704533  1.99352960
## 3  1.583585 1.265614 1.332935 2.705336  2.45642803
## 4  1.581319 1.265487 1.330757 2.677271  1.92500621
## 5  1.582995 1.265939 1.332485 2.693260  0.34697751
## 6  1.581700 1.265634 1.332006 2.677950  0.85822596
## 7  1.582849 1.265589 1.331439 2.684111  1.86698968
## 8  1.583526 1.265803 1.332727 2.692900  0.41358852
## 9  1.586128 1.266785 1.333874 2.715686  3.17501127
## 10 1.581880 1.265437 1.331833 2.675943 -0.25525184
## 11 1.581689 1.265420 1.332234 2.683923  0.69843713
## 12 1.583292 1.265927 1.332517 2.693855 -0.72420219
## 13 1.582366 1.265834 1.332438 2.687297  1.08858973
## 14 1.582497 1.265589 1.332164 2.684437 -1.70456568
## 15 1.582458 1.265526 1.331849 2.686088  0.07750202
## 16 1.582898 1.266224 1.331073 2.662353  3.69138280
## 17 1.579077 1.264376 1.330194 2.654919  1.40691565
## 18 1.579820 1.264946 1.331366 2.664133  1.57343698
## 19 1.582307 1.265499 1.332455 2.687913  0.31624231
## 20 1.581772 1.265426 1.331592 2.682929  0.39175972
## 21 1.584163 1.266267 1.332313 2.701671  2.84424929
## 22 1.582648 1.265463 1.331954 2.674255  1.88567267
## 23 1.581248 1.265266 1.331509 2.676371 -0.21867751
## 24 1.583354 1.265871 1.332563 2.699081  1.01581612
## 25 1.582444 1.265224 1.331741 2.677271  1.75303961
## 26 1.581783 1.265044 1.331311 2.673811  1.37060878
## 27 1.583773 1.265912 1.332691 2.691273  0.93918981
## 28 1.581377 1.265571 1.331929 2.680542  1.21602043
## 29 1.584333 1.266381 1.333386 2.705718  2.18150400
## 30 1.580331 1.264973 1.331215 2.662553  1.03435366
```

```
print(df)
```

```
##      V1   V2   V3   V4    V5
## 1  1889 1651 1561 1778  0.60
## 2  2403 2048 2087 2197  5.48
## 3  2119 1700 1815 2222  7.62
## 4  1645 1627 1110 1533  5.21
## 5  1976 1916 1614 1883  1.40
## 6  1712 1712 1439 1546  2.22
## 7  1943 1685 1271 1671  4.99
## 8  2104 1820 1717 1874  1.49
## 9  2983 2794 2412 2581 12.26
## 10 1745 1600 1384 1508  0.77
## 11 1710 1591 1518 1667  1.93
## 12 2046 1907 1627 1898  0.46
## 13 1840 1841 1595 1741  2.70
## 14 1867 1685 1493 1678  0.13
## 15 1859 1649 1389 1714  1.08
## 16 1954 2149 1180 1281 16.85
## 17 1325 1170 1002 1176  3.50
## 18 1419 1371 1252 1308  3.99
## 19 1828 1634 1602 1755  1.36
## 20 1725 1594 1313 1646  1.46
## 21 2276 2189 1547 2111  9.90
## 22 1899 1614 1422 1477  5.06
## 23 1633 1513 1290 1516  0.80
## 24 2061 1867 1646 2037  2.54
## 25 1856 1493 1356 1533  4.58
## 26 1727 1412 1238 1469  3.40
## 27 2168 1896 1701 1834  2.38
## 28 1655 1675 1414 1597  3.00
## 29 2326 2301 2065 2234  6.28
## 30 1490 1382 1214 1284  2.58
```

Let's see how we perform with the transformed data:

```
sim_cor <- FindcrikChi(n = length(transformed_df[,1]),
                       p = ncol(transformed_df),
                       alpha = 0.05,
                       n_simulations = 10000)
multi_var_norm(transformed_df, sim_cor, alpha = 0.05, name ="Transformed X1, X2, X3, X4, X5")
```



Now the data is normally distributed and it even has all the excted points within the critical value.