Advanced Mathematical Statistics: Assignment 3

Chris Hayduk

October 10, 2019

1 Problems to be Completed

Problem 4.23.

a) Let $\mathbf{X}' = \begin{bmatrix} -0.6 & 3.1 & 25.3 & -16.8 & -7.1 & -6.2 & 16.1 & 25.2 & 22.6 & 26.0 \end{bmatrix}$.

In order to construct a Q-Q plot for the data, we need to compute the quantiles for the data as well as the quantiles for the theoretical normal distribution. Then we can plot the ordered data against the theoretical quantile as an ordered pair. After doing so, we can assess the whether the distribution is normal by checking the linearity of the plot. In order to do this, I will use the R programming language.

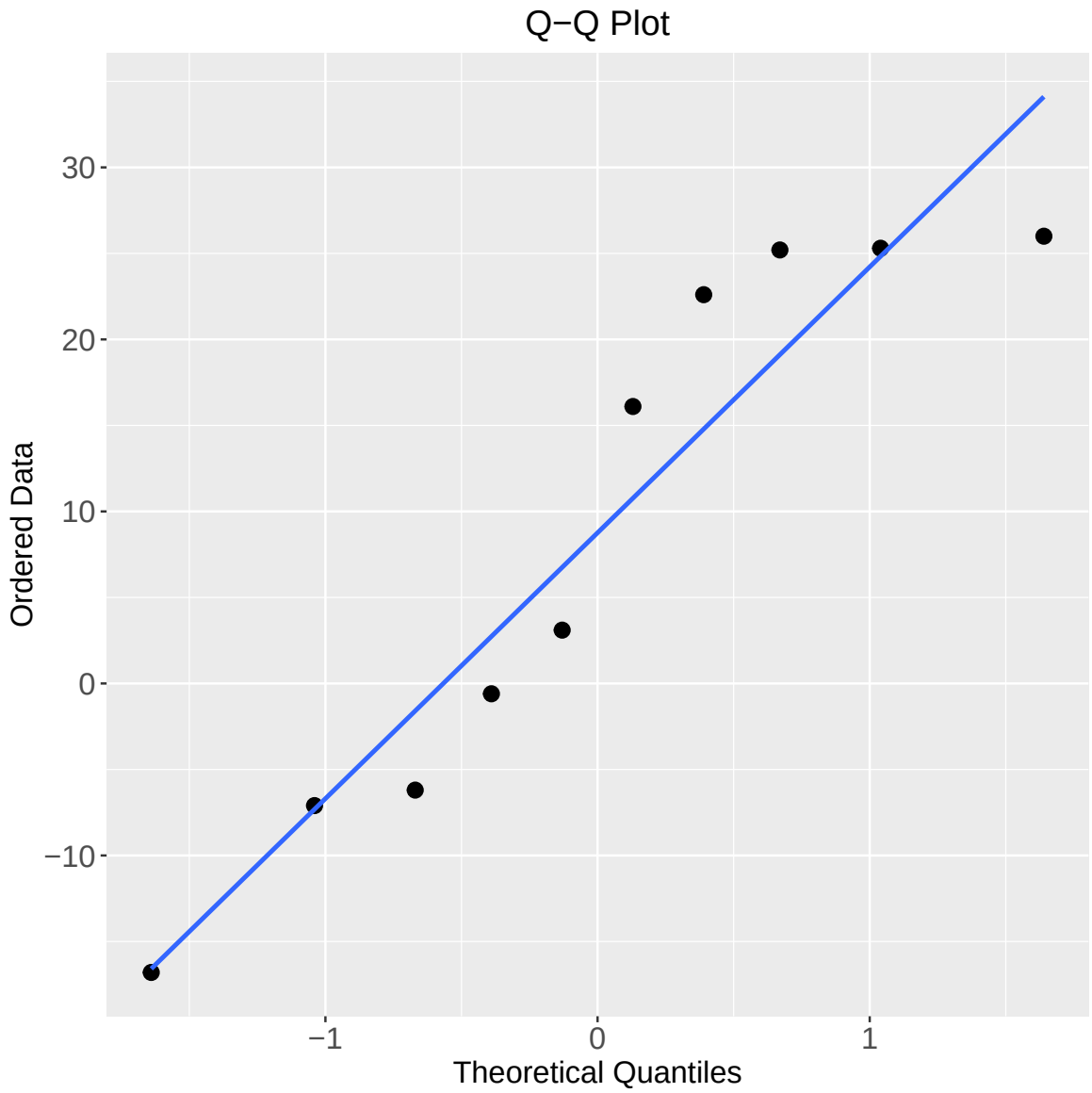
```
#Create data vector
x <- c(-0.6, 3.1, 25.3, -16.8, -7.1, -6.2, 16.1, 25.2, 22.6, 26.0)

#Sort data
x <- sort(x)

#Let n = # of observations
n <- length(x)

#Output sample size
n

## [1] 10

#Calculate the quantiles for the actual data
prob_levels <- ((1:n)-0.5)/n

#Calculate the theoretical normal quantiles
standard_normal_quantiles <- qnorm(prob_levels)

#Round the theoretical quantiles to 2 decimal places
```

```
standard_normal_quantiles <- round(standard_normal_quantiles,
                                   digits = 2)

#Create matrix of values
qq_matrix <- as.data.frame(cbind(x, prob_levels,
                                 standard_normal_quantiles))

#Print matrix
qq_matrix
```

```
##        x prob_levels standard_normal_quantiles
## 1  -16.8        0.05                     -1.64
## 2   -7.1        0.15                     -1.04
## 3   -6.2        0.25                     -0.67
## 4   -0.6        0.35                     -0.39
## 5    3.1        0.45                     -0.13
## 6   16.1        0.55                      0.13
## 7   22.6        0.65                      0.39
## 8   25.2        0.75                      0.67
## 9   25.3        0.85                      1.04
## 10  26.0        0.95                      1.64
```

Now that we have the data required to construct the Q-Q plot, we will use R to plot the ordered data against the theoretical quantiles. We will also include a line of best fit in order to see if the sample and theoretical quantiles are linearily related.

```
ggplot(qq_matrix, aes(standard_normal_quantiles, x)) +
  geom_point(size=3) +
  geom_smooth(method='lm', se=F) +
  xlab("Theoretical Quantiles") +
  ylab("Ordered Data") +
  ggtitle("Q-Q Plot") +
  theme.info
```



There do not appear to be any large deviations from the line of best fit, so it is reasonable to assume that the data is in fact normally distributed.

b) We will use R to compute the correlation coefficient between the standard normal quantiles and the sample quantiles.

```
#Calculate mean of data
x_bar <- mean(qq_matrix$x)

#Calculate three components of correlation equation separately
#(as in Example 4.11 in textbook)
part1 <- sum((qq_matrix$x - x_bar)*qq_matrix$standard_normal_quantiles)
part2 <- sum((qq_matrix$x - x_bar)^2)
part3 <- sum((qq_matrix$standard_normal_quantiles)^2)
```

```r
#Compute correlation
r_Q <- part1/(sqrt(part2)*sqrt(part3))

#Output correlation
r_Q

## [1] 0.9476294

#Check against R's built in correlation function
cor(qq_matrix$x, qq_matrix$standard_normal_quantiles)

## [1] 0.9476294
```

From Table 4.2, for a sample size of 10 and $\alpha = 0.1$, the correlation needs to be at least 0.9351 in order for us to not reject the hypothesis of normality. As can be seen from the computation above, the correlation coefficient between the data and the standard normal quantiles is greater than the threshold value. Thus, we cannot reject the hypothesis of normality.

Problem 4.24.

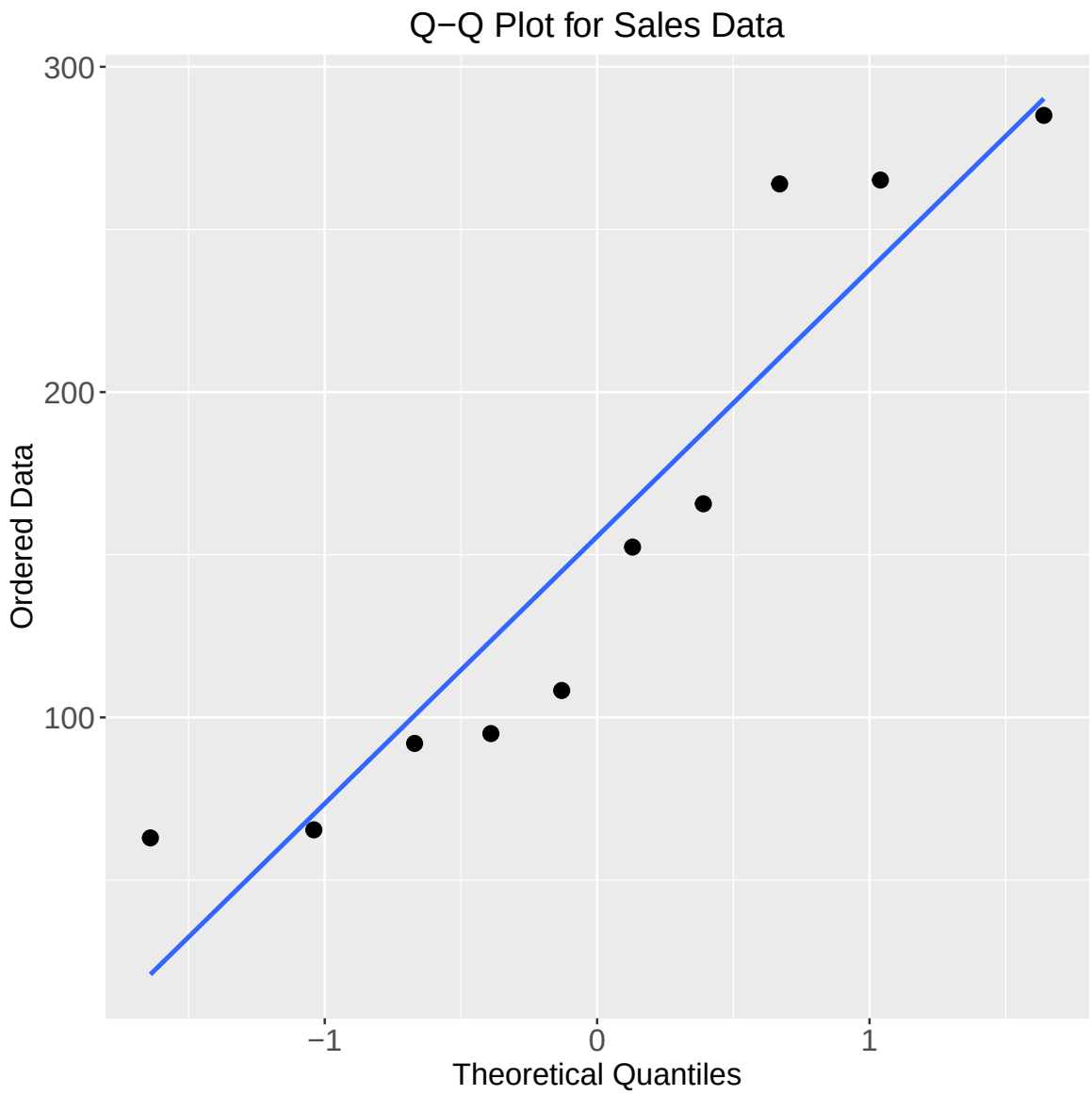
a) We will construct the Q-Q plots as in the above exercise. First, we will begin with the sales data.

```r
#Create data vector
sales <- c(108.28, 152.36, 95.04, 65.45, 62.97,
           263.99, 265.19, 285.06, 92.01, 165.68)
x <- sales

#Sort data
x <- sort(x)

#Let n = # of observations
n <- length(x)

#Output sample size
n

## [1] 10

#Calculate the quantiles for the actual data
prob_levels <- ((1:n)-0.5)/n
```

```r
#Calculate the theoretical normal quantiles
standard_normal_quantiles <- qnorm(prob_levels)

#Round the theoretical quantiles to 2 decimal places
standard_normal_quantiles <- round(standard_normal_quantiles,
                                   digits = 2)

#Create matrix of values
qq_matrix1 <- as.data.frame(cbind(x, prob_levels,
                                  standard_normal_quantiles))

#Print matrix
qq_matrix1
```

```
##         x prob_levels standard_normal_quantiles
## 1   62.97        0.05                     -1.64
## 2   65.45        0.15                     -1.04
## 3   92.01        0.25                     -0.67
## 4   95.04        0.35                     -0.39
## 5  108.28        0.45                     -0.13
## 6  152.36        0.55                      0.13
## 7  165.68        0.65                      0.39
## 8  263.99        0.75                      0.67
## 9  265.19        0.85                      1.04
## 10 285.06        0.95                      1.64
```

```r
#Output plot
ggplot(qq_matrix1, aes(standard_normal_quantiles, x)) +
  geom_point(size=3) +
  geom_smooth(method='lm', se=F) +
  xlab("Theoretical Quantiles") +
  ylab("Ordered Data") +
  ggtitle("Q-Q Plot for Sales Data") +
  theme.info
```



Most of the points appear close to the line of best fit. However, there are a couple points that may be considered significant outliers. As a result, it is difficult to say whether the data is normally distributed.

Now, we will perform the same analysis for the profits data.

```
#Create data vector
profits <- c(17.05, 16.59, 10.91, 14.14, 9.52,
             25.33, 18.54, 15.73, 8.10, 11.13)
x <- profits

#Sort data
x <- sort(x)

#Let n = # of observations
```

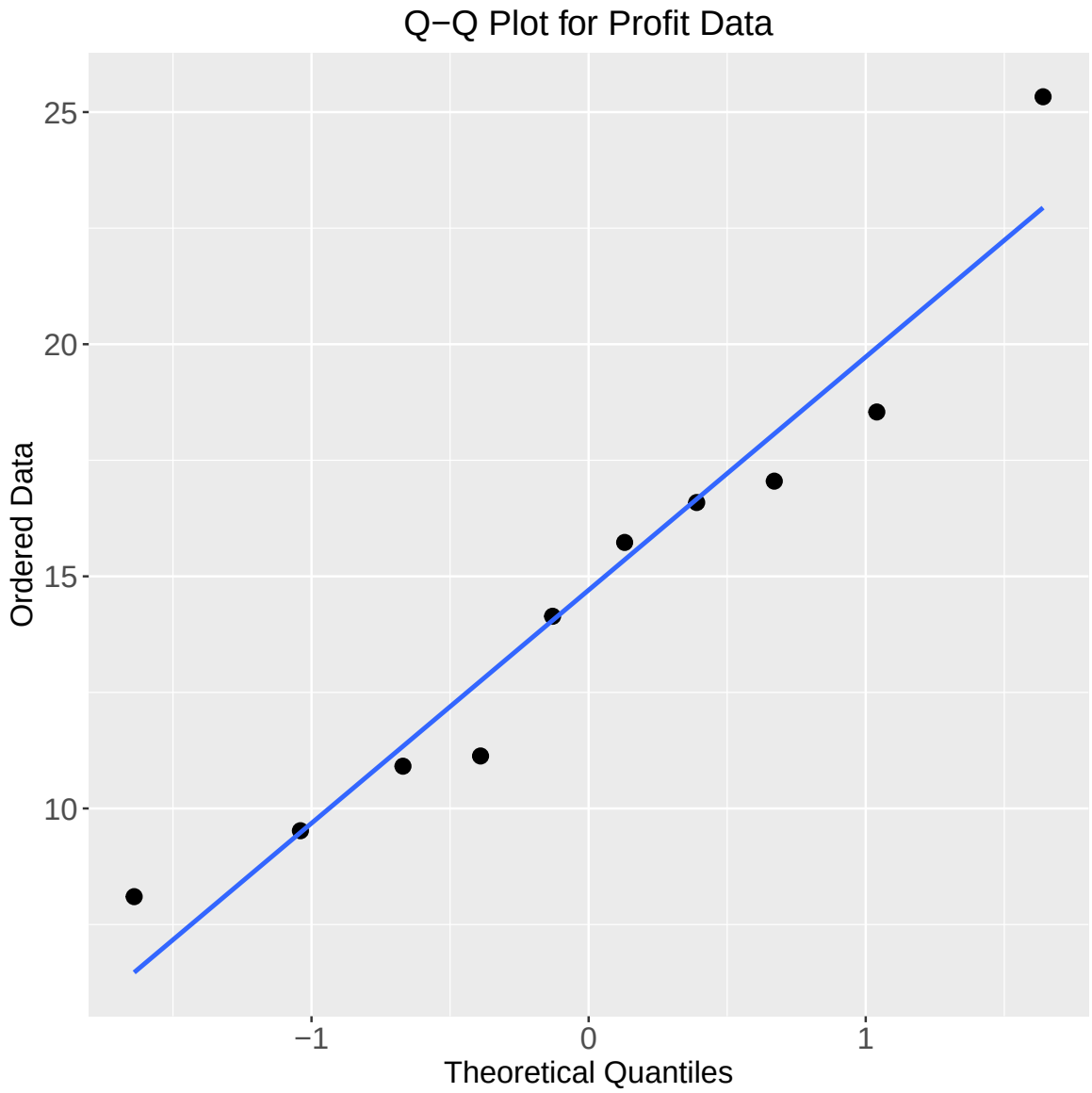
```r
n <- length(x)

#Output sample size
n
```

```
## [1] 10
```

```r
#Calculate the quantiles for the actual data
prob_levels <- ((1:n)-0.5)/n

#Calculate the theoretical normal quantiles
standard_normal_quantiles <- qnorm(prob_levels)

#Round the theoretical quantiles to 2 decimal places
standard_normal_quantiles <- round(standard_normal_quantiles,
                                   digits = 2)

#Create matrix of values
qq_matrix2 <- as.data.frame(cbind(x, prob_levels,
                                  standard_normal_quantiles))

#Print matrix
qq_matrix2
```

```
##        x prob_levels standard_normal_quantiles
## 1   8.10        0.05                     -1.64
## 2   9.52        0.15                     -1.04
## 3  10.91        0.25                     -0.67
## 4  11.13        0.35                     -0.39
## 5  14.14        0.45                     -0.13
## 6  15.73        0.55                      0.13
## 7  16.59        0.65                      0.39
## 8  17.05        0.75                      0.67
## 9  18.54        0.85                      1.04
## 10 25.33        0.95                      1.64
```

```r
#Output plot
ggplot(qq_matrix2, aes(standard_normal_quantiles, x)) +
  geom_point(size=3) +
  geom_smooth(method='lm', se=F) +
  xlab("Theoretical Quantiles") +
  ylab("Ordered Data") +
  ggtitle("Q-Q Plot for Profit Data") +
  theme.info
```



The points in this plot appear to closely follow the line of best fit. Thus, the data is likely normally distributed.

b) We will now perform correlation analysis for each of the two data vectors. We will again begin with the sales data.

```
#Calculate mean of data
x_bar <- mean(qq_matrix1$x)

#Calculate three components of correlation equation separately
#(as in Example 4.11 in textbook)
part1 <- sum((qq_matrix1$x - x_bar)*qq_matrix1$standard_normal_quantiles)
part2 <- sum((qq_matrix1$x - x_bar)^2)
part3 <- sum((qq_matrix1$standard_normal_quantiles)^2)
```

```r
#Compute correlation
r_Q <- part1/(sqrt(part2)*sqrt(part3))

#Output correlation
r_Q

## [1] 0.9374333

#Check against R's built in correlation function
cor(qq_matrix1$x, qq_matrix1$standard_normal_quantiles)

## [1] 0.9374333
```

From Table 4.2, for a sample size of 10 and $\alpha = 0.1$, the correlation needs to be at least 0.9351 in order for us to not reject the hypothesis of normality. As can be seen from the computation above, the correlation coefficient between the sales data and the standard normal quantiles is greater than the threshold value. Thus, we cannot reject the hypothesis of normality.

We will now perform the same analysis for the profit data.

```r
#Calculate mean of data
x_bar <- mean(qq_matrix2$x)

#Calculate three components of correlation equation separately
#(as in Example 4.11 in textbook)
part1 <- sum((qq_matrix2$x - x_bar)*qq_matrix2$standard_normal_quantiles)
part2 <- sum((qq_matrix2$x - x_bar)^2)
part3 <- sum((qq_matrix2$standard_normal_quantiles)^2)

#Compute correlation
r_Q <- part1/(sqrt(part2)*sqrt(part3))

#Output correlation
r_Q

## [1] 0.969226

#Check against R's built in correlation function
cor(qq_matrix2$x, qq_matrix2$standard_normal_quantiles)

## [1] 0.969226
```

From Table 4.2, for a sample size of 10 and $\alpha = 0.1$, the correlation needs to be at least 0.9351 in order for us to not reject the hypothesis of normality. As can be seen from the computation above, the correlation coefficient between the profit data and the standard normal quantiles is greater than the threshold value. Thus, we cannot reject the hypothesis of normality.

The above correlation values agree with our visual assessment of the Q-Q plots. Both the plots do not indicate that the data is not normally distributed, with the profit data Q-Q plot appearing to deviate from the line of best fit less than the sales data Q-Q plot. The correlation values provide a similar analysis, with both data sets meeting the correlation threshold, while the profit data exhibited a stronger correlation with the theoretical quantiles.

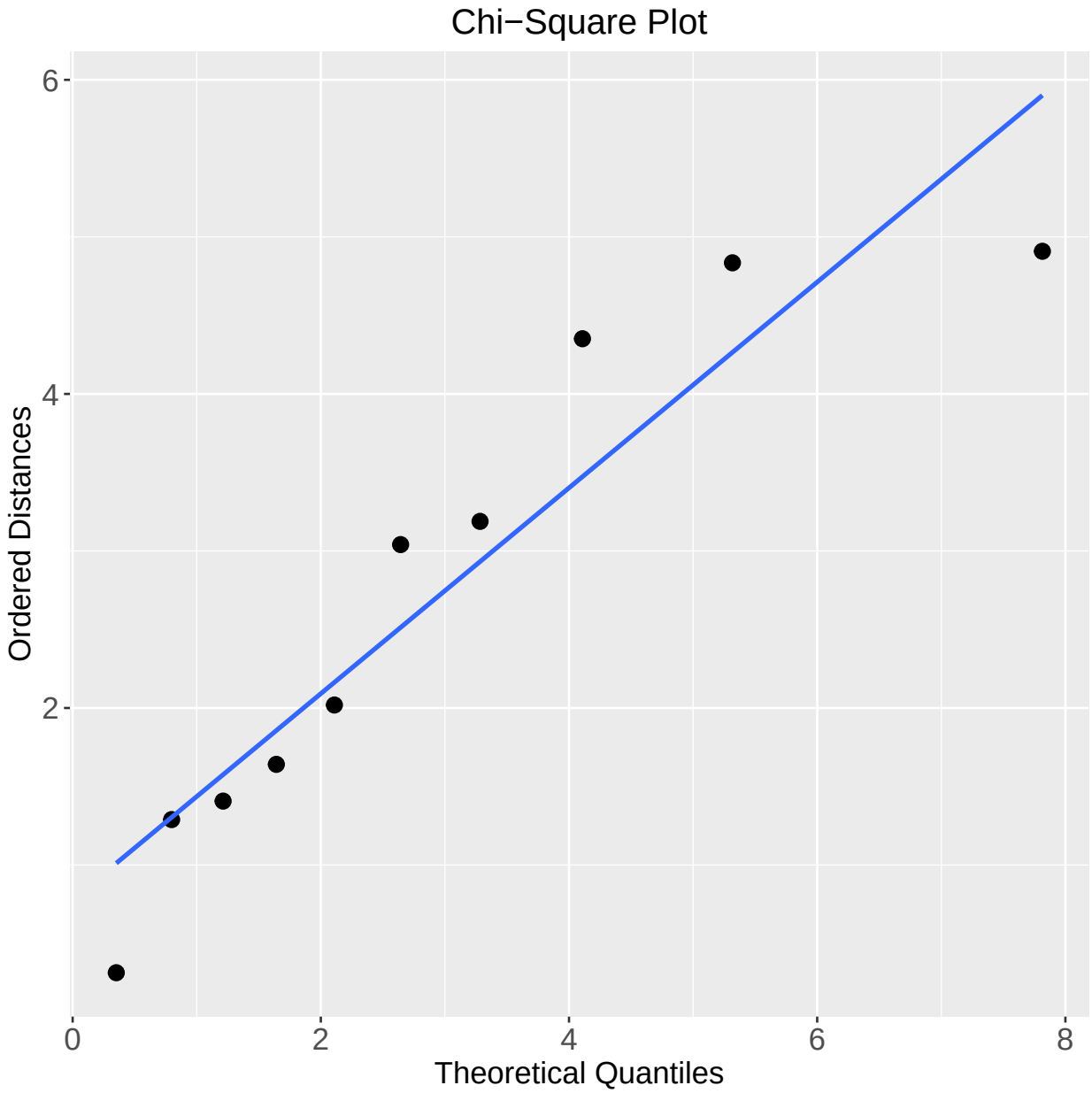
Problem 4.25.

We will again use the R programming language, this time to compute a Chi-Square plot for the data from Exercise 1.4.

```
#Create data matrix
data <- matrix(c(108.28, 17.05, 1484.10, 152.36, 16.59,
                 750.33, 95.04, 10.91, 766.42, 65.45,
                 14.14, 1110.46, 62.97, 9.52, 1031.29,
                 263.99, 25.33, 195.26, 265.19, 18.54,
                 193.83, 285.06, 15.73, 191.11, 92.01,
                 8.10, 1175.16, 165.68, 11.13, 211.15),
               nrow = 10, ncol = 3, byrow= TRUE)

#Get mean of each variable
means <- apply(data, 2, mean)

print(means)

## [1] 155.603  14.704 710.911

#Create matrix of means for subtraction purposes
means_matrix <- matrix(data = rep(means, times = 10),
                       nrow = 10, ncol = 3, byrow = TRUE)

#Subtract mean from data
data_minus_mean <- data - means_matrix

print(data_minus_mean)

##          [,1]   [,2]     [,3]
##  [1,] -47.323  2.346  773.189
```

```
##  [2,]  -3.243  1.886   39.419
##  [3,] -60.563 -3.794   55.509
##  [4,] -90.153 -0.564  399.549
##  [5,] -92.633 -5.184  320.379
##  [6,] 108.387 10.626 -515.651
##  [7,] 109.587  3.836 -517.081
##  [8,] 129.457  1.026 -519.801
##  [9,] -63.593 -6.604  464.249
## [10,]  10.077 -3.574 -499.761

#Get covariance matrix
S <- cov(data)

#Get inverse of covariance matrix
S_inv <- as.matrix(inv(S))

#Create vector to store distance values
d <- rep(0, times = 10)

#Compute distance values
for(i in 1:nrow(data_minus_mean)){
  d[i] <- (t(data_minus_mean[i,]) %*% S_inv %*% (data_minus_mean[i,]))
}

#Sort the distances
d <- sort(d)

d

##  [1] 0.3142218 1.2894389 1.4070422 1.6411732 2.0191102 3.0405613 3.1884264
##  [8] 4.3513949 4.8347957 4.9084233

#Chi-square quantiles from textbook exercise
chisq_quantiles <- c(0.3518, 0.7978, 1.2125, 1.6416,
                     2.1095, 2.6430, 3.2831, 4.1083,
                     5.3170, 7.8147)

#Create data frame for plot
chisq_plot <- as.data.frame(cbind(d, chisq_quantiles))

#Output plot
ggplot(chisq_plot, aes(chisq_quantiles, d)) +
  geom_point(size=3) +
  geom_smooth(method='lm', se=F) +
  xlab("Theoretical Quantiles") +
```

```
  ylab("Ordered Distances") +
  ggtitle("Chi-Square Plot") +
  theme.info
```



Given the sample size, it is difficult to reject trivariate normality based upon the evidence in this graph.

Problem 4.30.

For this problem, we will be implementing the Box-Cox solution in order to find an appropriate power for λ. I will implement a function here so that this can be performed repeatedly.

```
box_cox <- function(data, lambda, n){
  #Create vector to store values for l of lambda
  l_of_lambda <- rep(-100, times = length(lambda))

  #For loop to calculate l of lambda
  for(i in 1:length(lambda)){
    #Check if lambda is equal to 0
    if(lambda[i] != 0){
      y <- (data^(lambda[i]) - 1)/lambda[i]
    } else{
      y <- log(data)
    }

    #Calculate mean of transformed x_1 values
    y_bar <- mean(y)

    #Calculate value for l of lambda function and store it in the vector
    l_of_lambda[i] <- (-n/2)*log((1/n) * sum((y - y_bar)^2)) +
      (lambda[i] - 1) * sum(log(data))
  }

  #Find the lambda value that corresponds to the max value of l of lambda
  lambda_final <- lambda[which.max(l_of_lambda)]

  return(lambda_final)
}
```

Furthermore, we will be using the same λ vectors and n values for all parts of this problem, so I will define those here:

```
#Create vector of possible lambda values
lambda <- seq(from = -5.00, to = 5.00, by = 0.01)

#Let n = length of x_1 vector
n <- 10
```

Now, we can move on to computing these transformations.

a) We will begin by finding the power transformation $\hat{\lambda}_1$ that makes the x_1 values approximately normal.

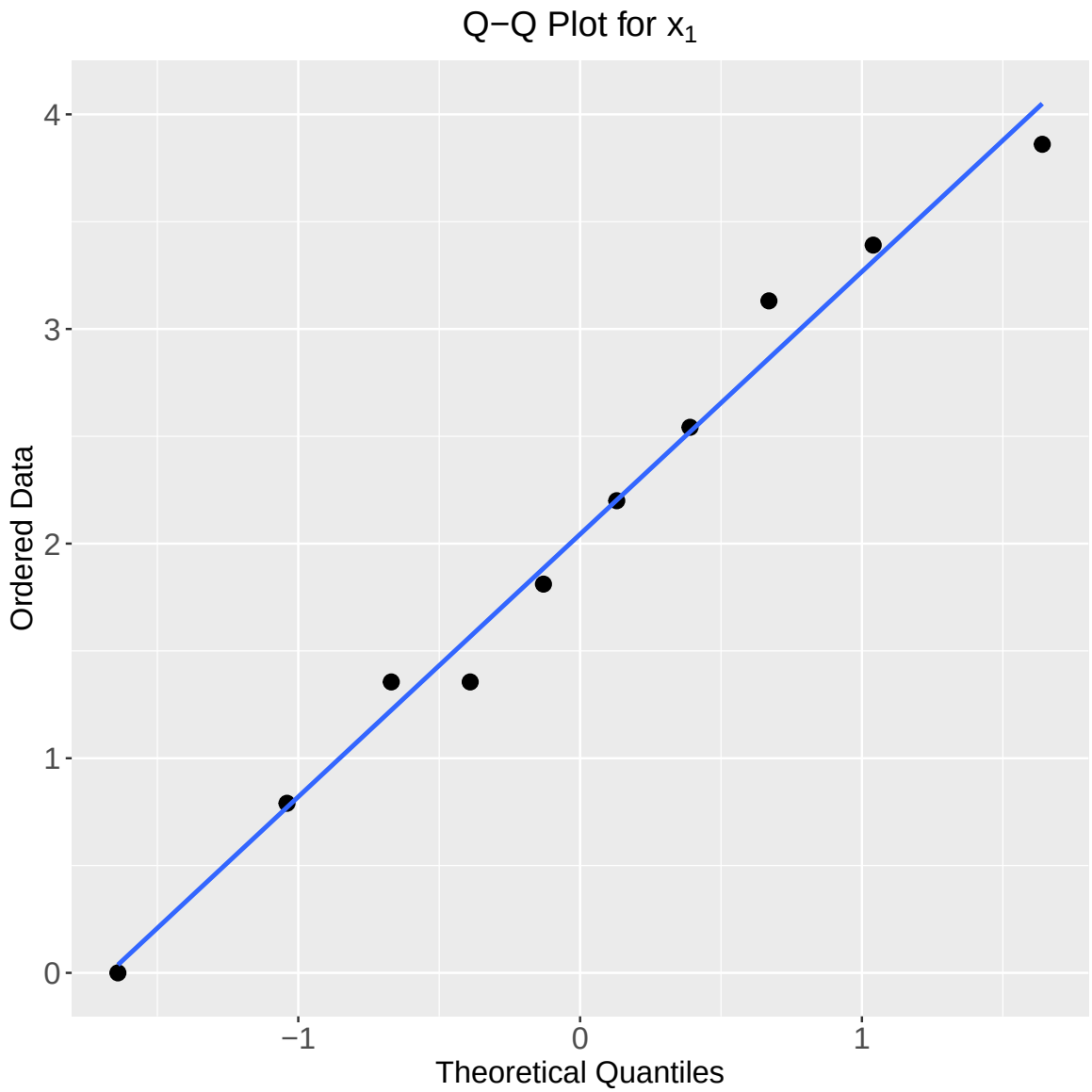
```r
#Create vector of x_1 values
x_1 <- c(1, 2, 3, 3, 4, 5, 6, 8, 9, 11)

#Find lambda value
lambda_final <- box_cox(x_1, lambda, n)

lambda_final
```

```
## [1] 0.37
```

```r
#Transform x_1 using this lambda value
transformed_x_1 <- (x_1^lambda_final - 1)/lambda_final

transformed_x_1 <- sort(transformed_x_1)

#Calculate the quantiles for the actual data
prob_levels <- ((1:n)-0.5)/n

#Calculate the theoretical normal quantiles
standard_normal_quantiles <- qnorm(prob_levels)

#Round the theoretical quantiles to 2 decimal places
standard_normal_quantiles <- round(standard_normal_quantiles,
                                   digits = 2)

#Create matrix of values
qq_matrix <- as.data.frame(cbind(transformed_x_1, prob_levels,
                                 standard_normal_quantiles))

#Print matrix
qq_matrix
```

```
##    transformed_x_1 prob_levels standard_normal_quantiles
## 1        0.0000000        0.05                     -1.64
## 2        0.7901428        0.15                     -1.04
## 3        1.3554944        0.25                     -0.67
## 4        1.3554944        0.35                     -0.39
## 5        1.8112861        0.45                     -0.13
## 6        2.1997925        0.55                      0.13
## 7        2.5419199        0.65                      0.39
## 8        3.1309634        0.75                      0.67
## 9        3.3908140        0.85                      1.04
## 10       3.8604660        0.95                      1.64
```

```
#Output plot
ggplot(qq_matrix, aes(standard_normal_quantiles, transformed_x_1)) +
  geom_point(size=3) +
  geom_smooth(method='lm', se=F) +
  xlab("Theoretical Quantiles") +
  ylab("Ordered Data") +
  ggtitle(TeX('Q-Q Plot for $x_1$')) +
  theme.info
```

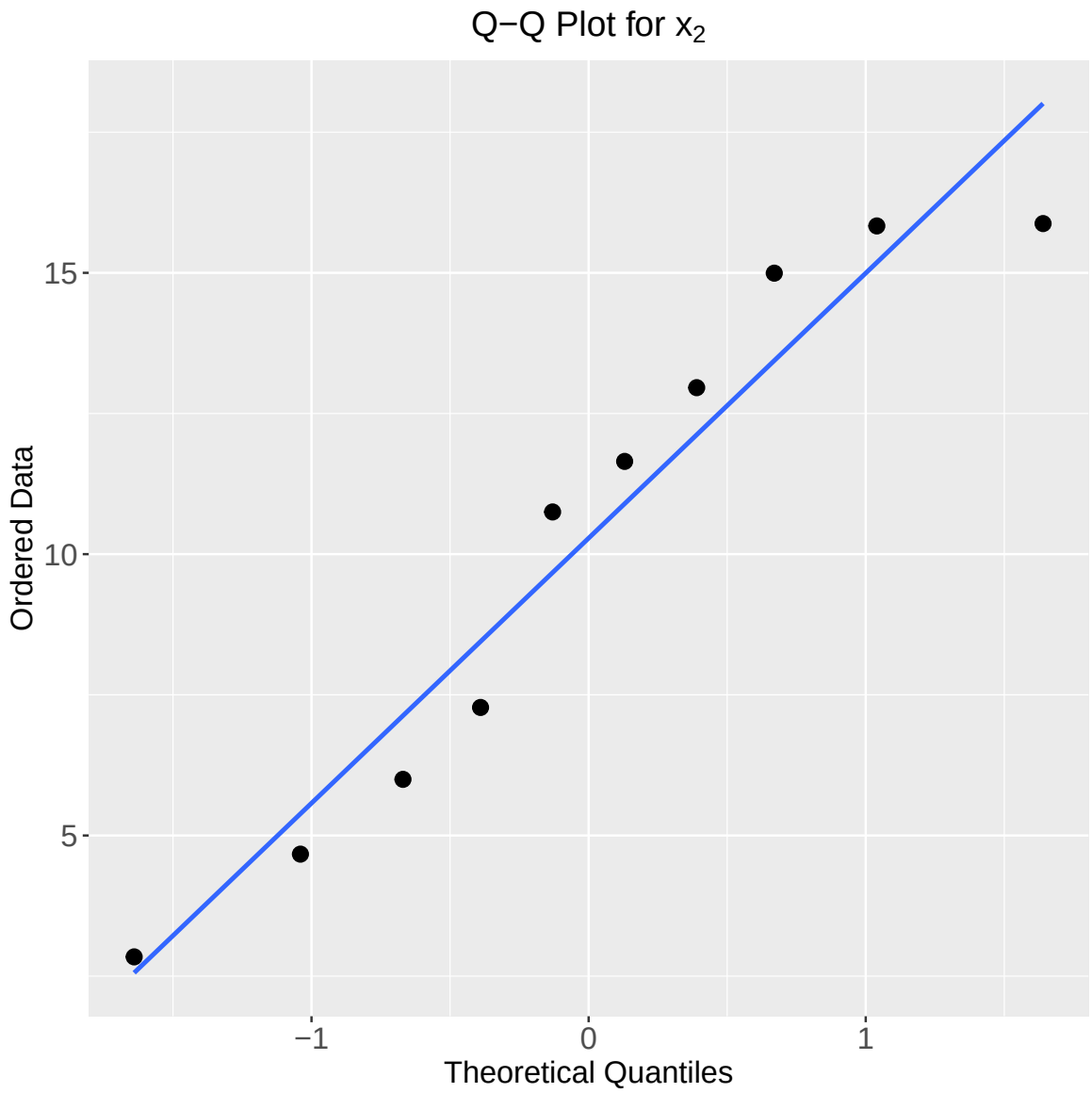


From the above, we have $\hat{\lambda}_1 = 0.37$. As we can see from the Q-Q plot, this transformation has caused the data to be distributed approximately normally.

b) Now we will find the power transformation $\hat{\lambda}_2$ that makes the x_2 values approximately normal.

```r
#Create vector of x_2 values
x_2 <- c(18.95, 19.00, 17.95, 15.54,
         14.00, 12.95, 8.94, 7.49, 6.00, 3.99)

#Find lambda value
lambda_final <- box_cox(x_2, lambda, n)

lambda_final

## [1] 0.94

#Transform x_2 using this lambda value
transformed_x_2 <- (x_2^lambda_final - 1)/lambda_final

transformed_x_2 <- sort(transformed_x_2)

#Calculate the quantiles for the actual data
prob_levels <- ((1:n)-0.5)/n

#Calculate the theoretical normal quantiles
standard_normal_quantiles <- qnorm(prob_levels)

#Round the theoretical quantiles to 2 decimal places
standard_normal_quantiles <- round(standard_normal_quantiles,
                                   digits = 2)

#Create matrix of values
qq_matrix <- as.data.frame(cbind(transformed_x_2, prob_levels,
                                 standard_normal_quantiles))

#Print matrix
qq_matrix

##    transformed_x_2 prob_levels standard_normal_quantiles
## 1         2.842660        0.05                     -1.64
## 2         4.668542        0.15                     -1.04
## 3         5.997477        0.25                     -0.67
## 4         7.275467        0.35                     -0.39
## 5        10.750409        0.45                     -0.13
## 6        11.648715        0.55                      0.13
## 7        12.959014        0.65                      0.39
## 8        14.994217        0.75                      0.67
## 9        15.833762        0.85                      1.04
## 10       15.875668        0.95                      1.64
```

```r
#Output plot
ggplot(qq_matrix, aes(standard_normal_quantiles, transformed_x_2)) +
  geom_point(size=3) +
  geom_smooth(method='lm', se=F) +
  xlab("Theoretical Quantiles") +
  ylab("Ordered Data") +
  ggtitle(TeX('Q-Q Plot for $x_2$')) +
  theme.info
```



From the above, we have $\hat{\lambda}_2 = 0.94$. As we can see from the Q-Q plot, this transformation has caused the data to be distributed approximately normally.

c) We will compute $\hat{\lambda}' = \begin{bmatrix} \hat{\lambda_1} & \hat{\lambda_2} \end{bmatrix}$ over a large grid of points in order to determine the appropriate transformation to make the $[x_1, x_2]$ values jointly normal.

```r
#Length of sequence for lambda values
ng <- 400

#Size of lambda matrix
n2 <- ng^2

#Create lambda vectors
lambda_1 <- seq(0.0001, 2.00, len = ng)
lambda_2 <- seq(0.0001, 2.00, len = ng)

#Repeat lambda values
lambda_1 <- outer(lambda_1, rep(1, ng))
lambda_2 <- outer(rep(1,ng), lambda_2)

#Form grid of lambda values
lambda_matrix <- cbind(lambda_1[1:n2], lambda_2[1:n2])

#Create vector to store function values
l_of_lambda_vec <- rep(0, times = nrow(lambda_matrix))

#Create matrix of x values
x <- cbind(x_1, x_2)

#For loop for function
for(i in 1:nrow(lambda_matrix)){
  #Get lambda values from grid
  lambda_vals <- lambda_matrix[i, ]

  #Compute transformed x matrix
  transformed_x <- cbind((x[,1]^(lambda_vals[1]) - 1)/lambda_vals[1],
                         (x[,2]^(lambda_vals[2]) - 1)/lambda_vals[2])

  #Compute covariance of transformed x matrix
  S <- cov(transformed_x)

  #Compute determinant of covariance
  det_S <- det(S)

  #Calculate function value and store it in vector
  l_of_lambda_vec[i] <- (-n/2) * log(det_S) +
    (lambda_vals[1] - 1) * sum(log(x[,1])) +
    (lambda_vals[2] - 1) * sum(log(x[,2]))
}
```

```r
#Find the lambda vector that gives the max value of l of lambda
lambda_vec_final <- lambda_matrix[which.max(l_of_lambda_vec),]

print(lambda_vec_final)

## [1] 1.27321930 0.03017368
```

From the above, we have that $\hat{\lambda}' = \begin{bmatrix} 1.2732 & 0.0302 \end{bmatrix}$. Surprisingly, these values differ significantly from the values obtained in the calculations for the marginal distributions of x_1 and x_2.

2 Problems to be Described

Problem 4.28.

For this question, we would construct a Q-Q plot and compute the resulting correlation coefficient, as we did in problems 4.23 and 4.24.

Problem 4.32.

We could first construct Q-Q plots for each of the variables $X_1, ...X_6$. For any variable X_k that appears to not be normally distributed, we could find the value λ_k that maximizes the Box-Cox equation. We could then apply this λ_k transformation to the variable, construct another Q-Q plot, and assess its normality once again.

Problem 4.34.

First, construct Q-Q plots for each variable. If either appears to not be normally distributed, use the Box-Cox formula to find and apply a transformation to it, then reassess its normality through a Q-Q plot.

Next, construct a Chi-square plot to assess the bivariate normality of the data. If it does not appear to be normally distributed, use the joint Box-Cox formula to find a transformation for the joint distribution. Apply this transformation and construct another Chi-square plot in order to assess normality.

Problem 4.35.

Once again, construct Q-Q plots for each variable. If any appear to not be normally distributed, use the Box-Cox formula to find and apply a transformation to it, then reassess its normality through a Q-Q plot.

Next, construct a Chi-square plot to assess the multivariate normality of the data. If it does not appear to be normally distributed, use the joint Box-Cox formula to find a transformation for the joint distribution. Apply this transformation and construct another Chi-square plot in order to assess normality.

Problem 4.39(a)(b).

a) Construct Q-Q plots for each variable. If any appear to not be normally distributed, use the Box-Cox formula to find and apply a transformation to it, then reassess its normality through a Q-Q plot.

b) Construct a Chi-square plot to assess the multivariate normality of the data. If it does not appear to be normally distributed, use the joint Box-Cox formula to find a transformation for the joint distribution. Apply this transformation and construct another Chi-square plot in order to assess normality.

Problem 4.40.

a) Construct scatter plots for each pair of variables and calculate the generalized squared distance for each observation in order to identify outliers.

b) Use the univariate Box-Cox formula to identify a $\hat{\lambda}_1$ value. Then construct a Q-Q plot.

c) Use the univariate Box-Cox formula to identify a $\hat{\lambda}_2$ value. Then construct a Q-Q plot.

d) Use the joint Box-Cox formula to identify a $\hat{\lambda}$ value.

Problem 4.41.

a) Construct scatter plots for each pair of variables and calculate the generalized squared distance for each observation in order to identify outliers.

b) Use the univariate Box-Cox formula to identify a $\hat{\lambda}_1$ value. Then construct a Q-Q plot.

c) Use the univariate Box-Cox formula to identify a $\hat{\lambda}_2$ value. Then construct a Q-Q plot.

d) Use the joint Box-Cox formula to identify a $\hat{\lambda}$ value.