Modules 13-14 Assignment

JARED YU

1. Use any math/stat software (e.g., www.numbergenerator.org/randomnumbergenerator) of your choice to find a random number generator to randomly select only 5 rows of Table shown in Problem 12.8. Do Problem 12.8 (a), (b), (c), (d), page 417 of Textbook, using your generated data. Make sure that you detail every intermediate step to get your results and highlight the numbers in your computer output.
   a. Fit the nonlinear regression model
$$y = \theta_1 e^{\theta_2 x} + \varepsilon$$
   to these data. Discuss how you obtained the starting values.

Ans:
The textbook explains that we can do parameter estimation in a nonlinear system such as this by using the Taylor series expansion. On pp. 400-401, it details a generalized format for linearizing a nonlinear equation and then updating the parameters from an initial parameter value iteratively. First, it is required to find the partial derivatives of our equation,
$$y_i = f(x_i, \boldsymbol{\theta}) = \theta_1 e^{\theta_2 x_i} + \varepsilon_i$$
These are relatively simple and also given by the textbook. They can be written as follows,
$$\frac{\partial f}{\partial \theta_1} = e^{\theta_2 x} \text{ and } \frac{\partial f}{\partial \theta_2} = \theta_1 x e^{\theta_2 x}$$
The linearized regression model can be written as
$$\mathbf{y}_0 = \mathbf{Z}_0 \boldsymbol{\beta}_0 + \boldsymbol{\varepsilon}$$
In the above equation, $\mathbf{Z}_0$ is a matrix of partial derivatives being applied to each of the sample data points. Here, $\boldsymbol{\beta}_0$ is estimated with
$$\widehat{\boldsymbol{\beta}}_0 = (\mathbf{Z}_0'\mathbf{Z}_0)^{-1}\mathbf{Z}_0'\mathbf{y}_0 = (\mathbf{Z}_0'\mathbf{Z}_0)^{-1}\mathbf{Z}_0'(\mathbf{y} - \mathbf{f}_0)$$
By definition, $\boldsymbol{\beta}_0 = \boldsymbol{\theta} - \boldsymbol{\theta}_0$, and so we can also perform an update so that
$$\widehat{\boldsymbol{\theta}}_1 = \widehat{\boldsymbol{\beta}}_0 + \boldsymbol{\theta}_0$$
This update leads to the following generalize update rule for the kth iteration
$$\widehat{\boldsymbol{\theta}}_{k+1} = \widehat{\boldsymbol{\theta}}_k + \widehat{\boldsymbol{\beta}}_0 = \widehat{\boldsymbol{\theta}}_k + (\mathbf{Z}_k'\mathbf{Z}_k)^{-1}\mathbf{Z}_k'(\mathbf{y} - \mathbf{f}_k)$$

Looking closely, this can be programmed fairly easily after initializing a set of starting values. By plotting the randomly select points on a graph, a line was drawn on top of it which follows the $y = \theta_1 e^{\theta_2 x}$ formula. Simply by testing random values, it became apparent that $\theta_1 = 1$ and $\theta_2 = 0.5$ provided a very solid fit to the few data points available.

So, given that the initial vector $\boldsymbol{\theta}_0 = (\theta_1, \theta_2)^\top = (1, 0.5)^\top$, then we plug in these values to the partial derivatives and apply these to the sample data to get $\mathbf{Z}_0$. After getting $\mathbf{Z}_0$, the initialized $\widehat{\boldsymbol{\beta}}_0$ can be calculated. Then, we can get the initial $k = 1$ update value $\widehat{\boldsymbol{\theta}}_1 = \widehat{\boldsymbol{\beta}}_0 + \boldsymbol{\theta}_0$.

At this point, it'd be good to quickly check the cutoff value, $\left[\frac{\hat{\theta}_{j,k+1} - \hat{\theta}_{jk}}{\hat{\theta}_{jk}}\right] < \delta, j = 1,2$. Using $\delta = 1 \times 10^{-6}$, we can allow some cutoff variable to evaluate to true if all $\hat{\theta}_j$ terms are less than δ within the $\left[\frac{\hat{\theta}_{j,k+1} - \hat{\theta}_{jk}}{\hat{\theta}_{jk}}\right]$ ratio. In the initialization of the variables, the cutoff still evaluates to false.

From this stage, we can begin to do a looping process, where the while-loop will continue until the cutoff evaluates to true. The looping stage involves recalculating $\mathbf{Z}_k$, by applying the $\widehat{\boldsymbol{\theta}}_k$ terms to the partial derivatives and then reapplying these updated partial derivative functions to the sample data. Then, the $\widehat{\boldsymbol{\theta}}_{k+1} = \widehat{\boldsymbol{\theta}}_k + (\mathbf{Z}_k'\mathbf{Z}_k)^{-1}\mathbf{Z}_k'(\mathbf{y} - \mathbf{f}_k)$ can be applied, where an important difference now is that we are calculating $(\mathbf{y} - \mathbf{f}_k)$, rather than just $\mathbf{y}$ as with the initialization stage. Here, $\mathbf{f}_k$ is the vector that results from applying the $\theta_1 e^{\theta_2 x}$ function to the sample data where the θ_j terms are substituted with the current k'th estimates. A useful programming trick would be to create a theta_old and theta_new type of variable. The theta_old can store the current (which becomes the previous in the next iteration) $\widehat{\boldsymbol{\theta}}_k$ and save to theta_new the $\widehat{\boldsymbol{\theta}}_{k+1}$ term. Using these two variables, the cutoff variable can then be calculated at the end of the while-loop to determine if the threshold has been reached. The while-loop will check this cutoff variable each time. A last note that the textbook mentions is that it's worthwhile to calculate the residual sum of squares (RSS) at each iteration. The result of this process can be seen below:

Table 1 The below table shows the values of the θ_1 and θ_2 estimates at each iteration along with the corresponding RSS value.

k'th Iteration	θ_1, θ_2 estimates	RSS
0	1; 0.5	33,750.62
1	-0.5648; 0.002	6.6747
2	0.7637; 20.5265	2.685
3	0.7637; 20.5265	2.6848

It is worth noting that since the process completed relatively quickly, there is more confidence in the result. At the same time, this could be biased since it's based on a limited sample size. The plot below shows the difference in the line generated by the initial estimates (black) and the updated estimates (red).

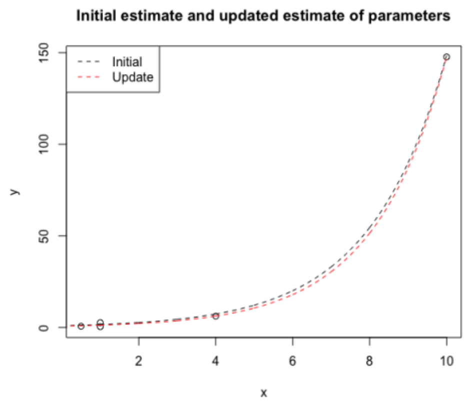

Figure 1 The above plot shows the initial estimate and updated estimate of the true line plotted over the sample data.

b. Test for significance of regression.

<u>Ans:</u>

Here, the test for significance of regression tests the following:

$$H_0: \theta_1 = \theta_2 = 0 \text{ vs. } H_1: \theta_j \neq 0 \text{ for at least one } j$$

The test statistic here is $F_0 = \frac{\frac{SS_R}{k}}{\frac{SS_{Res}}{n-k-1}} = \frac{MS_R}{MS_{Res}}$ which follows a $F_{k,n-k-1}$ distribution. The resulting test statistic is $F_0 = \frac{8{,}443.033}{0.8949} = 9{,}434.305$ which is significantly larger than the corresponding critical value of 9.5521. Therefore, we reject the null hypothesis in favor of the alternative at the 95% confidence level.

c. Estimate the error variance σ^2.

Ans:
We can estimate σ^2 by using the following formula,

$$\hat{\sigma} = MS_{Res} = \frac{\sum_{i=1}^{5}(y_i - \hat{y}_i)^2}{5-2}$$

The above is simply the SS_{Res} (RSS) calculated earlier. So dividing that value by the degrees of freedom ($n - p = 5 - 2 = 3$) leads to $\boxed{\hat{\sigma} = 0.8949}$. *Note: This was calculated previously in part b.*

d. Test the hypotheses $H_0: \theta_1 = 0$ and $H_0: \theta_2 = 0$. Are both model parameters different from zero? If not, refit an appropriate model.

Ans:
To test the hypotheses $H_0: \theta_1 = 0$ and $H_0: \theta_2 = 0$, it requires calculating the following test statistic,

$$t_0 = \frac{\hat{\theta}_j}{\sqrt{\hat{\sigma}^2 C_{jj}}}$$

where C_{jj} is the diagonal j'th element from $(\mathbf{Z}_k'\mathbf{Z}_k)^{-1}$ (given that k is the final iteration). The corresponding critical value is $t_{\frac{\alpha}{2},n-k-1} = 3.1824$. The resulting test statistics are 4.4552 and 23.4363 for θ_1 and θ_2 respectively. Therefore, in both cases we reject the null hypotheses that $H_0: \theta_1 = 0$ and $H_0: \theta_2 = 0$ in favor of the alternative that they both are nonzero at the 95% confidence level.

e. Analyze the residuals from this model. Discuss model adequacy.

Ans:
We can first look at the normal probability plot of the sorted residuals, shown below in Figure 2. Ideally, we would want to see them plotted fairly proportionately around the line drawn through them. An issue however seems to be the limited amount of data. In other words, drawing a line through such few points makes it difficult to gauge if there is any strange behavior, or even what normal behavior looks like. However, given what's available, it doesn't seem like there's significant violation such as light or heavy tails. Again, this is based on very few points so it's difficult to say what's "good" or "bad."

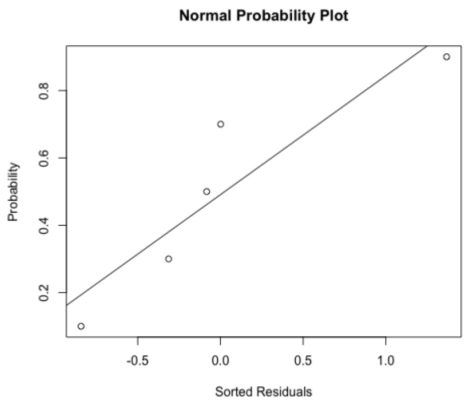

Figure 2 The above plot shows the sorted residuals plotted on a normal probability plot.

The next plot shows the issue with this limited sample size. The plot of the residuals against the predicted response can be seen below in Figure 3. Looking at this plot, it's quite evident that there's a single "outlier," or at least one point that's far away from the others. There's already knowledge about two things, one being that this is a random sample of only 5 points from the original and the other being that it's a nonlinear problem with some sort of exponential type of relationship.

The fact that there are only 5 random points makes it difficult to see what the general behavior could be for the entire population. Although the plot appears to be showing an outliers, it's not at all certain whether it really is, since there's a good chance that there are many points in the middle that aren't shown due to the random sample of 5 points. Furthermore, in an exponential type of plot, we know that the values start small and increase dramatically. Therefore, the pattern seen on the x-axis of having one extremely large value isn't at all strange. A similar problem exists here also as in with Figure 2. There are too few points, and although the residual plot may seem slightly strange, there's not enough evidence to say it's "good" or "bad," since the understanding of the limited sample size and exponential data makes it difficult to criticize the results.

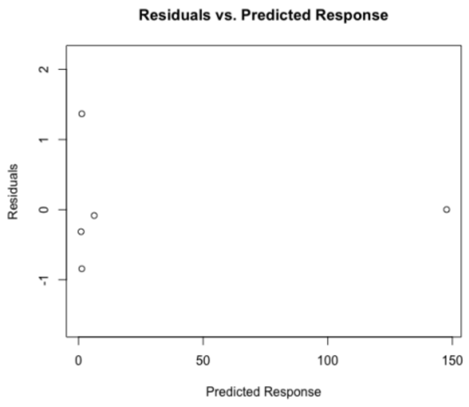

Figure 3 The above plot shows the residuals plotted against the predicted response.

2. Use any math/stat software (e.g., www.numbergenerator.org/randomnumbergenerator) of your choice to find a random number generator to randomly select only 14 rows of Table shown in Problem 12.11. Do Problem 12.11 (a), (b), (c), (d), (e), page 418 of Textbook, using your generated data. Make sure that you detail every intermediate step to get your results and highlight the numbers in your computer output.
   a. Construct a scatterplot of the data.

Ans:
Below is a scatterplot of the data seen in Figure 4.

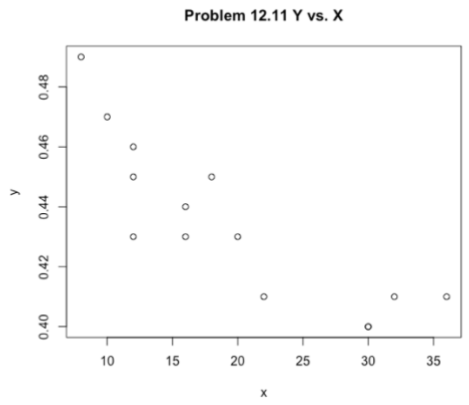

Figure 4

   b. Fit the Mitcherlich law (see Problem 12.10) to these data. Discuss how you obtained the starting values.

Ans:
One of the first steps is to calculate the partial derivatives. The original model is as follows,

$$y = \theta_1 - \theta_2 e^{-\theta_3 x} + \varepsilon$$

The partial derivatives of this model can be seen as follows,

$$\frac{\partial f}{\partial \theta_1} = 1; \frac{\partial f}{\partial \theta_2} = -e^{-\theta_3 x}; \frac{\partial f}{\partial \theta_3} = \theta_2 x e^{-\theta_3 x}$$

(Note: Much of below is the same as problem 1.)
The linearized regression model can be written as

$$\mathbf{y}_0 = \mathbf{Z}_0 \boldsymbol{\beta}_0 + \boldsymbol{\varepsilon}$$

In the above equation, $\mathbf{Z}_0$ is a matrix of partial derivatives being applied to each of the sample data points. Here, $\boldsymbol{\beta}_0$ is estimated with

$$\widehat{\boldsymbol{\beta}}_0 = (\mathbf{Z}_0'\mathbf{Z}_0)^{-1}\mathbf{Z}_0'\mathbf{y}_0 = (\mathbf{Z}_0'\mathbf{Z}_0)^{-1}\mathbf{Z}_0'(\mathbf{y} - \mathbf{f}_0)$$

By definition, $\boldsymbol{\beta}_0 = \boldsymbol{\theta} - \boldsymbol{\theta}_0$, and so we can also perform an update so that

$$\widehat{\boldsymbol{\theta}}_1 = \widehat{\boldsymbol{\beta}}_0 + \boldsymbol{\theta}_0$$

This update leads to the following generalize update rule for the kth iteration

$$\widehat{\boldsymbol{\theta}}_{k+1} = \widehat{\boldsymbol{\theta}}_k + \widehat{\boldsymbol{\beta}}_0 = \widehat{\boldsymbol{\theta}}_k + (\mathbf{Z}_k'\mathbf{Z}_k)^{-1}\mathbf{Z}_k'(\mathbf{y} - \mathbf{f}_k)$$

The more difficult part this time was finding a set of suitable values. The reason is that this time there are three variables to initialize rather than two. Using the same methodology of plotting and checking how well it fits the sample data worked but was more time consuming. Playing around with the numbers resulted in the following initial vector which were felt to be satisfactory, $\boldsymbol{\theta}_0 =$

$(\theta_1, \theta_2, \theta_3)^\top = (0.5, -0.3, 0.15)^\top$. So, given that the initial vector $\boldsymbol{\theta}_0$, then we can plug in these values to the partial derivatives and apply these to the sample data to get $\mathbf{Z}_0$. After getting $\mathbf{Z}_0$, the initialized $\widehat{\boldsymbol{\beta}}_0$ can be calculated. Then, we can get the initial $k = 1$ update value $\widehat{\boldsymbol{\theta}}_1 = \widehat{\boldsymbol{\beta}}_0 + \boldsymbol{\theta}_0$. The same process of also finding a cutoff variable was used as before, with the same $\delta = 1 \times 10^{-6}$.

The below table shows the results from running the while-loop

Table 23 The below table shows the values of the θ_1, θ_2, and θ_3 estimates at each iteration along with the corresponding RSS value.

k'th Iteration	$\theta_1, \theta_2, \theta_3$ estimates	RSS
0	0.5; -0.3; 0.15	4.1433
1	0.4011; -0.2046; -0.2046	0.0015

Surprisingly, the while-loop ended after a single iteration, where the correspond RSS was quite small after one try.

The following plot shows the line corresponding to the initial vector and the updated vector.

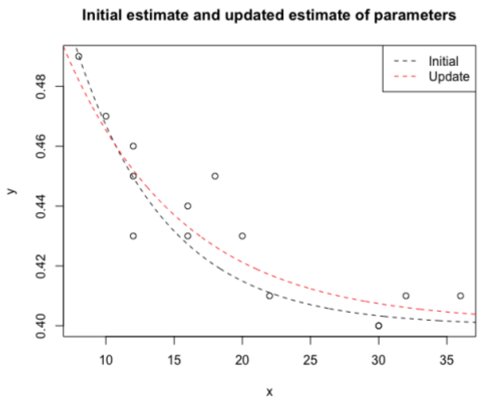

Figure 5 The above plot shows the initial estimate and updated estimate of the true line plotted over the sample data.

c. Test for significance of regression.

Ans:

Here, the test for significance of regression tests the following:

$$H_0\colon \theta_1 = \theta_2 = \theta_3 = 0 \text{ vs.} H_1\colon \theta_j \neq 0 \text{ for at least one } j$$

The test statistic here is $F_0 = \frac{\frac{SS_R}{k}}{\frac{SS_{Res}}{n-k-1}} = \frac{MS_R}{MS_{Res}}$ which follows a $F_{k,n-k-1}$ distribution. The resulting test statistic is $F_0 = \frac{0.0027}{0.0001} = 19.6438$ which is significantly larger than the corresponding critical value of 3.5874. Therefore, we reject the null hypothesis in favor of the alternative at the 95% confidence level.

d. Find approximate 95% confidence intervals on the parameters θ_1, θ_2, and θ_3. Is there evidence to support the claim that all three parameters are different from zero?

Ans:
To find the confidence intervals (C.I.'s), we first need to find the standard errors. They can be found using the same method for the denominator of the test statistic from the previous problem, for the individual coefficient hypothesis test. These are $\sqrt{\hat{\sigma}^2 C_{jj}}$ where C_{jj} is the diagonal j'th element from $(\mathbf{Z}_k'\mathbf{Z}_k)^{-1}$. $\hat{\sigma}^2$ is also estimated using the same method for MS_{Res} as before, where here,

$$\hat{\sigma} = MS_{Res} = \frac{\sum_{i=1}^{5}(y_i - \hat{y}_i)^2}{14-3} \approx 0.0001$$

The corresponding critical value from the t-distribution for the C.I. is $t_{\frac{a}{2},n-p} = 2.2010$. The formula for each of the three C.I.'s then becomes,

$$\hat{\theta}_j - t_{\frac{a}{2},n-p}\sqrt{\hat{\sigma}^2 C_{jj}} \le \theta_j \le \hat{\theta}_j + t_{\frac{a}{2},n-p}\sqrt{\hat{\sigma}^2 C_{jj}}$$

The table below shows the C.I. for each of the parameters. It can be seen that none of the C.I.'s contain the value zero, and so it can be said that the 95% C.I.'s below support the claim that all three parameters are nonzero.

Table 4 The below table shows the 95% C.I. for each of the parameters.

θ_j	95% C.I.
θ_1	$(0.3795, 0.4227)$
θ_2	$(-0.3688, -0.0404)$
θ_3	$(0.0748, 0.1570)$

e. Analyze the residuals and comment on model adequacy.

Ans:
We can first look at the normal probability plot for the residuals, which is shown below in Figure 6. It can be seen that in general, the point seem to be following a line. It is worth noting however that the line itself is drawn simply by using a simple linear regression formula. So, looking at the points in general they seem to follow a fairly straight path. However, there are two points each at the edges which seem to be causing possible problems. It is unsure however if these points themselves are outliers, or it's simply due to the few data points available. We can also see that the pattern is not entirely ideal, since we'd like to see the value moving fairly evenly above and below the line. Once again, it's difficult to make a conclusive decision since with so few points, it's difficult to gauge if the movement about the line is extreme or "normal." However, the data is in fact non-linear, so it'd make sense to see some sort of behavior that is a bit different from what's expected in a more ideal case.

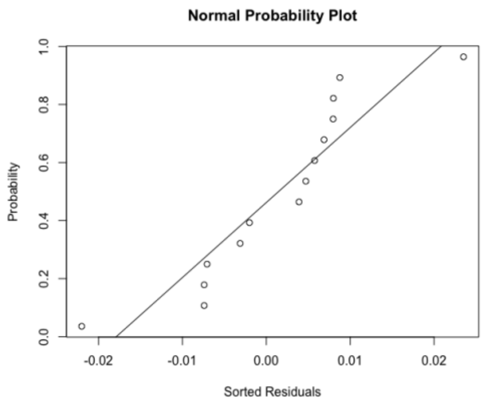

Figure 6 The above plot shows the sorted residuals plotted on a normal probability plot.

The below plot in Figure 7 shows the residuals plotted against the predicted response. This plot looks quite good, as there are points scattered about the x-axis fairly evenly, and there is for the most part a horizontal band containing all the points. There's evidently one point slightly higher than the rest, and one point slightly lower than the rest. It's difficult to determine for sure if these are outliers, but it's likely that with more data that these points would appear less extreme. Also, there seems to be a slightly higher density towards the left than towards the right. However, looking back to the original plot of the data, given the pattern of the scatterplot it makes sense that there's a greater density to the left than to the right.

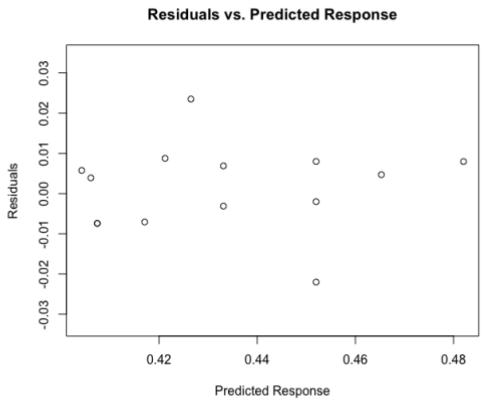

Figure 7 The above plot shows the residuals plotted against the predicted response.

It's interesting to see then that in this problem with more points, the "odd" patterns in the residuals seem to reflect the fact that the original data follows some nonlinear pattern associated with some exponential function.

Code Appendix

```
### Problem 1
df <- MPV::p12.8
n <- 5
set.seed(1); chosen_rows <- sort(sample(seq(1, nrow(df)), n))
df <- df[chosen_rows,]

# part (a)
plot(df$x, df$y)
f_0 <- function(x, theta1, theta2) {
  return(theta1 * exp(theta2 * x))
}
x_seq <- seq(0, 10, length.out = 1e3)
lines(x_seq, f_0(x = x_seq, theta1 = 1, theta2 = 0.5))

# Functions start
theta1_deriv_calc <- function(x, theta2) {
  return(exp(theta2 * x))
}
theta2_deriv_calc <- function(x, theta1, theta2) {
  return(theta1 * x * exp(theta2 * x))
}
cutoff_fun <- function(theta_new, theta_old, delta) {
  cutoff_calc <- (theta_new - theta_old)  / theta_hat_old

  # Cutoff when both less than delta
  if (cutoff_calc[1] < delta & cutoff_calc[2] < delta) {
    return(TRUE)
  } else {
    return(c(FALSE, round(cutoff_calc[1], 4), round(cutoff_calc[2], 4)))
  }
}
beta_hat_calc <- function(x, y, Z, theta1, theta2) {
  return(solve(t(Z) %*% Z) %*% t(Z) %*%
           (y - f_0(x = x, theta1 = theta1, theta2 = theta2)))
}
RSS_calc <- function(y, y_hat) {
  return(sum((y_hat - y)^2))
}
# Functions end

# Initialize theta-0
theta1 <- 1; theta2 <- 0.5
theta_vec <- c(theta1, theta2)

# Initialize derivatives, Z
theta1_deriv <- theta1_deriv_calc(x = df$x, theta2 = theta_vec[2])
theta2_deriv <- theta2_deriv_calc(x = df$x,
  theta1 = theta_vec[1], theta2 = theta_vec[2])
Z <- cbind(theta1_deriv, theta2_deriv)

# Initialize beta-hat
beta_hat <- solve(t(Z) %*% Z) %*% t(Z) %*% df$y

# Generate theta-hat-1
theta_hat_update <- beta_hat + theta_vec
theta_hat_old <- theta_vec

# Initialize cutoff
delta <- 1e-6
cutoff <- cutoff_fun(theta_new = theta_hat_update,
                     theta_old = theta_hat_old,
                     delta = delta)

# Initial RSS
y_hat <- f_0(x = df$x,
             theta1 = theta_hat_update[1], theta2 = theta_hat_update[2])
RSS <- RSS_calc(y = df$y, y_hat = y_hat)
```

```r
counter <- 0
while(!cutoff[1]) {
  # Update counter
  counter <- counter + 1
  print(paste0('Counter: ', counter))
  # Calculate Z
  theta1_deriv <- theta1_deriv_calc(x = df$x, theta2 = theta_hat_update[2])
  theta2_deriv <- theta2_deriv_calc(x = df$x,
    theta1 = theta_hat_update[1], theta2 = theta_hat_update[2])
  Z <- cbind(theta1_deriv, theta2_deriv)

  # Calculate beta-hat
  beta_hat <- beta_hat_calc(x = df$x, y = df$y, Z = Z,
    theta1 = theta_hat_update[1], theta2 = theta_hat_update[2])

  # theta_{k+1} = theta_k + beta_k
  theta_hat_old <- theta_hat_update
  theta_hat_update <- beta_hat + theta_hat_update

  # Calculate cutoff
  cutoff <- cutoff_fun(theta_new = theta_hat_update, theta_old = theta_hat_old, delta = delta)
  print(paste0('Cutoff:', cutoff[2], '; ', cutoff[3]))

  # Print update values
  print(paste0('theta1:',
               round(theta_hat_update[1], 4),
               '; theta2',
               round(theta_hat_update[2], 4)))

  # Print RSS
  y_hat <- f_0(x = df$x,
    theta1 = theta_hat_update[1], theta2 = theta_hat_update[2])
  RSS <- RSS_calc(y = df$y, y_hat = y_hat)
  print(paste0('RSS:', round(RSS, 4)))
}

plot(df$x, df$y,
     main = 'Initial estimate and updated estimate of parameters',
     xlab = 'x', ylab = 'y')
legend("topleft", legend = c('Initial', 'Update'),
       lty = c(2,2), col = c('black', 'red'))
lines(x_seq, f_0(x = x_seq, theta1 = 1, theta2 = 0.5), lty = 2)
lines(x_seq, f_0(x = x_seq,
  theta1 = theta_hat_update[1],
  theta2 = theta_hat_update[2]),
  col = 'red', lty = 2)

# part (b)
SS_T <- t(df$y) %*% df$y - ((sum(df$y))^2) / n
SS_model <- SS_T - RSS
MS_Res <- RSS / (n - 2)

F_0 <- (SS_model / 2) / MS_Res
qf(0.95, df1 = 2, df2 = 3)

# part (c)
cov_mat <- MS_Res * solve(t(Z) %*% Z)
sqrt(diag(cov_mat))

# part (d)
C <- solve(t(Z) %*% Z)
standard_errors <- sqrt(diag(MS_Res * C))
t_test_statics <- theta_hat_update / standard_errors
alpha <- 0.05
qt(1 - alpha/2, df = 3)

# part (e)
```

```r
e <- df$y - y_hat
beta_hat_calc <- function(X, y) {
  X <- as.matrix(X)
  beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y
  return(beta_hat)
}
norm_prob_plot <- function(residual_var, x_label,
                           main_title = 'Normal Probability Plot',
                           y_label = 'Probability', n_size=n) {
  ones <- rep(1, n)
  sorted_residuals <- sort(residual_var)
  cumulative_probability <- (1:n_size - 0.5) / n_size
  plot(sorted_residuals, cumulative_probability, main = main_title,
       xlab = x_label,
       ylab = y_label)
  X_temp <- cbind(ones, sorted_residuals)
  beta_hat_temp <- beta_hat_calc(X=X_temp,y=cumulative_probability)
  abline(beta_hat_temp)
}
norm_prob_plot(residual_var = e, x_label = 'Sorted Residuals')
order(e, decreasing = FALSE)
e[order(e, decreasing = FALSE)]

res_vs_fitted_plot <- function(residual_var,
                               main_title,
                               y_label,
                               x_label = 'Predicted Response',
                               pred_response = y_hat) {
  plot(pred_response, residual_var, main = main_title,
       xlab = x_label,
       ylab = y_label,
       ylim = c(min(residual_var)-sd(residual_var),
                max(residual_var)+sd(residual_var)))
}
res_vs_fitted_plot(residual_var = e,
                   main_title = 'Residuals vs. Predicted Response',
                   y_label = 'Residuals')


### Problem 2
df <- MPV::p12.11
n <- 14
set.seed(1); chosen_rows <- sort(sample(seq(1, nrow(df)), n))
df <- df[chosen_rows,]

# part (a)
plot(df$x, df$y, main = 'Problem 12.11 Y vs. X', xlab = 'x', ylab = 'y')

# part (b)
f_0 <- function(x, theta1, theta2, theta3) {
  return(theta1 - theta2 * exp(- theta3 * x))
}

x_seq <- seq(0, 40, length.out = 1e3)
lines(x_seq, f_0(x = x_seq, theta1 = 0.4, theta2 = -0.3, theta3 = 0.15))

# Functions start
theta1_deriv_calc <- function(x) {
  return(rep(1, length(x)))
}
theta2_deriv_calc <- function(x, theta3) {
  return(-exp(-theta3 * x))
}
theta3_deriv_calc <- function(x, theta2, theta3) {
  return(theta2 * x * exp(-theta3 * x))
}
cutoff_fun <- function(theta_new, theta_old, delta) {
  cutoff_calc <- (theta_new - theta_old)  / theta_hat_old
  # Cutoff when both less than delta
  if (cutoff_calc[1] < delta &
      cutoff_calc[2] < delta &
      cutoff_calc[3] < delta) {
    return(TRUE)
  } else {
    return(c(FALSE,
             round(cutoff_calc[1], 4),
             round(cutoff_calc[2], 4),
             round(cutoff_calc[3], 4)))
  }
}
beta_hat_calc <- function(x, y, Z, theta1, theta2, theta3) {
  return(solve(t(Z) %*% Z) %*% t(Z) %*%
           (y - f_0(x = x, theta1 = theta1, theta2 = theta2, theta3 = theta3)))
}
RSS_calc <- function(y, y_hat) {
  return(sum((y_hat - y)^2))
}
# Functions end

# Initialize theta-0
theta1 <- 0.5; theta2 <- -0.3; theta3 <- 0.15
theta_vec <- c(theta1, theta2, theta3)

# Initialize derivatives, Z
theta1_deriv <- theta1_deriv_calc(x = df$x)
theta2_deriv <- theta2_deriv_calc(x = df$x, theta3 = theta_vec[3])
theta3_deriv <- theta3_deriv_calc(x = df$x,
                                  theta2 = theta_vec[2], theta3 = theta_vec[3])

Z <- cbind(theta1_deriv, theta2_deriv, theta3_deriv)

# Initialize beta-hat
beta_hat <- solve(t(Z) %*% Z) %*% t(Z) %*% df$y

# Generate theta-hat-1
theta_hat_update <- beta_hat + theta_vec
theta_hat_old <- theta_vec

# Initialize cutoff
delta <- 1e-6
cutoff <- cutoff_fun(theta_new = theta_hat_update,
                     theta_old = theta_hat_old,
                     delta = delta)

# Initial RSS
y_hat <- f_0(x = df$x,
             theta1 = theta_hat_update[1],
             theta2 = theta_hat_update[2],
             theta3 = theta_hat_update[3])
RSS <- RSS_calc(y = df$y, y_hat = y_hat)

# Run the while-loop
counter <- 0
while(!cutoff[1]) {
  # Update counter
  counter <- counter + 1
  print(paste0('Counter: ', counter))
  # Calculate Z
  theta1_deriv <- theta1_deriv_calc(x = df$x)
  theta2_deriv <- theta2_deriv_calc(x = df$x, theta3 = theta_hat_update[3])
  theta3_deriv <- theta3_deriv_calc(x = df$x,
                                    theta2 = theta_hat_update[2], theta3 = theta_hat_update[3])
  Z <- cbind(theta1_deriv, theta2_deriv, theta3_deriv)

  # Calculate beta-hat
```

```r
  beta_hat <- beta_hat_calc(x = df$x, y = df$y, Z = Z,
                            theta1 = theta_hat_update[1],
                            theta2 = theta_hat_update[2],
                            theta3 = theta_hat_update[3])

  # theta_{k+1} = theta_k + beta_k
  theta_hat_old <- theta_hat_update
  theta_hat_update <- beta_hat + theta_hat_update

  # Calculate cutoff
  cutoff <- cutoff_fun(theta_new = theta_hat_update, theta_old = theta_hat_old, delta = delta)
  print(paste0('Cutoff:', cutoff[2], '; ', cutoff[3], '; ', cutoff[3]))

  # Print update values
  print(paste0('theta1:', round(theta_hat_update[1], 4),
               '; theta2', round(theta_hat_update[2], 4),
               '; theta2', round(theta_hat_update[2], 4)))

  # Print RSS
  y_hat <- f_0(x = df$x,
               theta1 = theta_hat_update[1],
               theta2 = theta_hat_update[2],
               theta3 = theta_hat_update[3])
  RSS <- RSS_calc(y = df$y, y_hat = y_hat)
  print(paste0('RSS:', round(RSS, 4)))
}

plot(df$x, df$y,
     main = 'Initial estimate and updated estimate of parameters',
     xlab = 'x', ylab = 'y')
legend("topright", legend = c('Initial', 'Update'),
       lty = c(2,2), col = c('black', 'red'))
x_seq <- seq(0, 40, length.out = 1e3)
lines(x_seq, f_0(x = x_seq, theta1 = 0.4, theta2 = -0.3, theta3 = 0.15), lty = 2)
lines(x_seq, f_0(x = x_seq,
                 theta1 = theta_hat_update[1],
                 theta2 = theta_hat_update[2],
                 theta3 = theta_hat_update[3]), lty = 2, col = 'red')

# part (c)
SS_T <- t(df$y) %*% df$y - ((sum(df$y))^2) / n
SS_model <- SS_T - RSS
MS_Res <- RSS / (n - 3)

F_0 <- (SS_model / 3) / MS_Res
qf(0.95, df1 = 3, df2 = n - 3)

# part (d)
C <- solve(t(Z) %*% Z)
standard_errors <- sqrt(diag(MS_Res * C))
alpha <- 0.05
t_value <- qt(1 - alpha / 2, n - 3)
round(theta_hat_update + t_value * standard_errors, 4)
round(theta_hat_update - t_value * standard_errors, 4)

# part (e)
e <- df$y - y_hat
beta_hat_calc <- function(X, y) {
  X <- as.matrix(X)
  beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y
  return(beta_hat)
}
norm_prob_plot <- function(residual_var, x_label,
                           main_title = 'Normal Probability Plot',
                           y_label = 'Probability', n_size=n) {
  ones <- rep(1, n)
  sorted_residuals <- sort(residual_var)
  cumulative_probability <- (1:n_size - 0.5) / n_size
  plot(sorted_residuals, cumulative_probability, main = main_title,
       xlab = x_label,
       ylab = y_label)
  X_temp <- cbind(ones, sorted_residuals)
  beta_hat_temp <- beta_hat_calc(X=X_temp,y=cumulative_probability)
  abline(beta_hat_temp)
}
norm_prob_plot(residual_var = e, x_label = 'Sorted Residuals')
order(e, decreasing = FALSE)
e[order(e, decreasing = FALSE)]

res_vs_fitted_plot <- function(residual_var,
                               main_title,
                               y_label,
                               x_label = 'Predicted Response',
                               pred_response = y_hat) {
  plot(pred_response, residual_var, main = main_title,
       xlab = x_label,
       ylab = y_label,
       ylim = c(min(residual_var)-sd(residual_var),
                max(residual_var)+sd(residual_var)))
}
res_vs_fitted_plot(residual_var = e,
                   main_title = 'Residuals vs. Predicted Response',
                   y_label = 'Residuals')
```