Implementing a custom Linear Regression function

Theodore Li

10/30/2024

Part 1.)

Creating custom mylm() function using matrix multiplication

```
library(ggplot2)

mylm <- function(y, x, method = "qr", interval = "confidence", level = 0.95, plot = TRUE){
  #Compute least squares coefficient using matrix algebra

  #Placeholder values for Qmat and Rmat if we don't use qr decomposition
  Qmat <- 0
  Rmat <- 0
  #Combines a column of 1s used as a intercept term
  x <- cbind(1, x)
  xtx_inv <- solve(t(x) %*% x)  # Compute (X^T X)^(-1)
  if (method == "inverse") {
    # Solve using the normal equation: betahat = (X^T X)^(-1) X^T Y
    betahat <- xtx_inv %*% t(x) %*% y  # Compute beta
  }

  # Solve using QR decomposition
  else if (method == "qr") {
    QR_decomp <- qr(x)  # QR decomposition of X
    Qmat <- qr.Q(QR_decomp) #Extracting the Q matrix
    Rmat <- qr.R(QR_decomp) #Extracting the R matrix

    betahat <- backsolve(Rmat, t(Qmat) %*% y) #Using backsolve to find betahat
  }

  residuals = y - x %*% betahat

  fitted.values = x %*% betahat

  #ybar is the average of the observed output
  ybar <- mean(y)


  #Calculating the variance among predicted values
  mean_fitted <- mean(fitted.values) #Average predicted value
  n <- (nrow(x)) #Number of values
  p <- (ncol(x)) #Number of predictors
  df <- n - p #degrees of freedom
  sigmahat <- sum((fitted.values - mean_fitted)^2) / df

# ~~~~~~~~~~~~~ Creating a Estimate table
```

```r
  SE_betahat <- sqrt(sigmahat * diag(xtx_inv))
  #SE_betahat <- sigmahat * sqrt(xtx_inv)


  #t -statistics for betahat

  t_statistics <- betahat / SE_betahat

  # p values for a two tailed test
  p_values <- 2 * pt(-abs(t_statistics), df = n - p)

  #Create a searchable Estimate table to display the previous calculated values
  EstimateTab <- data.frame(
    Label = paste0("beta_", 0:(length(betahat) - 1)),  # Add the Label first
    Coefficient = as.vector(betahat),
    StdError = as.vector(SE_betahat),
    tStatistic = as.vector(t_statistics),
    pValue = as.vector(p_values)
  )


# ~~~~~~~~~~~~~~ Calculating CI/PI of fitted values
  alpha <- 1 - level #Significance level

  x_pred <- x[,2] #Get the second column, the predictor
  s_x2 <- var(x_pred) #Variance of the input predictors
  x_mean <- mean(x_pred) #Mean of the input predictors
  t_value <- qt(1 - alpha/2, df) #T value for fitted value CI/PI

  if ( interval == "none"){
    intervals <- data.frame(
      Observed = y,
      Fitted = fitted.values
    )
  } else if (interval == "confidence"){

    #SE confidence of the fitted values: derived from slide 40 of lecture 8
    se_conf <- sigmahat * sqrt(1/n + (x_pred - x_mean)^2 / as.numeric(((n - 1) * s_x2)))

    t_value <- qt(1 - alpha/2, df) #For two tailed lb and ub
    lower_bound_CI <- fitted.values - t_value * se_conf
    upper_bound_CI <- fitted.values + t_value * se_conf

    #Update out intervals data frame with Confidence Intervals
    intervals <- data.frame(
      Observed = y,
      fit = fitted.values,
      CI_LB = lower_bound_CI,
      CI_UB = upper_bound_CI
    )
  } else if (interval == "prediction"){

    #SE prediction of the fitted values: derived from slide 43 of lecture 8
    se_pred <- sigmahat * sqrt(1 + 1/n + (x_pred - x_mean)^2 / as.numeric(((n - 1) * s_x2)))

    t_value <- qt(1 - alpha/2, df) #For two tailed lb and ub
    lower_bound_PI <- fitted.values - t_value * se_pred
    upper_bound_PI <- fitted.values + t_value * se_pred
```

```
    #Update our intervals data frame with Prediction Intervals
    intervals <- data.frame(
      Observed = y,
      fit = fitted.values,
      PI_LB = lower_bound_PI,
      PI_UB = upper_bound_PI
    )
  } else {
    se_conf <- sigmahat * sqrt(1/n + (x_pred - x_mean)^2 / as.numeric(((n - 1) * s_x2)))
    se_pred <- sigmahat * sqrt(1 + 1/n + (x_pred - x_mean)^2 / as.numeric(((n - 1) * s_x2)))

    t_value <- qt(1 - alpha/2, df) #For two tailed lb and ub
    lower_bound_PI <- fitted.values - t_value * se_pred
    upper_bound_PI <- fitted.values + t_value * se_pred
    lower_bound_CI <- fitted.values - t_value * se_conf
    upper_bound_CI <- fitted.values + t_value * se_conf

    #Combine Confidence and Prediction intervals into our dataframe
    intervals <- data.frame(
      Observed = y,
      fit = fitted.values,
      CI_LB = lower_bound_CI,
      CI_UB = upper_bound_CI,
      PI_LB = lower_bound_PI,
      PI_UB = upper_bound_PI
    )

  }

  # ~~~~~~~~~~~~~~~ Creating ANOVA table


  #Despite calculating these values above,
  #I computed it again with matrix algebra following the formula from slide 72

  SS_regr <- as.numeric(t(betahat) %*% t(x) %*% x %*% betahat - n*ybar^2)


  SS_residual <- as.numeric(t(y - x %*% betahat) %*% (y - x %*% betahat))
  SS_total <- as.numeric( t(y) %*% y - n*ybar^2)

  MSR <- SS_regr / p #Mean Squared Residuals
  MSE <- SS_residual / (n - p - 1) #Calculating Mean Squared Error

  F_stat <- MSR / MSE #Calculating F statistic

  #Calculating the P value
  p_value <-  pf(F_stat, p, n - p - 1, lower.tail = FALSE)

  #Putting all callculated values above together into a table:
  AOVtab <- data.frame(
  Source = c("Regression", "Residual", "Total"),
  df = c(p, n - p - 1, n - 1),
  SS = c(SS_regr, SS_residual, SS_total),
  MS = c(MSR, MSE, NA),
  F = c(F_stat, NA, NA),
  `P-value` = c(p_value, NA, NA)
)
  #Using the ANOVA table to calculate R squared
  Rsquared <- SS_regr/SS_total
```

```r
# ~~~~~~~~~~~~~~ Calculating CI's of coefficients

    t_value <- qt(1 - alpha/2, n-p-1) #T value for coefficient CI
    #LB and UB formulas from slide 70 lecture 8
    sigma_squared <- MSE
    beta_CI_LB <- betahat - t_value * sqrt(sigma_squared * diag(xtx_inv))
    beta_CI_UB <- betahat + t_value * sqrt(sigma_squared * diag(xtx_inv))

    estimateCIs <- data.frame(
      Label = paste0("beta_", 0:(length(betahat) - 1)),  # Add the Label first
      Coefficients = betahat,
      CI_LB = beta_CI_LB,
      CI_UB = beta_CI_UB
    )


  # ~~~~~~~~~~~~~~~~~~ Plotting the data

  # Plot the data and the regression line
fitted.values <- as.vector(fitted.values)  # Convert to a numeric vector
y <- as.vector(y)


plot_data <- data.frame(
  x = x[,2],
  y = y,
  fitted.values = fitted.values
              )



  # Add confidence or prediction bands depending on the selected interval type
if (interval == "confidence") {
 regressionplot <- ggplot(plot_data, aes(x = x)) +
    geom_point(aes(y = y)) +  # Plot the original data points
      # Plot the fitted line
    geom_line(aes(y = fitted.values, color = "Least Squares Regression Line"), show.legend = TRUE) +
    geom_ribbon(aes(ymin = intervals$CI_LB, ymax = intervals$CI_UB,
                    fill = "Confidence Interval"),
                alpha = 0.2) +
    labs(title = "Scatter Plot with Regression Line and Confidence Interval",
         x = "X", y = "Y") +
    scale_color_manual(values = c("Least Squares Regression Line" = "blue", "Data Points" = "black")) +
    scale_fill_manual(values = c("Confidence Interval" = "red")) +
    guides(color = guide_legend(title = NULL),
           fill = guide_legend(title = NULL,
                               override.aes = list(alpha = 0.2))) +
    theme(legend.position = "bottom")

} else if (interval == "prediction") {
   regressionplot <- ggplot(plot_data, aes(x = x)) +
    geom_point(aes(y = y)) +  # Plot the original data points
    geom_line(aes(y = fitted.values, color = "Least Squares Regression Line"), show.legend = TRUE) +  # Plot t
    geom_ribbon(aes(ymin = intervals$PI_LB, ymax = intervals$PI_UB,
                    fill = "Prediction Interval"),
                alpha = 0.2) +
    labs(title = "Scatter Plot with Regression Line and Prediction Interval",
         x = "X", y = "Y") +
```

```
    scale_color_manual(values = c("Least Squares Regression Line" = "blue", "Data Points" = "black")) +
    scale_fill_manual(values = c("Prediction Interval" = "green")) +
    guides(color = guide_legend(title = NULL),
           fill = guide_legend(title = NULL,
                               override.aes = list(alpha = 0.2))) +
    theme(legend.position = "bottom")

} else if( interval == "both"){


  regressionplot <- ggplot(plot_data, aes(x = x)) +
    geom_point(aes(y = y)) +  # Plot the original data points
    geom_line(aes(y = fitted.values, color = "Least Squares Regression Line"), show.legend = TRUE) +
    geom_ribbon(aes(ymin = intervals$CI_LB, ymax = intervals$CI_UB,
                    fill = "Confidence Interval"),
                alpha = 0.2) +
     geom_ribbon(aes(ymin = intervals$PI_LB, ymax = intervals$PI_UB,
                    fill = "Prediction Interval"),
                alpha = 0.2) +
    labs(title = "Scatter Plot with Regression Line and Confidence Interval",
         x = "X", y = "Y") +
    scale_color_manual(values = c("Least Squares Regression Line" = "blue", "Data Points" = "black")) +
    scale_fill_manual(values = c("Confidence Interval" = "red", "Prediction Interval" = "green")) +
    guides(color = guide_legend(title = NULL),
           fill = guide_legend(title = NULL,
                               override.aes = list(alpha = 0.2))) +
    theme(legend.position = "bottom")


}

  # The returned values are:
  return(list("coefficients" = betahat, # The regression coefficients
              "Q" = Qmat, # The Q matrix only if method = "qr"
              "R" = Rmat, # The R matrix only if method = "qr"
              "sigmahat" = sigmahat, # The estimate of sigma
              "fitted.values" = fitted.values, # The fitted values
              "residuals" = residuals, # The residuals
              "Rsquared" = Rsquared, # The value of R-squared
              "EstimateTab" = EstimateTab, # The estimate table (searchable)
              "AOVtab" = AOVtab,  # The ANOVA Table (searchable)
              "estimateCIs" = estimateCIs, # A table of the CIs for beta0 and beta1
              "intervals" = intervals, # yhat with the lower & upper bounds for the CIs and/or PIs
              "regressionplot" = regressionplot # plot of regression fit and CI/PI  bands, if requested
              ))

}
```

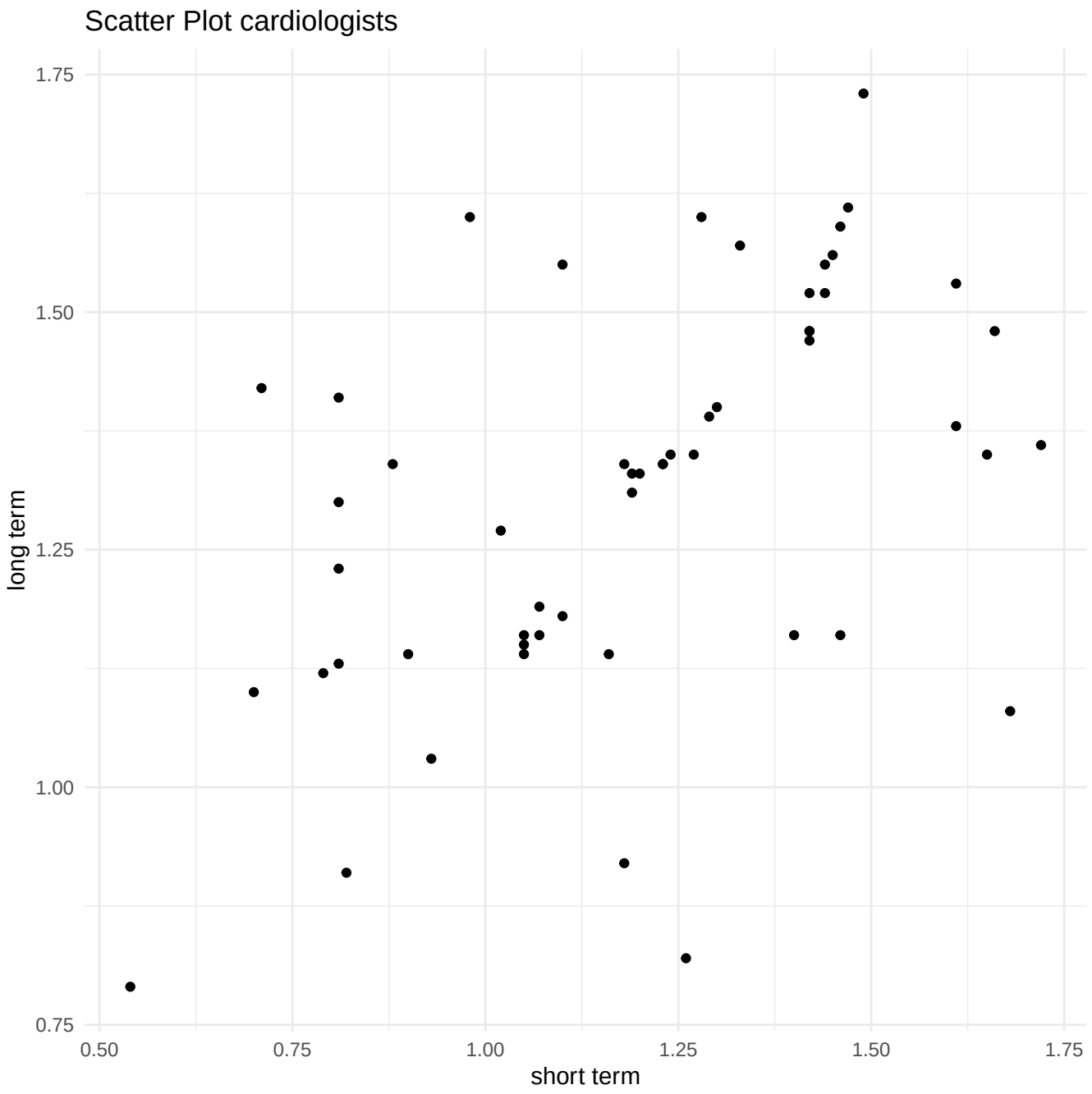
Problem 1.)

```
subdirectory <- "data_storage"

file_path <- list.files(path = subdirectory, pattern = "cardiologists.csv", full.names = TRUE)
file.exists(file_path)
```

```
[1] TRUE
```

```
data <- read.csv(file_path)
```

```
ggplot(data, aes(x = Short, y = Long)) +
  geom_point() +  # Plot the original data points
  labs(title = "Scatter Plot cardiologists", x = "short term", y = "long term") +
  theme_minimal()
```



Yes, there appears to be a linear trend and thus a linear relationship is possible although not a overwhelmingly strong relationship.

```
x = as.matrix(data$Short)
results <- mylm(data$Long, x, method = "inverse", interval = "both")
Qmat <- results$Qmat
Rmat <- results$Rmat
sigmahat <- results$sigmahat
fitted.values <- results$fitted.values
residuals <- results$residuals
Rsquared <- results$Rsquared
```

```
EstimateTab <- results$EstimateTab
AOVtab <- results$AOVtab
estimateCIs <- results$estimateCIs
intervals <- results$intervals
regressionplot <- results$regressionplot
```

a.)

Output of the betahat vector

```
print(EstimateTab$Coefficient)
```

```
[1] 0.8755347 0.3641873
```

Table for 95% confidence intervals for beta vector

```
print(estimateCIs)
```

```
   Label Coefficients     CI_LB     CI_UB
1 beta_0    0.8755347 0.6524121 1.0986572
2 beta_1    0.3641873 0.1830835 0.5452911
```

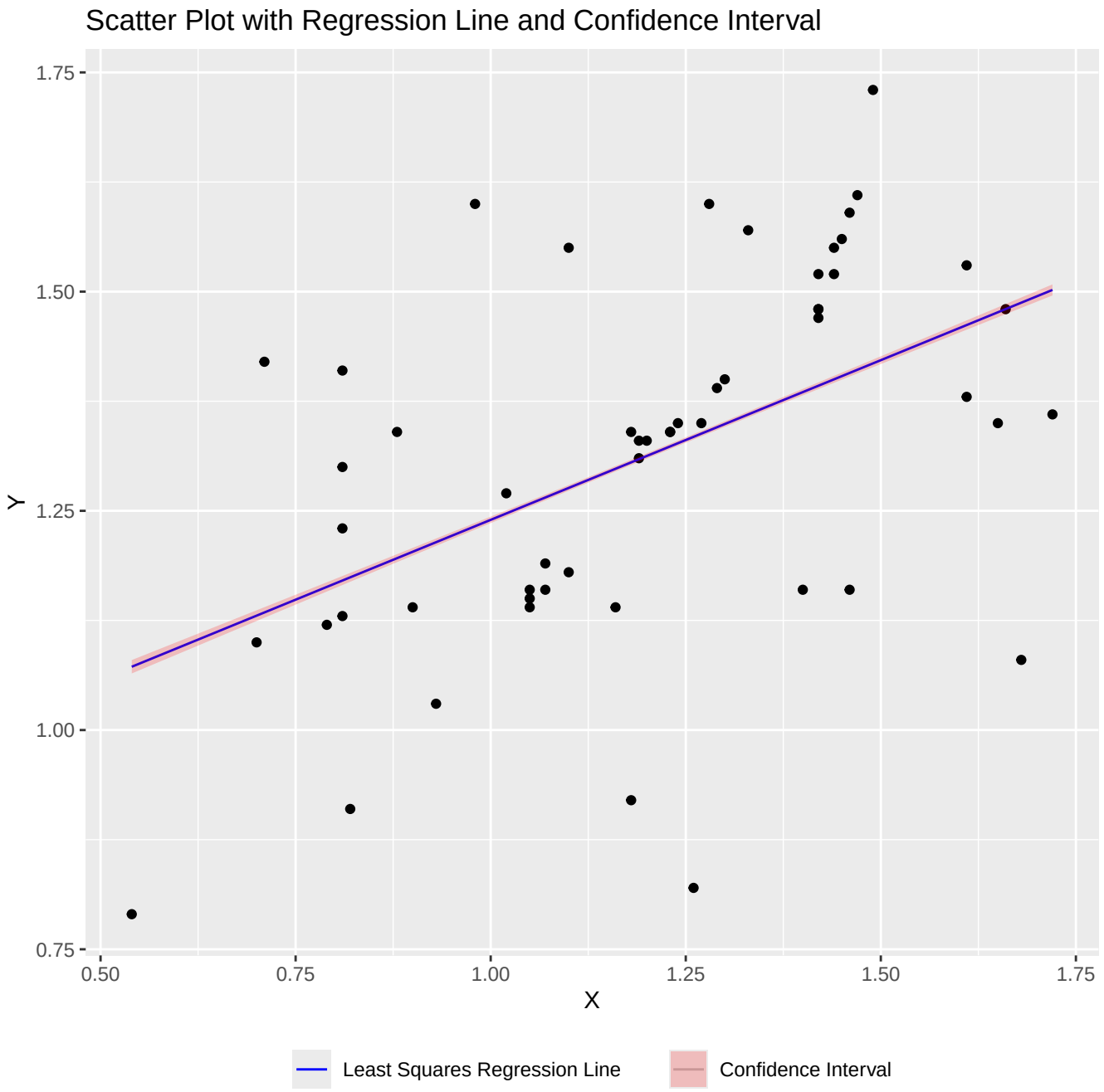
Here are all outputs of the function with QR decomposition. As you can see it matches the exact values using the inverse matrix method.

Inverse Matrix Method Outputs:

```
#Fitted with QR decomposition
qr_output = mylm(data$Long, x, method = "qr")

#Display the statistics of the coefficients and a scatterplot with least squares regression line
print(qr_output$estimateCIs)
```

```
   Label Coefficients     CI_LB     CI_UB
1 beta_0    0.8755347 0.6524121 1.0986572
2 beta_1    0.3641873 0.1830835 0.5452911
```

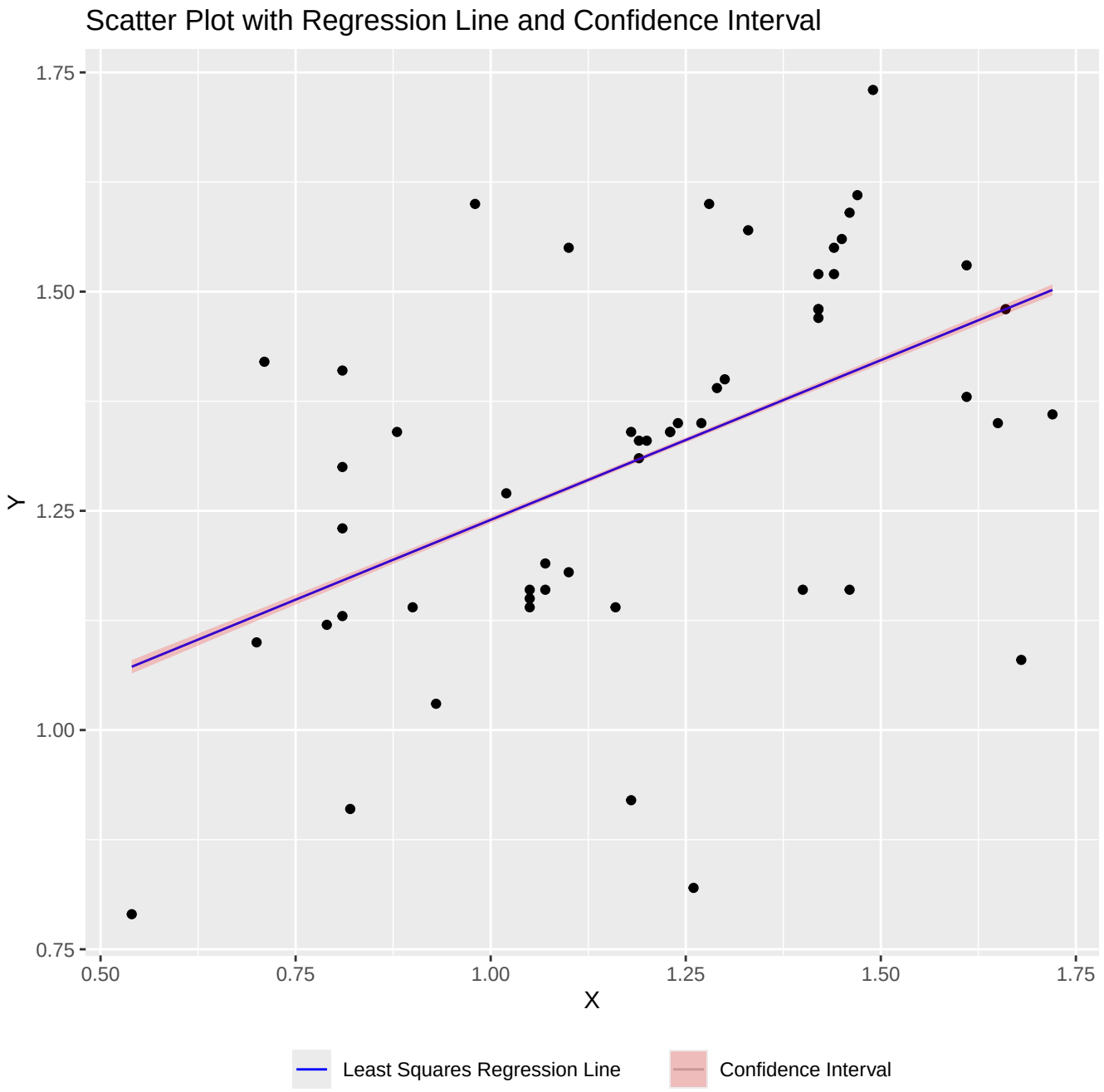
```
print(qr_output$regressionplot)
```




```
#Fitted with inverse matrix method
inverse_output = mylm(data$Long, x, method = "inverse")
print(inverse_output$estimateCIs)
```

```
   Label Coefficients     CI_LB     CI_UB
1 beta_0    0.8755347 0.6524121 1.0986572
2 beta_1    0.3641873 0.1830835 0.5452911
```

```
print(inverse_output$regressionplot)
```



Judging from the coefficients and confidence intervals, both methods of using QR decomposition and inverse matrix reaches the same result.

Equation for the fitted regression line: $Y = 0.8755347 + 0.3641873(X)$

b.)

```
print(EstimateTab)
```

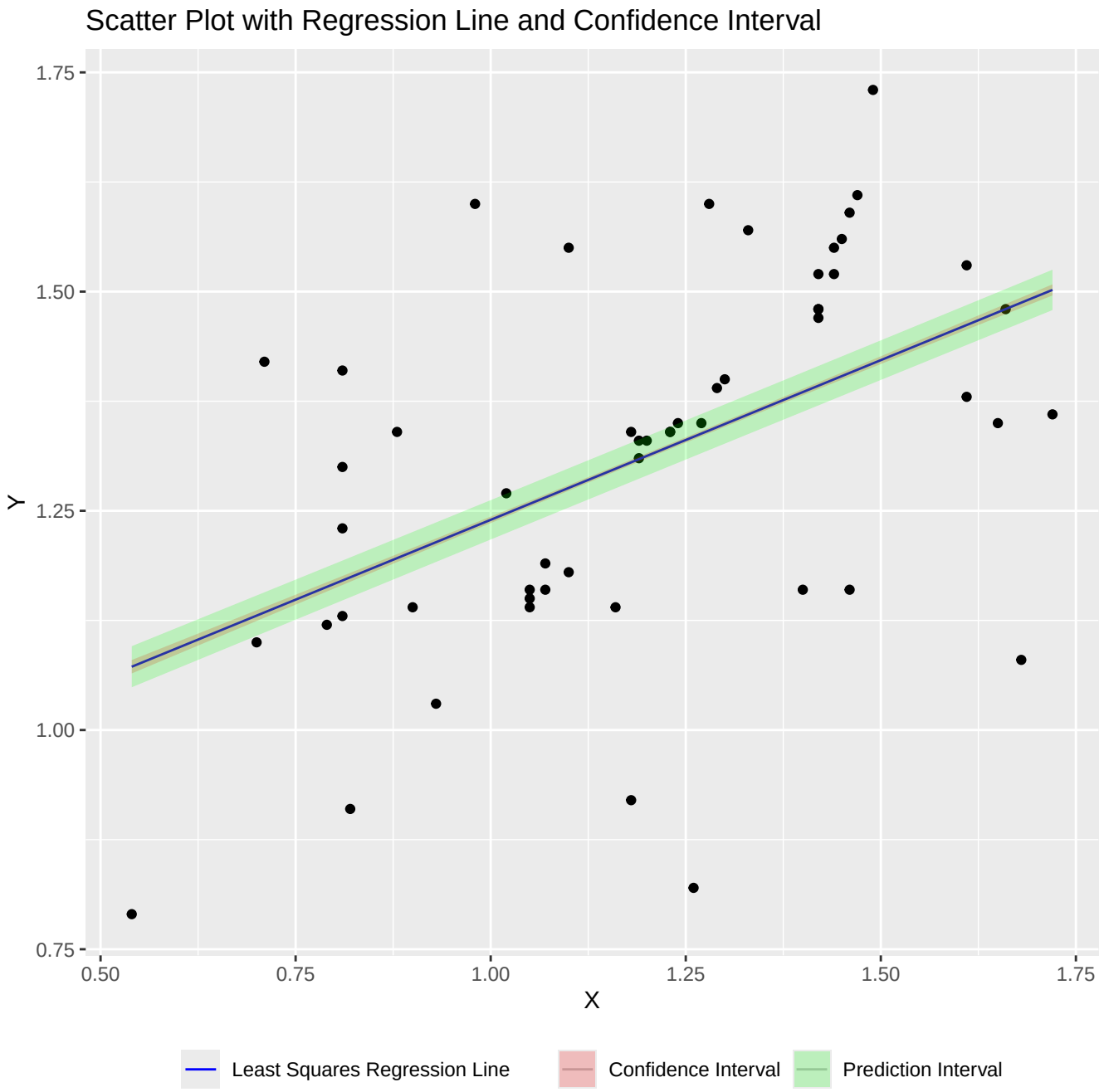
```
   Label Coefficient   StdError tStatistic       pValue
1 beta_0   0.8755347 0.06222131  14.071300 2.290761e-19
2 beta_1   0.3641873 0.05050369   7.211103 2.279673e-09
```

Looking at our Estimate Table we can see the coefficient for betahat_1 is positive therefore the relationship between our short term measurements and long term measurements is positive, meaning as our x values increase so does the y values.

Additionally, when the short term measurement is 0, the long term measurement is 0.8755347 given by the value of beta_0

c.) Here is a scatterplot overlayed with a least squares regression line and both confidence and prediction intervals.

```
print(regressionplot)
```



d.)

```
print(AOVtab)
```

```
      Source df        SS         MS        F      P.value
1 Regression  2 0.5735626 0.28678131 8.149145 0.0008490568
2   Residual 51 1.7947707 0.03519158       NA           NA
3      Total 53 2.3683333         NA       NA           NA
```

```
print(Rsquared)
```

```
[1] 0.2421799
```

The Rsquared value is: 0.2421799 which is the ratio of SSR/SST: variation in Y explained by our regression Line / total variation in Y Thus 24.21799% of variation in Y is explained in our linear regression model.

Problem 2

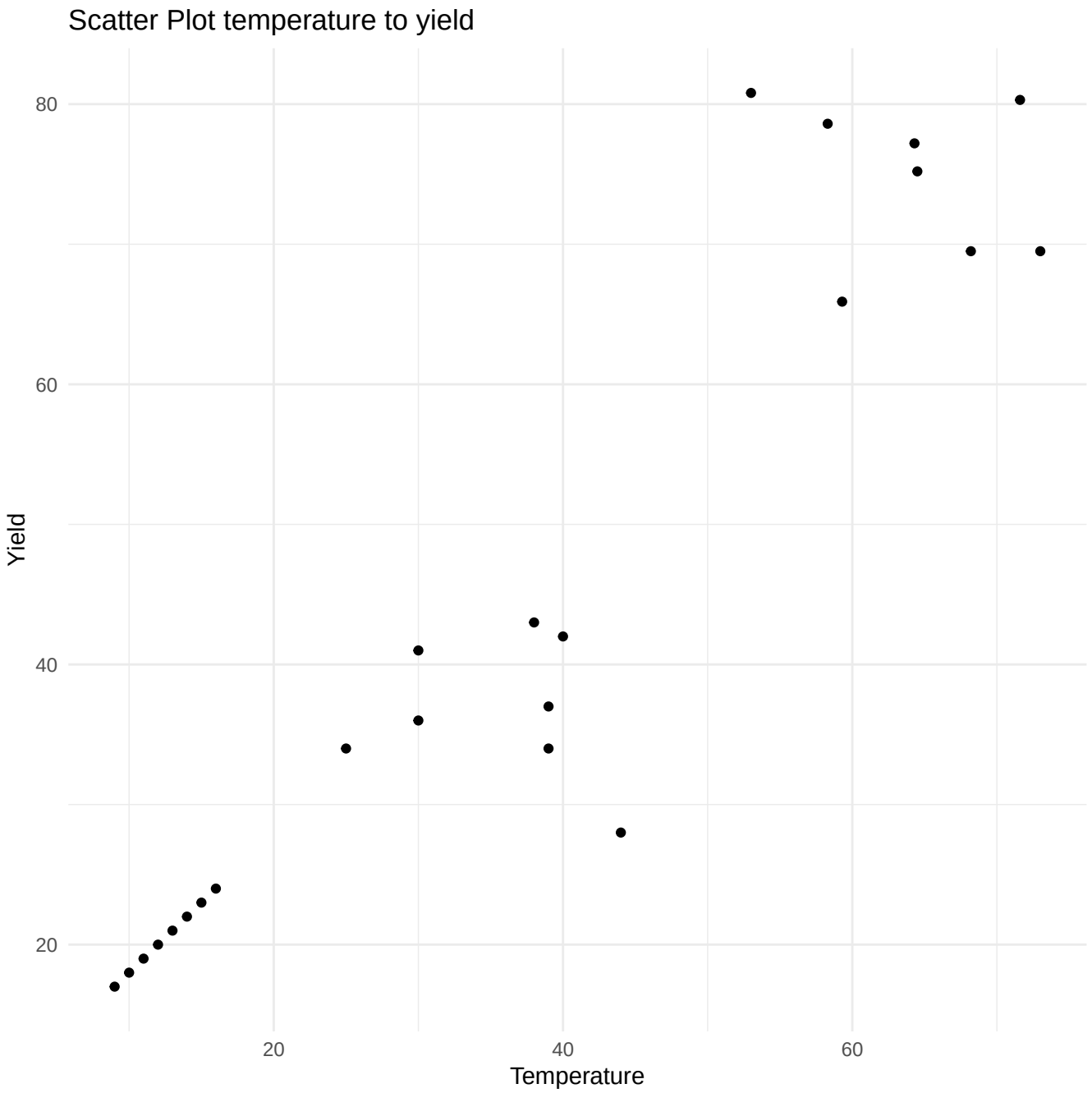
a.) First, importing the dataset

```
file_path <- list.files(path = subdirectory, pattern = "chemical.csv", full.names = TRUE)

chemical_data <- read.csv(file_path)

head(chemical_data)
```

```
  Order Temp Yield
1   1.0  9.0  17.0
2  30.0 25.0  34.0
3  49.2 59.3  65.9
4   2.0 10.0  18.0
5  32.0 38.0  43.0
6  55.3 64.5  75.2
```

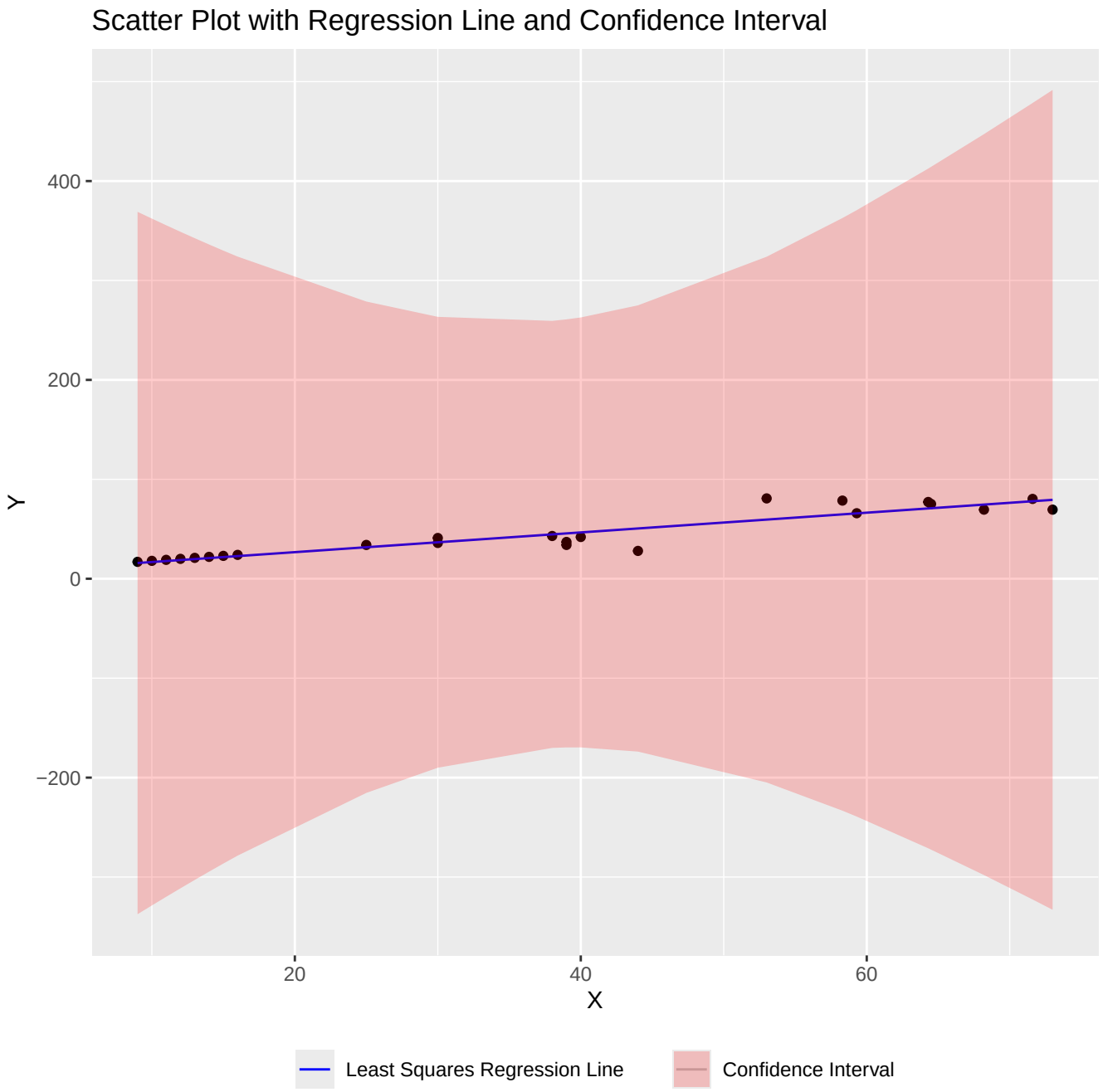
Getting a visual scatterplot of the chemical dataset

```
ggplot(chemical_data, aes(x = Temp, y = Yield)) +
  geom_point() +  # Plot the original data points
  labs(title = "Scatter Plot temperature to yield", x = "Temperature", y = "Yield") +
  theme_minimal()
```



2a.) Fitting the chemical data to mylm function

```
x = as.matrix(chemical_data$Temp)
results <- mylm(chemical_data$Yield, x, method = "inverse")
Qmat <- results$Qmat
Rmat <- results$Rmat
sigmahat <- results$sigmahat
fitted.values <- results$fitted.values
residuals <- results$residuals
Rsquared <- results$Rsquared
EstimateTab <- results$EstimateTab
AOVtab <- results$AOVtab
estimateCIs <- results$estimateCIs
intervals <- results$intervals
regressionplot <- results$regressionplot

print(regressionplot)
```




```
print(Rsquared)
```

```
[1] 0.8737721
```

```
print(mean(residuals))
```

```
[1] 5.995204e-15
```

Looking at the graph we can see a very close match between the least regression line and the points.

The R-squared value is 0.8737721 meaning our model explains 87.37721% of the total variance in the yield.

Additionally the mean of the residuals is 5.995204e-15 which is incredibly low meaning the real data points deviates minimally from our least squares regression line.

Therefore, looking at this plot and its test statistics, it seems to follow a linear regression model pretty well thus a linear model is appropriate for this data.

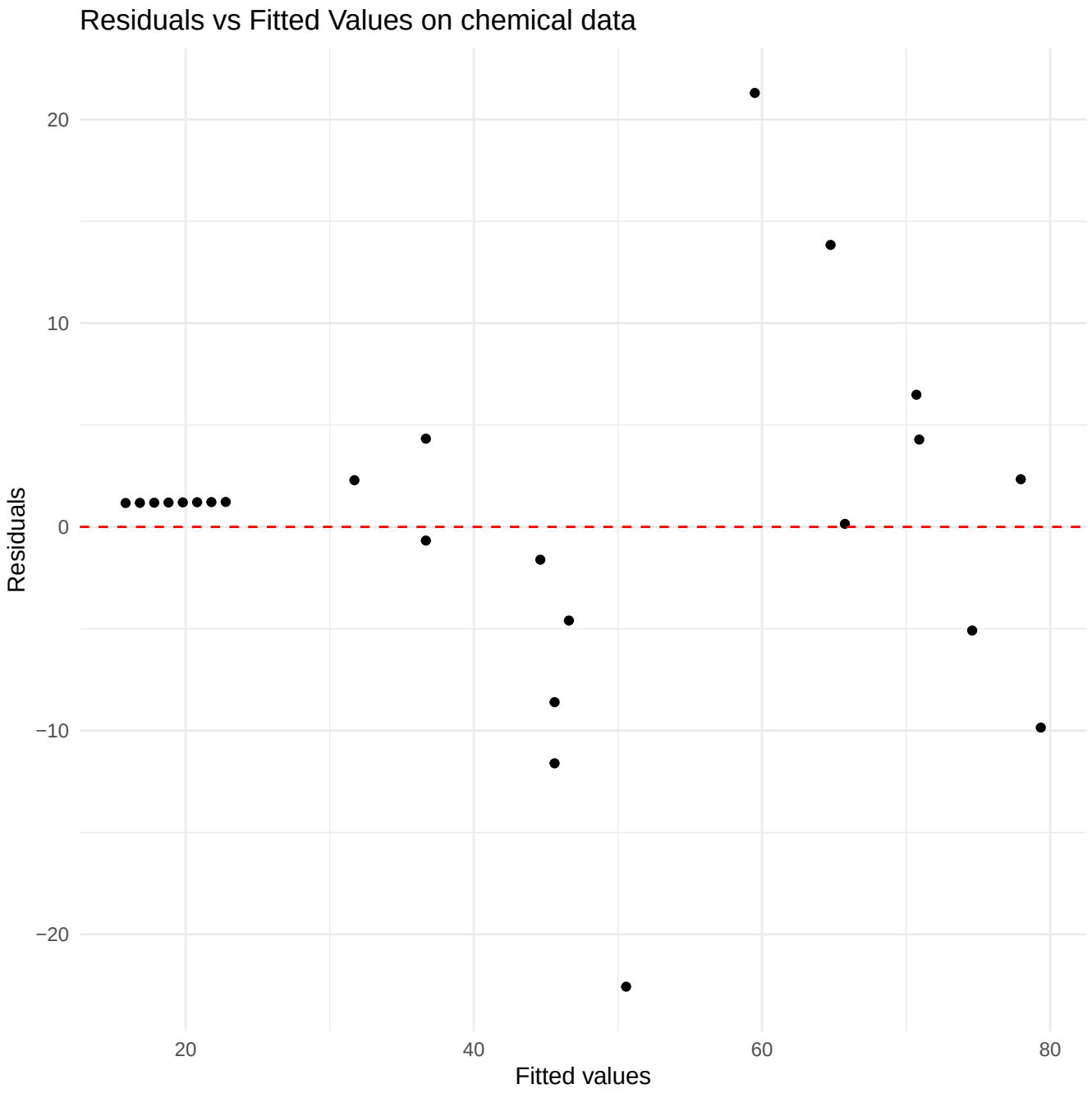
2b.)

A violation of the SLR (Simple Linear Regression) is the observations not being independent. The observations are done in order meaning there is a time dependency.

A proposition would be to have the order conducted as another predictor variable. Therefore the model takes into account when the data point was conducted and can make adjustments accordingly.

Let us get a residual plot to find any potential problems:

```
#Organize the fitted.values and residuals into a dataframe
residual_data <- data.frame(
  fitted = fitted.values,
  residuals = residuals
)

# Create the residual plot
residual_plot <- ggplot(residual_data, aes(x = fitted, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(x = "Fitted values",
       y = "Residuals",
       title = "Residuals vs Fitted Values on chemical data") +
  theme_minimal()
print(residual_plot)
```



We see a apparent trend that the smaller fitted values have really small residuals and as the fitted values grow, the residual values grow dramatically. Therefore we can confirm that the assumption of the linear model does not hold.

Problem 3.)

a.)

First import the data:

```
file_path <- list.files(path = subdirectory, pattern = "myanimals.csv", full.names = TRUE)

animal_data <- read.csv(file_path)

head(animal_data)
```

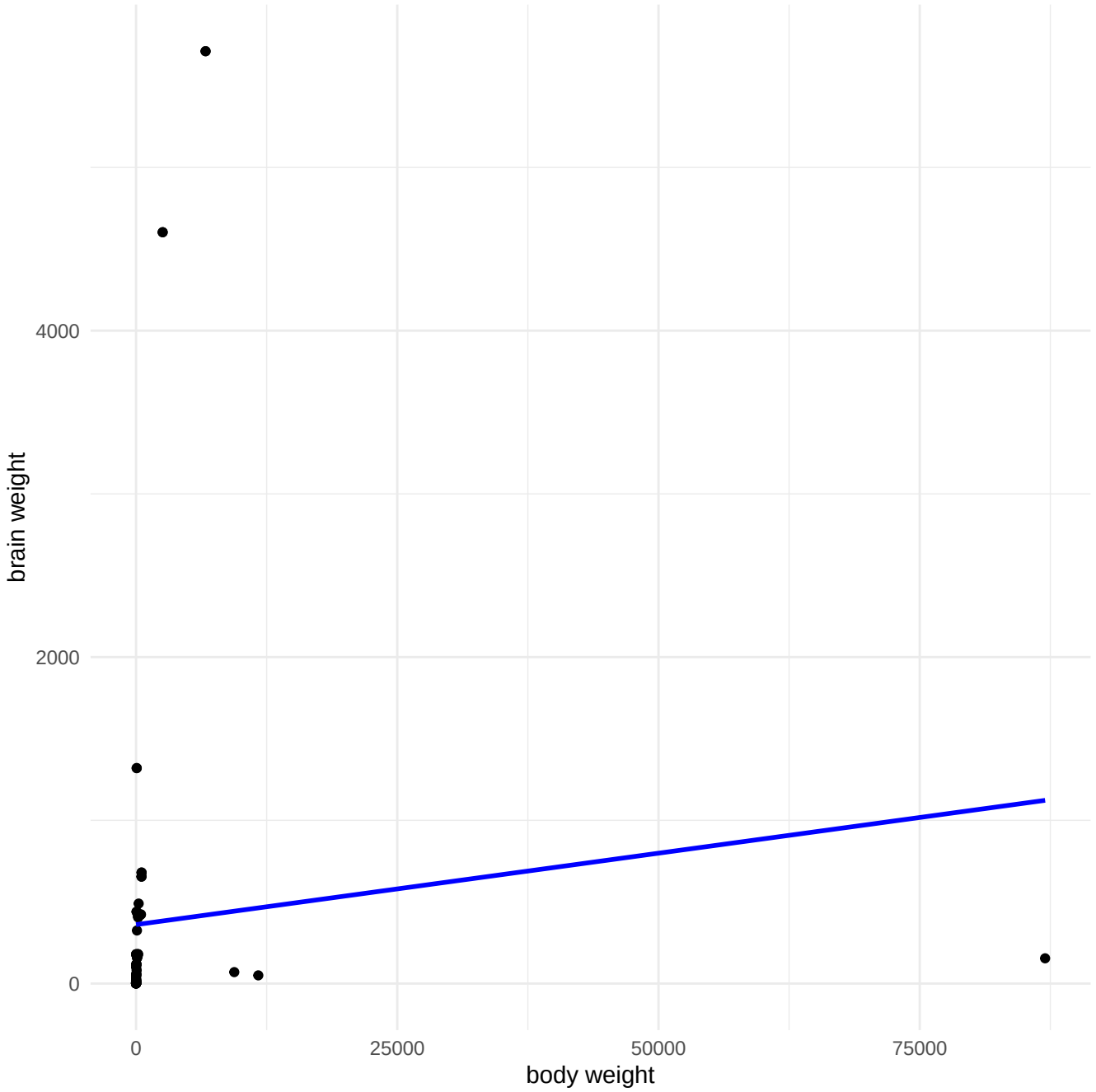
```
           animal     body brain
1 Mountain beaver     1.35   8.1
2             Cow   465.00 423.0
3       Grey wolf    36.33 119.5
4            Goat    27.66 115.0
5      Guinea pig     1.04   5.5
6     Dipliodocus 11700.00  50.0
```

Fit the model with body weight as predictor and brain weight as prediction.

```
animal_model <- lm(brain ~ body, data=animal_data)
```

Make a scatterplot to visualize the data

```
ggplot(animal_data, aes(x = body, y = brain)) +
  geom_point() +  # Add scatter points
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  # Add regression line with confidence interval
  labs(
       x = "body weight",
       y = "brain weight") +
  theme_minimal()
```



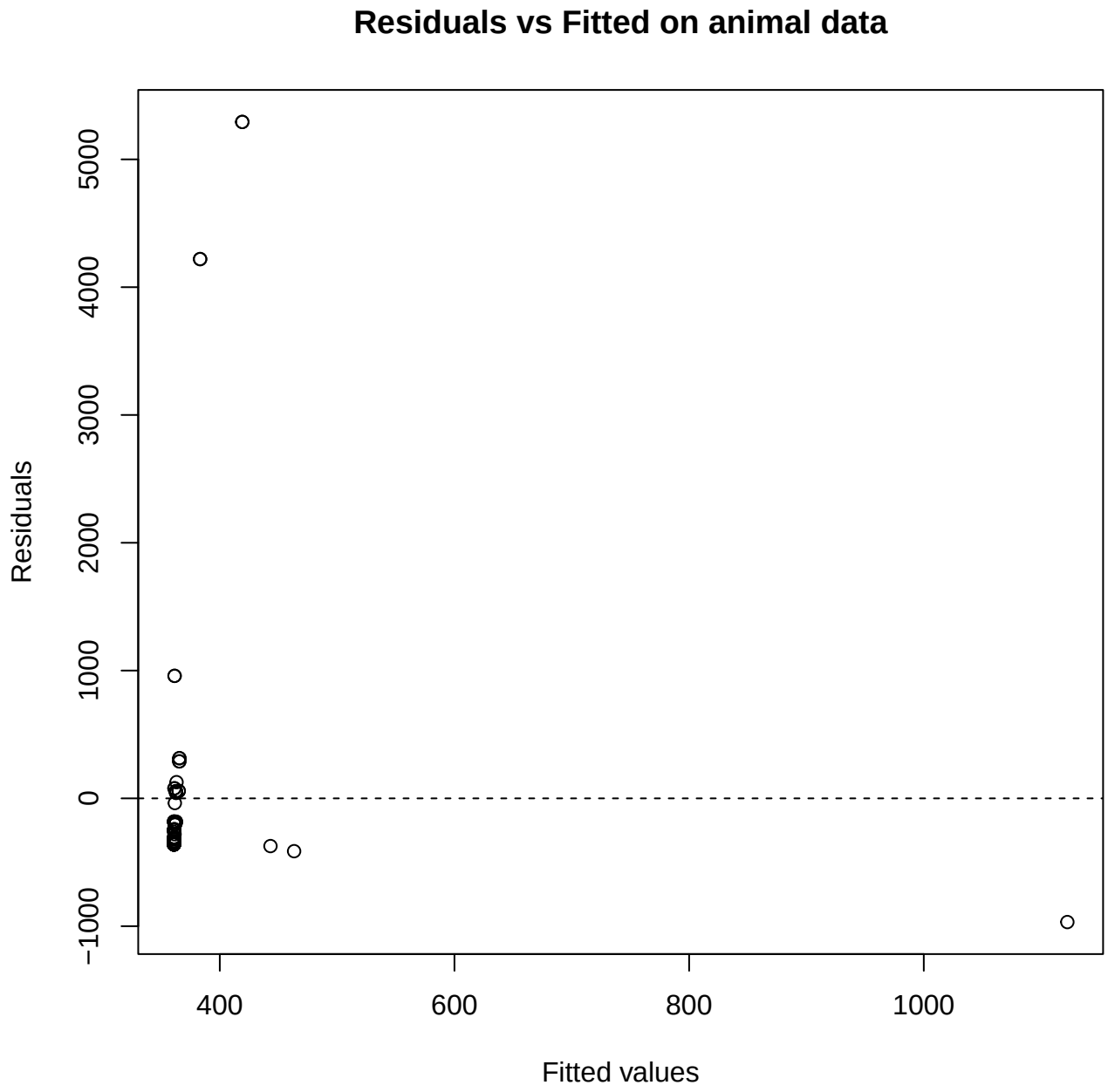

```
animal_model_stats <- summary(animal_model)
print(animal_model_stats)
```

```

Call:
lm(formula = brain ~ body, data = animal_data)

Residuals:
   Min     1Q Median     3Q    Max
-967.9 -356.7 -335.4 -182.2 5292.8

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.609e+02  1.149e+02   3.142  0.00229 **
body        8.752e-03  1.226e-02   0.714  0.47732
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1076 on 88 degrees of freedom
Multiple R-squared:  0.005754,  Adjusted R-squared:  -0.005544
F-statistic: 0.5093 on 1 and 88 DF,  p-value: 0.4773
```

Print out a residual plot to see any trends.

```
plot(fitted(animal_model), residuals(animal_model),
     xlab = "Fitted values",
     ylab = "Residuals",
     main = "Residuals vs Fitted on animal data")
abline(h = 0, lty = 2)  # Add a horizontal line at y = 0
```



There doesn't appear to be any curves thus, suggesting that the assumptions of the linear model holds. So while no assumption of the linear model is broken, the model does not fit to the data well looking at the summary statistics.

With .5% of variance in the predictions being explained in our model.

A P value of .47732 strongly implies that we cannot reject the null hypothesis of the coefficient beta1 being 0.

We must transform this data to fit a linear regression model better.

b.)

I noticed a lot of the data points are clustered near the center (0, 0). Thus a potential transformation would be to use Log(brain weight) against the Log(body weight).

The reason I would like to use the Log transformation is that I see a lot of outliers far into the y and x axis. Therefore using this transformation I am able to get a better spread of data throughout both x and y axis.

c.)

Fit the model to the log of the prediction values, the brain weight

```
Log_animal_model <- lm(log(brain) ~ log(body), data=animal_data)
```

Make a new prediction on the body of a animal being 4kg

```
new_data <- data.frame(body = 4)

log_predictions <- predict(Log_animal_model, newdata = new_data, interval = "prediction", level = 0.95)
predictions <- exp(log_predictions) #Since the outputted predictions are in log form we will need to convert i
print(predictions)
```

```
       fit     lwr      upr
1 21.92687 2.42684 198.1126
```

The predicted outputted for brain being 4kg is 21.92687. The lower bound of the 95% prediction interval is 2.42684 kg, and the upper bound of the 95% prediction interval is 198.1126 kg.

d.)

Now to see the statistics of whether or not our transformation worked.

```
log_model_stats <- summary(Log_animal_model)
print(log_model_stats)
```

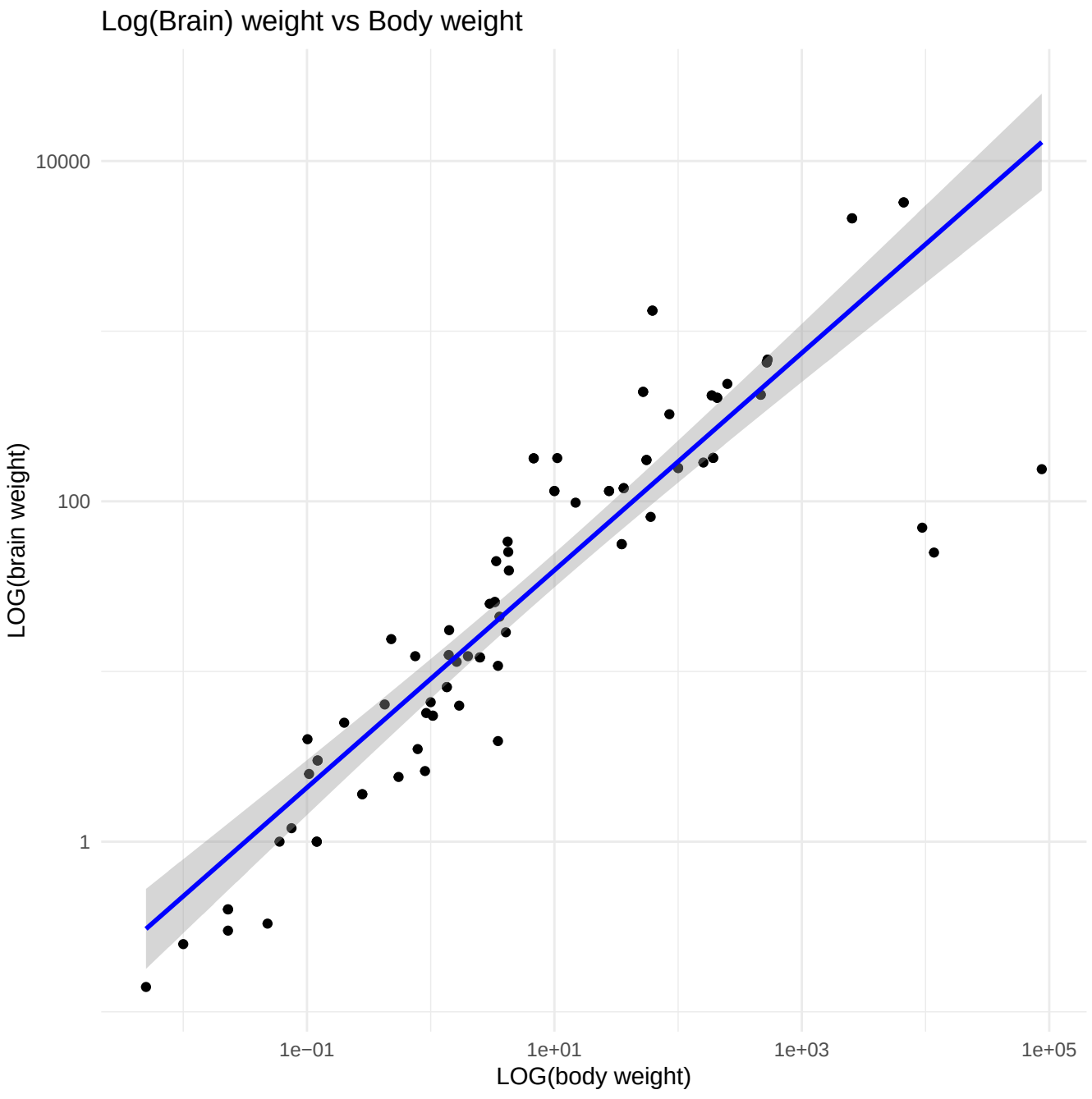
```

Call:
lm(formula = log(brain) ~ log(body), data = animal_data)

Residuals:
    Min      1Q  Median      3Q     Max
-4.4247 -0.4411  0.1795  0.4938  2.3476

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.20253    0.13544   16.26   <2e-16 ***
log(body)    0.63852    0.03331   19.17   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.101 on 88 degrees of freedom
Multiple R-squared:  0.8068,    Adjusted R-squared:  0.8046
F-statistic: 367.5 on 1 and 88 DF,  p-value: < 2.2e-16
```

P value is well below $< .05$, our significance level, therefore we can reject the null hypothesis of the coefficients being 0, meaning there is a linear relation.

R-squared is .8 meaning 80% of the variance in the predictions can be explained in our model. Significantly improving upon the original statistics.

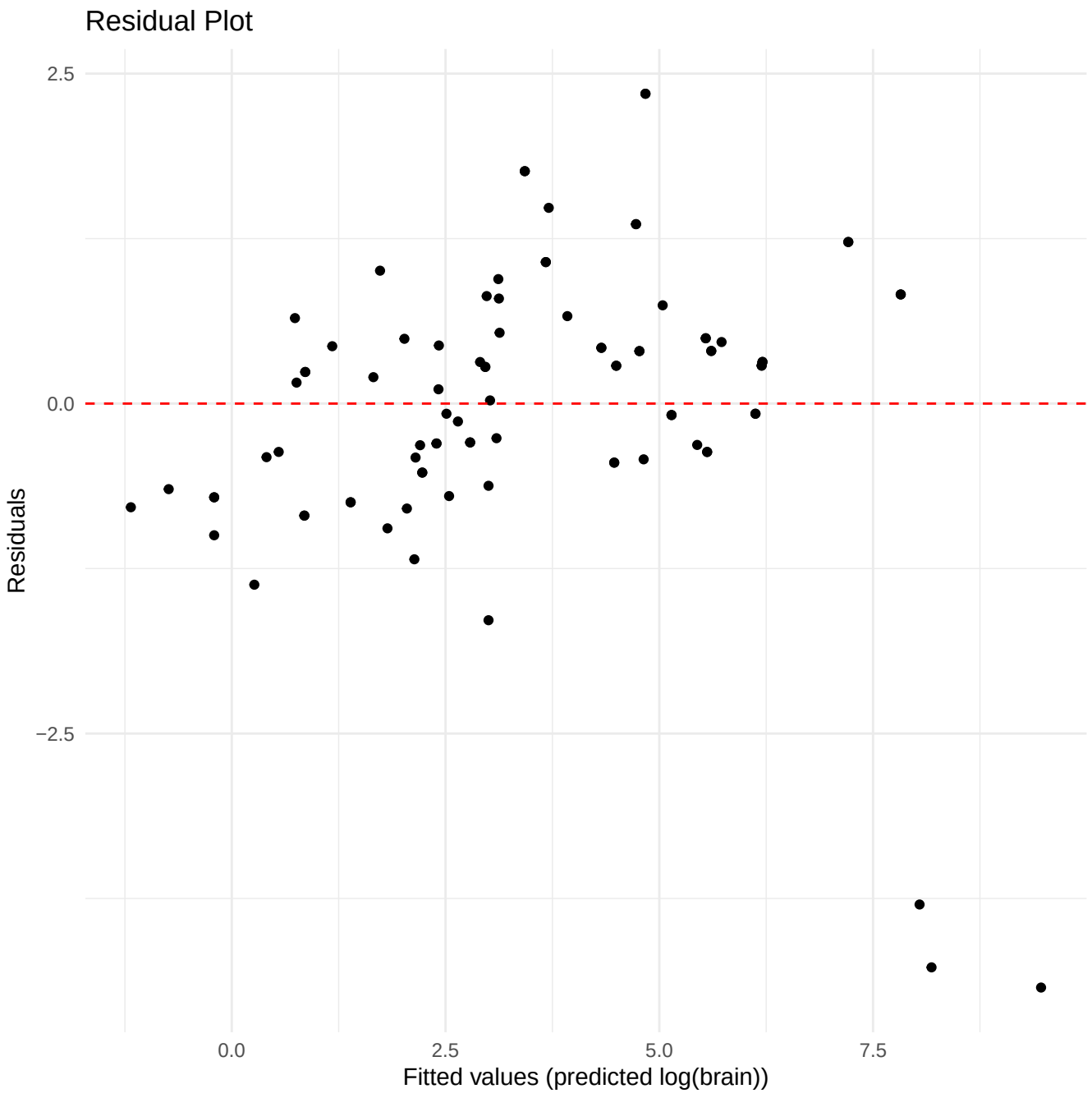
Now to visualize these improvements.

```
ggplot(animal_data, aes(x = body, y = brain)) +
  geom_point() +  # Add scatter points
  geom_smooth(method = "lm", se = TRUE, color = "blue") +  # Add regression line with confidence interval
  scale_y_log10() +  # This transforms the y-axis to log scale
  scale_x_log10()+
  labs(title = "Log(Brain) weight vs Body weight",
       x = "LOG(body weight)",
       y = "LOG(brain weight)") +
  theme_minimal()
```



Clearly, there is a much stronger linear relationship than the figure before this. Now let's plot the residuals to see if there are any trends hinting against a linear relationship still.

```
plot_data <- data.frame(
  fitted = fitted(Log_animal_model),
  residuals = residuals(Log_animal_model)
)

# Create the correct residual plot
ggplot(plot_data, aes(x = fitted, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(x = "Fitted values (predicted log(brain))",
       y = "Residuals",
       title = "Residual Plot") +
  theme_minimal()
```



Unfortunately there still appears to be a trend, thus still not a entirely linear relationship.

Specifically, the larger the fitted value the more variance there is in its residuals. Furthermore, once we reach past 7.5 (log scale) of the fitted values, the residuals dip heavily in the negative direction.

Looking at the scatterplot of Log(Brain weight) vs Log(body weight), we see the potential problems in its outliers. The outliers shouldn't be deleted as their existence can't be ignored. Thus I believe that my next step would be to fit a polynomial regression rather than a linear regression to the model which potentially could capture a nonlinear relationship in the data. Specifically I would try adding the quadratic term (log(body weight)^2) to our model and see how that fits.

Problem 4

Import the data

```
file_path <- list.files(path = subdirectory, pattern = "nonlinear_data.csv", full.names = TRUE)
nonlinear_data <- read.csv(file_path)

head(nonlinear_data)
```

```
         X1        Y1          X2        Y2
1 0.9173578 1.0332159 -2.69725933 -9.004791
2 0.4124069 0.3989528 -1.54282303 -1.568273
3 0.4976991 0.5638831 -1.91720835 -3.850695
4 0.6582827 0.7420445 -0.08528522  3.806207
5 0.9265662 0.9649855 -2.57551112 -7.781956
6 0.3613455 0.3057001 -2.65955930 -6.211738
```

a.)

Fitting a simple linear regression fit, one for Y1 ~ X1 another for Y2 ~ X2

```
lr_model_1 <- lm(Y1 ~ X1, data = nonlinear_data)
lr_model_2 <- lm(Y2 ~ X2, data = nonlinear_data)
```

View the test statistics of both models

```
model_1_stats <- summary(lr_model_1)
model_2_stats <- summary(lr_model_2)
print(model_1_stats)
```

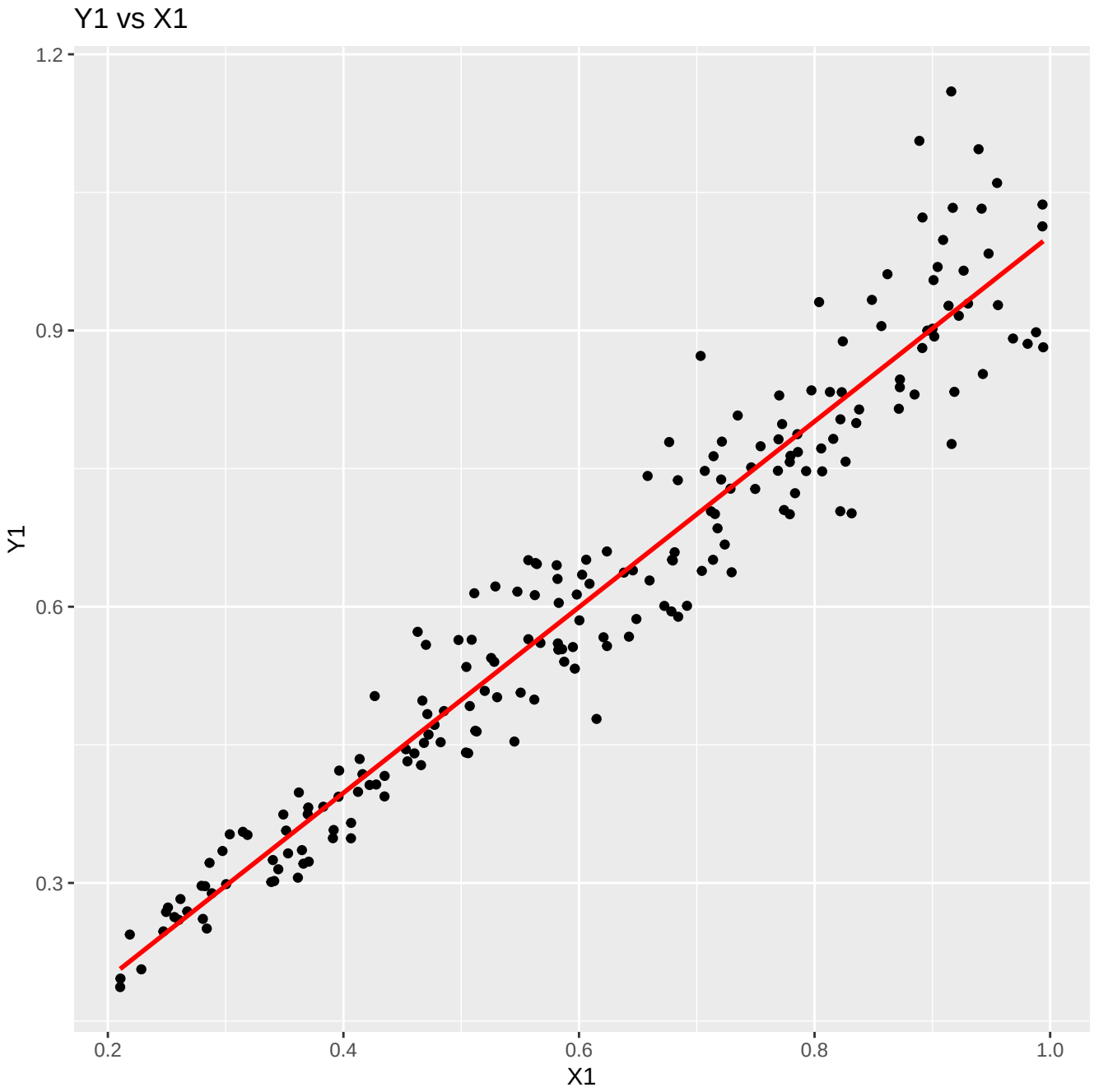
```

Call:
lm(formula = Y1 ~ X1, data = nonlinear_data)

Residuals:
      Min        1Q    Median        3Q       Max
-0.141860 -0.035989 -0.006098  0.032806  0.241313

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.005676   0.012892   -0.44     0.66
X1           1.008642   0.019798   50.95   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06026 on 198 degrees of freedom
Multiple R-squared:  0.9291,    Adjusted R-squared:  0.9288
F-statistic:  2596 on 1 and 198 DF,  p-value: < 2.2e-16
```

```
print(model_2_stats)
```

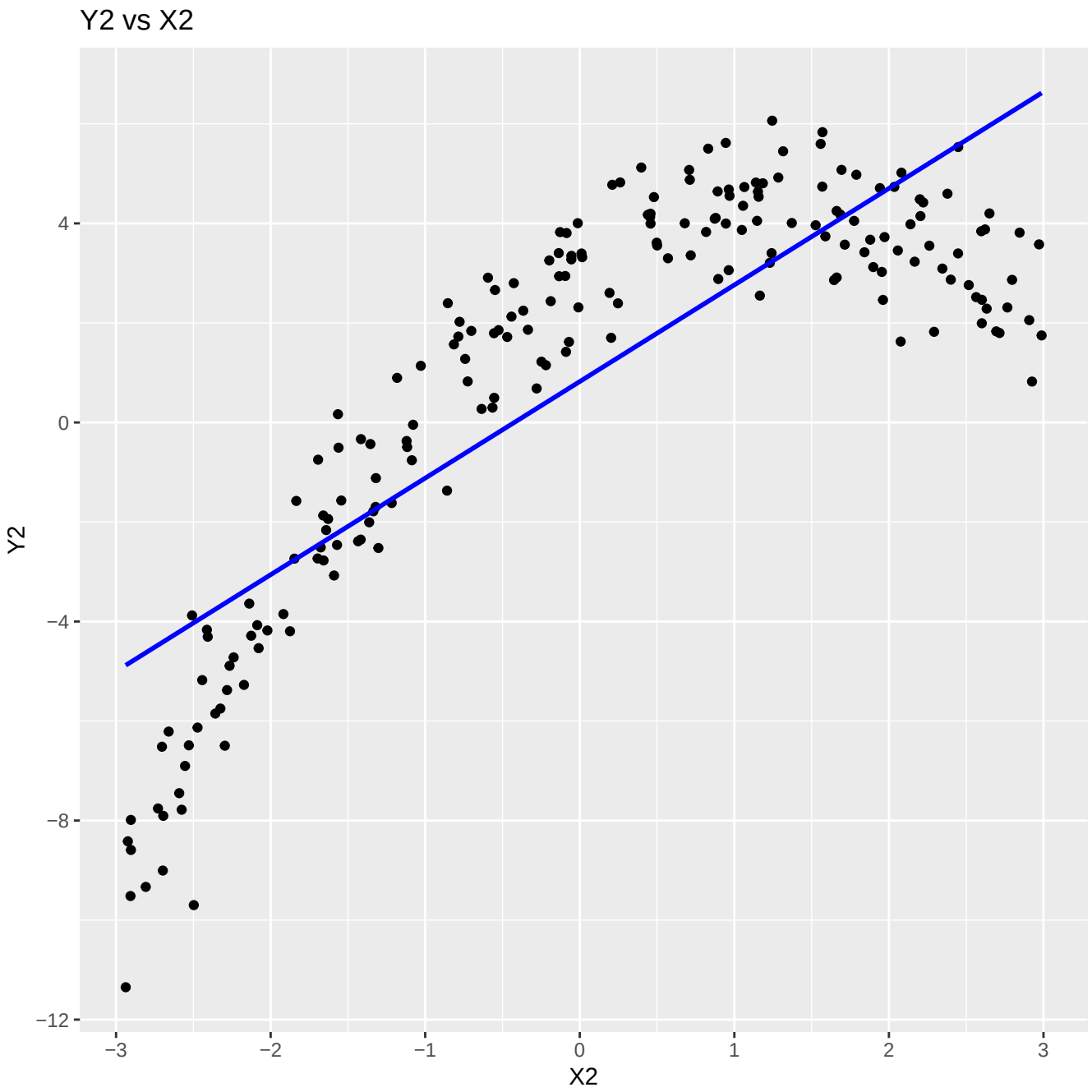
```
Call:
lm(formula = Y2 ~ X2, data = nonlinear_data)

Residuals:
    Min      1Q  Median      3Q     Max
-6.4699 -1.3650  0.2011  1.7931  3.5460

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.82193    0.15713   5.231 4.28e-07 ***
X2           1.94091    0.09149  21.215  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.221 on 198 degrees of freedom
Multiple R-squared:  0.6945,    Adjusted R-squared:  0.6929
F-statistic: 450.1 on 1 and 198 DF,  p-value: < 2.2e-16
```

Now to plot the data:

```
ggplot(nonlinear_data, aes(x = X1, y = Y1)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Y1 vs X1", x = "X1", y = "Y1")
```




```
ggplot(nonlinear_data, aes(x = X2, y = Y2)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(title = "Y2 vs X2", x = "X2", y = "Y2")
```



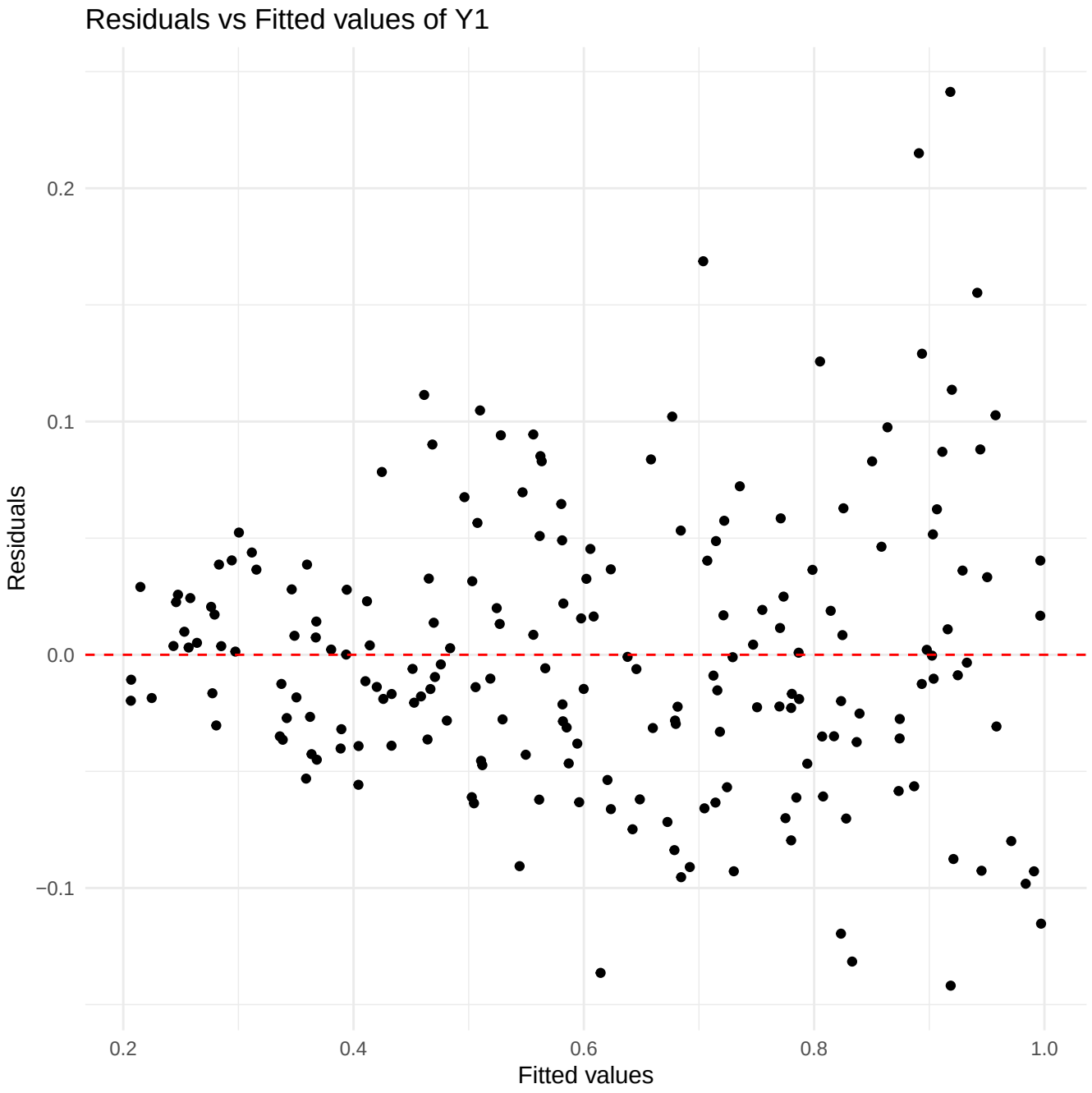
We could tell from the summary statistics tables that the model for Y2 ~ X2 doesn't perform as well as Y1 ~ X1 but looking at the scatterplots we can see there is a strong distinction in that Y2 ~ X2 not having a linear relationship.

b.)

Scatterplot for residuals vs fitted Y1 values

```
plot_data <- data.frame(
  fitted = fitted(lr_model_1),
  residuals = residuals(lr_model_1)
)

# Create the correct residual plot
ggplot(plot_data, aes(x = fitted, y = residuals)) +
  geom_point() +
```

```
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(x = "Fitted values",
       y = "Residuals",
       title = "Residuals vs Fitted values of Y1") +
  theme_minimal()
```

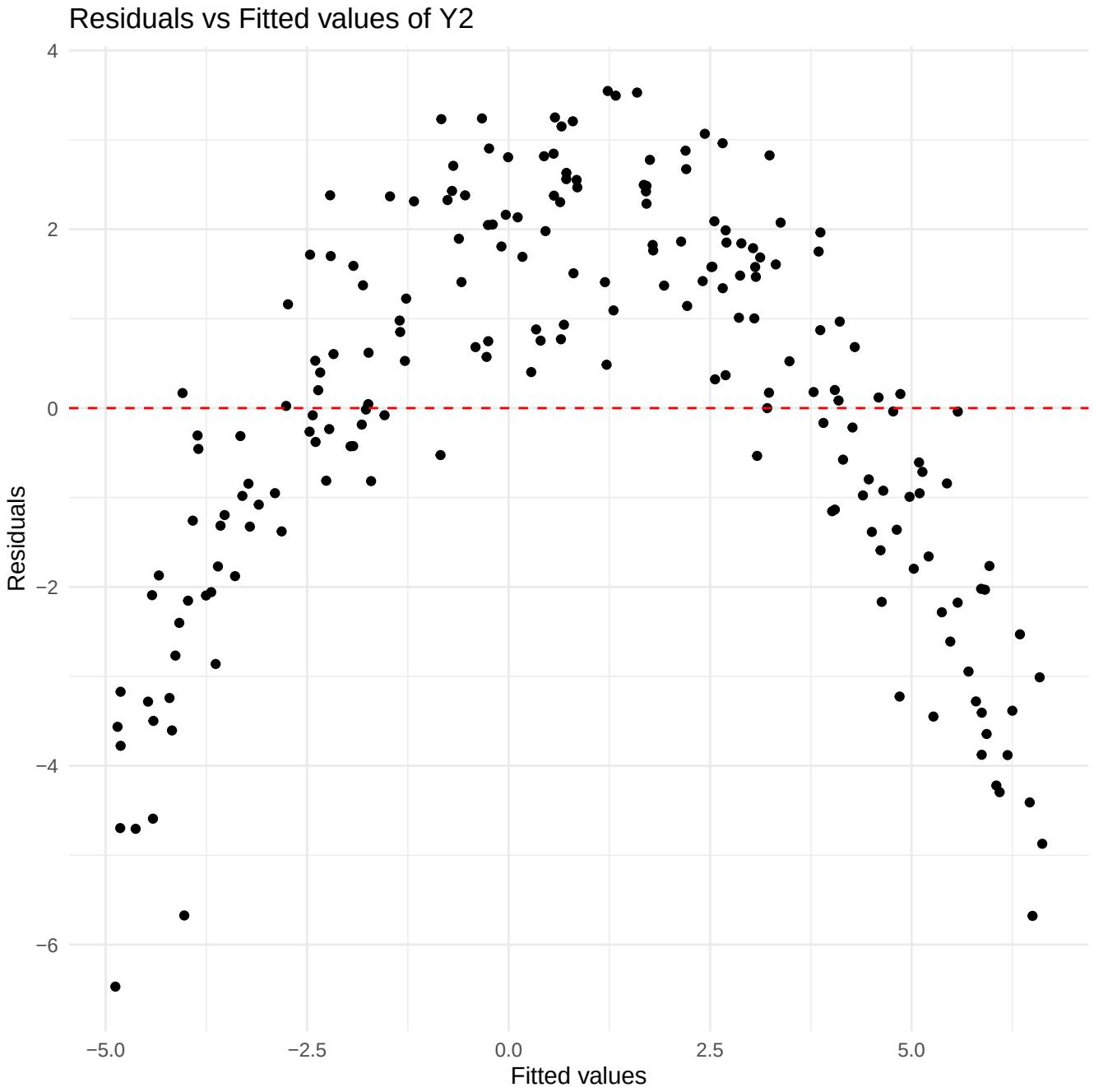


As we can see, not a perfect linear fit. There is a clear trend that the sd for residuals increases as fitted values increase.

Scatterplot for Residuals vs Fitted values of Y2

```
# HELP. FITTED VALUE BEING THE X AXIS VALUES? SHOULDN"T IT BE Y?
plot_data <- data.frame(
  fitted = fitted(lr_model_2),
  residuals = residuals(lr_model_2)
)

# Create the correct residual plot
```

```
ggplot(plot_data, aes(x = fitted, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(x = "Fitted values",
       y = "Residuals",
       title = "Residuals vs Fitted values of Y2") +
  theme_minimal()
```

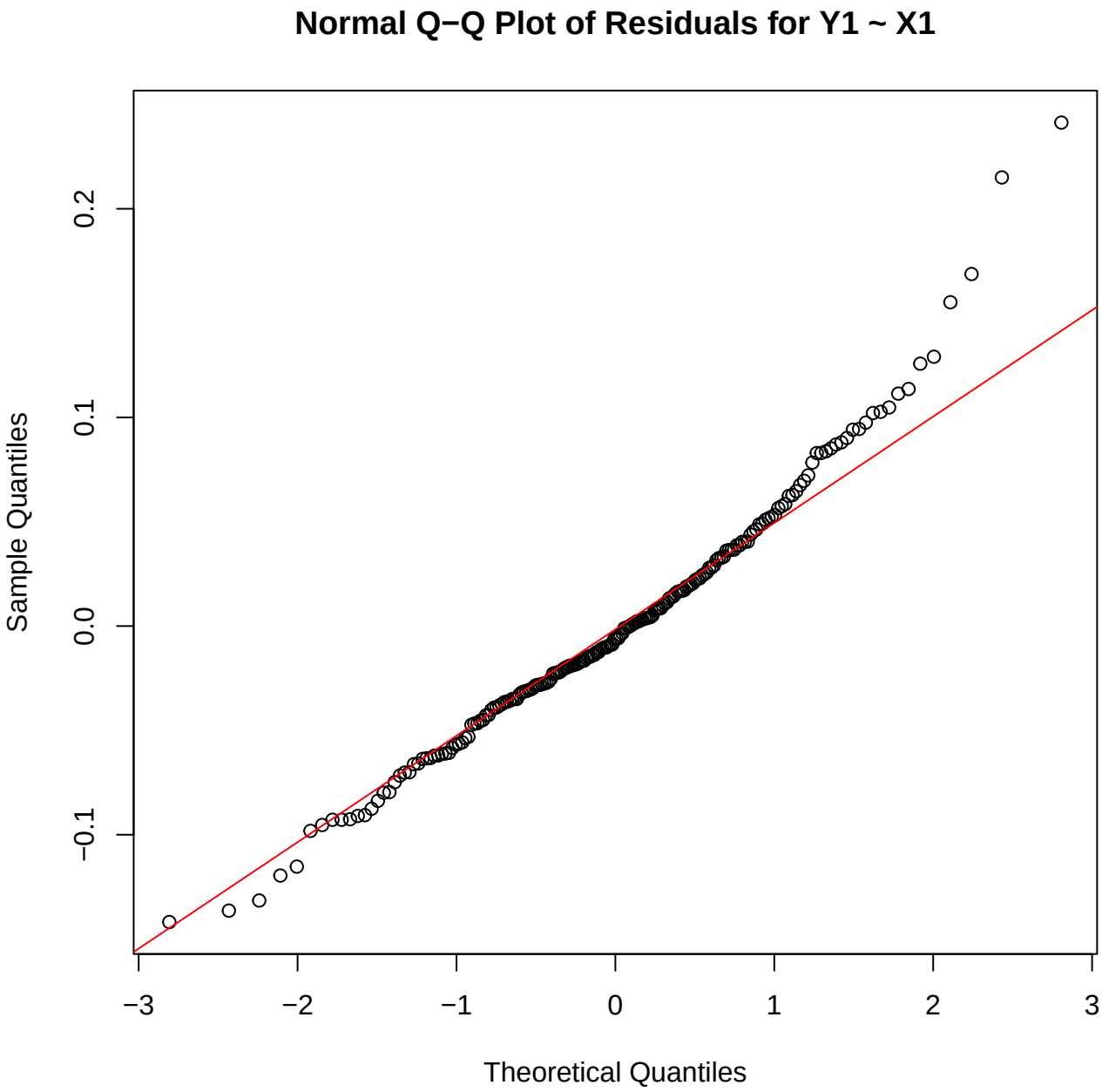


Even more apparent here, not a good linear fit. We can clearly see a curvature trend in the plot therefore we would need to perform transformations to get a proper fit before making linear predictions.

Q-Q plot of lr_model_1:

```
residuals <- residuals(lr_model_1)

# Create the Q-Q plot
```

```
qqnorm(residuals, main = "Normal Q-Q Plot of Residuals for Y1 ~ X1")
qqline(residuals, col = "red")
```

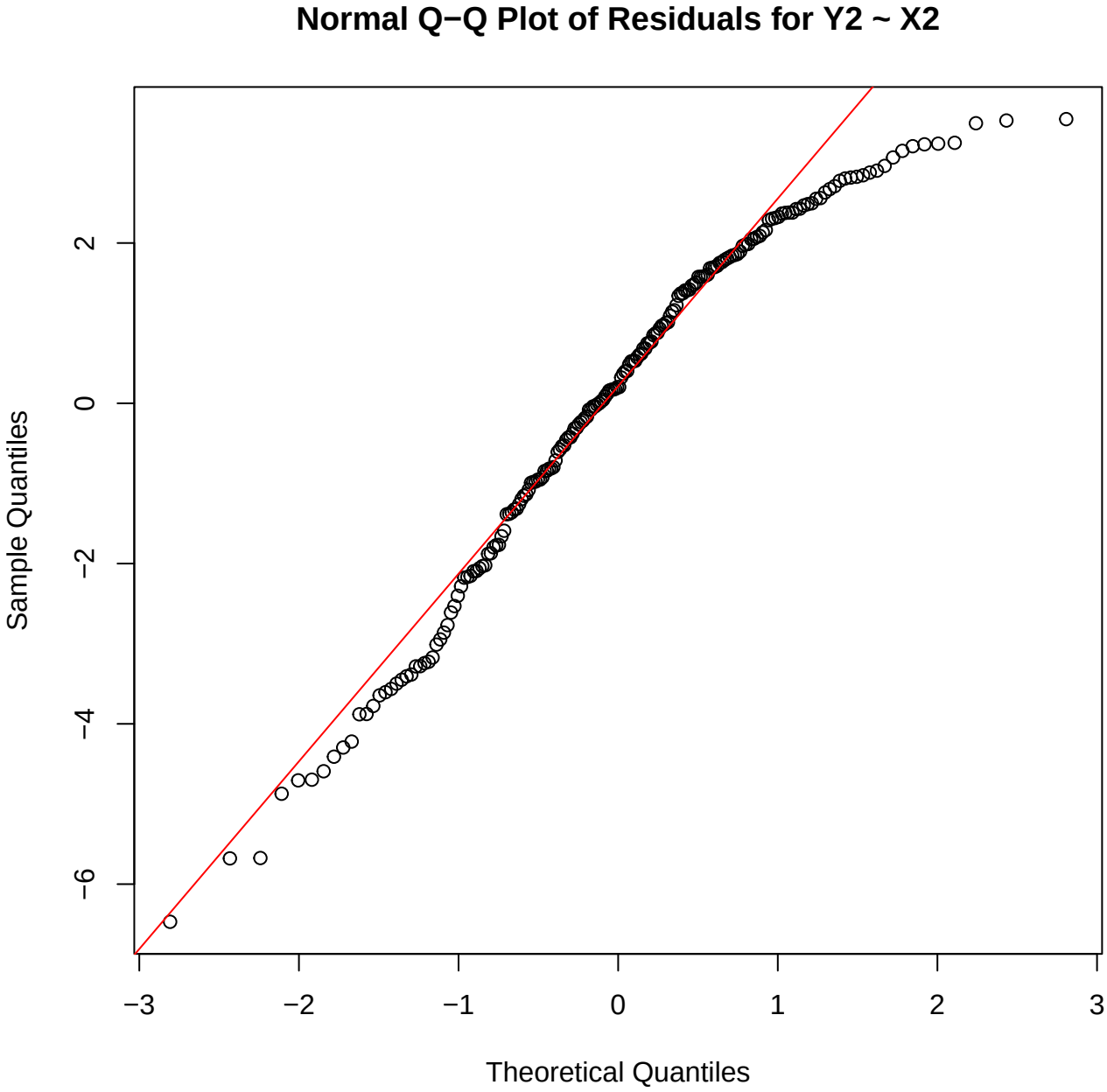



```
plot_data <- data.frame(residuals = residuals(lr_model_1))
```

We can see that the left and right ends of the plot deviate from the line indicating that the data has more extreme values than what is expected for a normal distribution.

Q-Q plot of lr_model_2:

```
residuals <- residuals(lr_model_2)

# Create the Q-Q plot
qqnorm(residuals,  main = "Normal Q-Q Plot of Residuals for Y2 ~ X2")
qqline(residuals, col = "red")
```



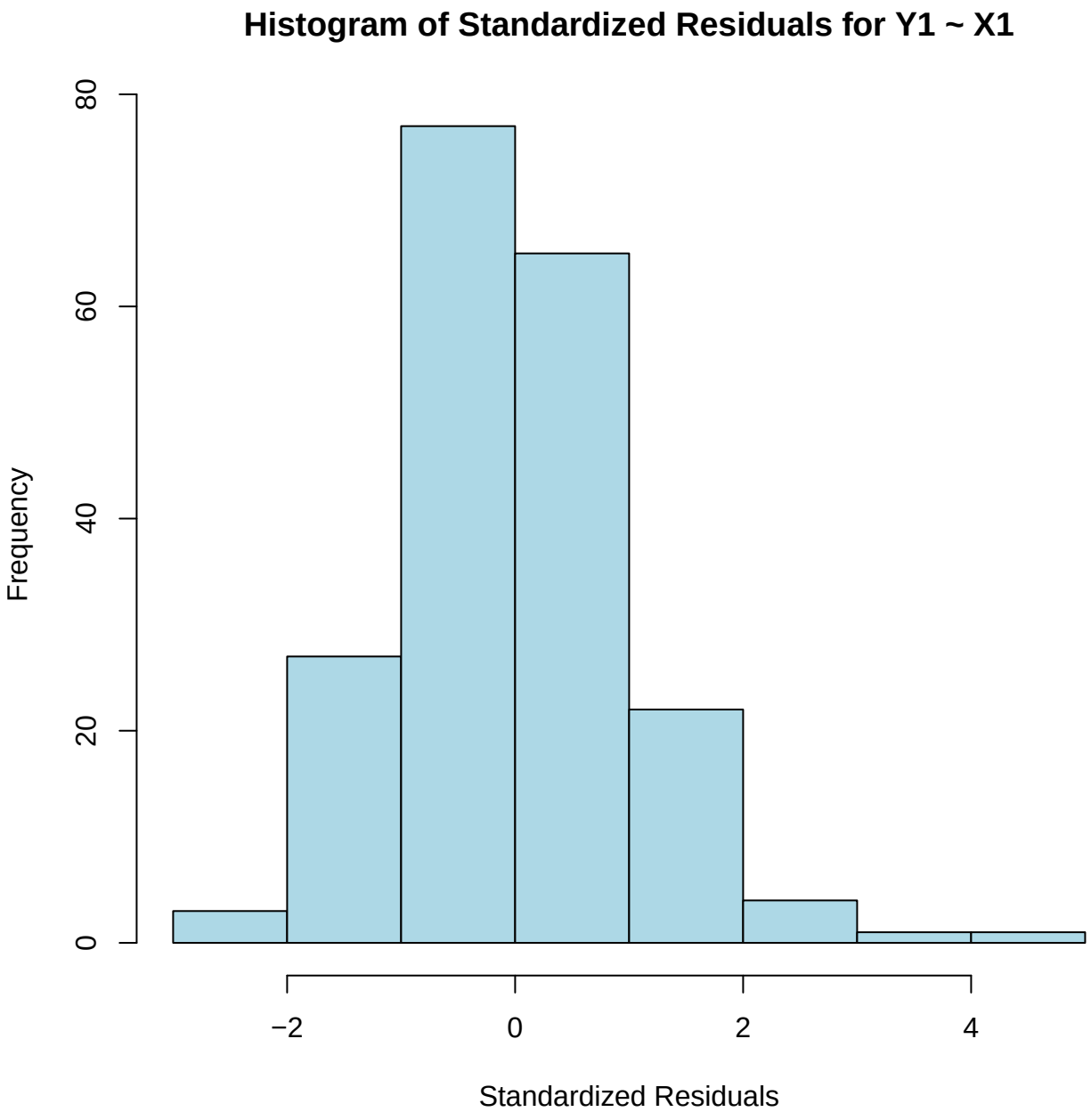

```
plot_data <- data.frame(residuals = residuals(lr_model_2))
```

We can see the right ends of the data separating from the diagonal line, specifically splitting downwards. When the points are below the diagonal line, it implies that the sample residuals is less than what the residuals should be compared to a normal distribution. Since this appears most evidently on the right side, this suggests the positive residuals is less than what they would be if predicted by a normal distribution. Therefore the data is skewed to the right.

Create a histogram of standardized regression residuals

```
residuals <- residuals(lr_model_1) #Get our residuals from the model

# Standardize the residuals
std_residuals <- rstandard(lr_model_1)

# Create a histogram
hist(std_residuals,
     main = "Histogram of Standardized Residuals for Y1 ~ X1",
```

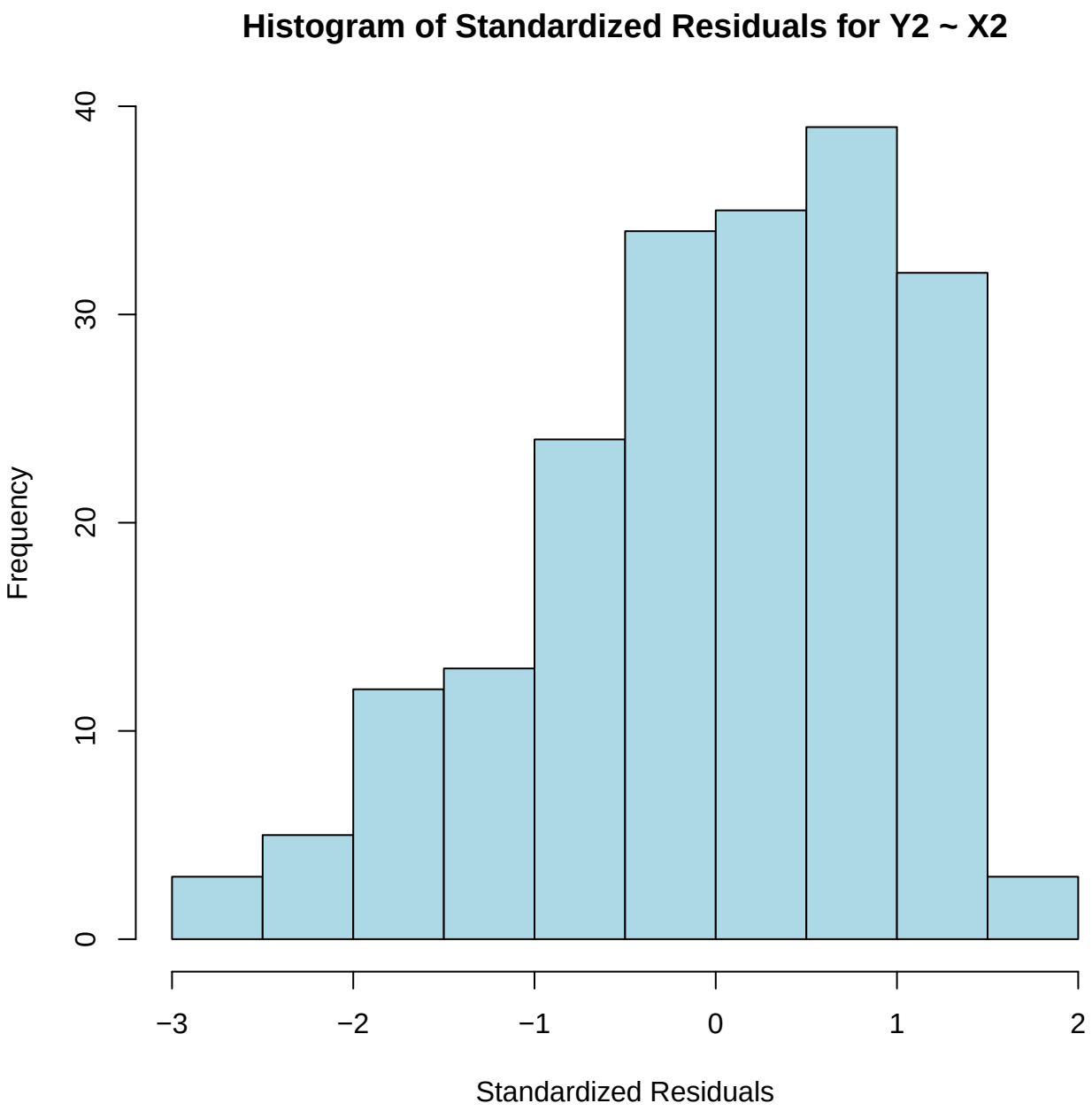
```
xlab = "Standardized Residuals",
ylab = "Frequency",
col = "lightblue",
border = "black")
```



As we predicted, there are heavy tails in this histogram. A normal distribution shouldn't have as many extreme values as what is found here.

```
# Extract residuals from the model
residuals <- residuals(lr_model_2)

# Standardize the residuals
std_residuals <- rstandard(lr_model_2)

# Create a histogram
hist(std_residuals,
```

```
main = "Histogram of Standardized Residuals for Y2 ~ X2",
xlab = "Standardized Residuals",
ylab = "Frequency",
col = "lightblue",
border = "black")
```



Follows our prediction from the Q-Q plot. The data is indeed skewed toward the right, therefore residuals in the right is less than what they should be in a normal distribution.

c.)

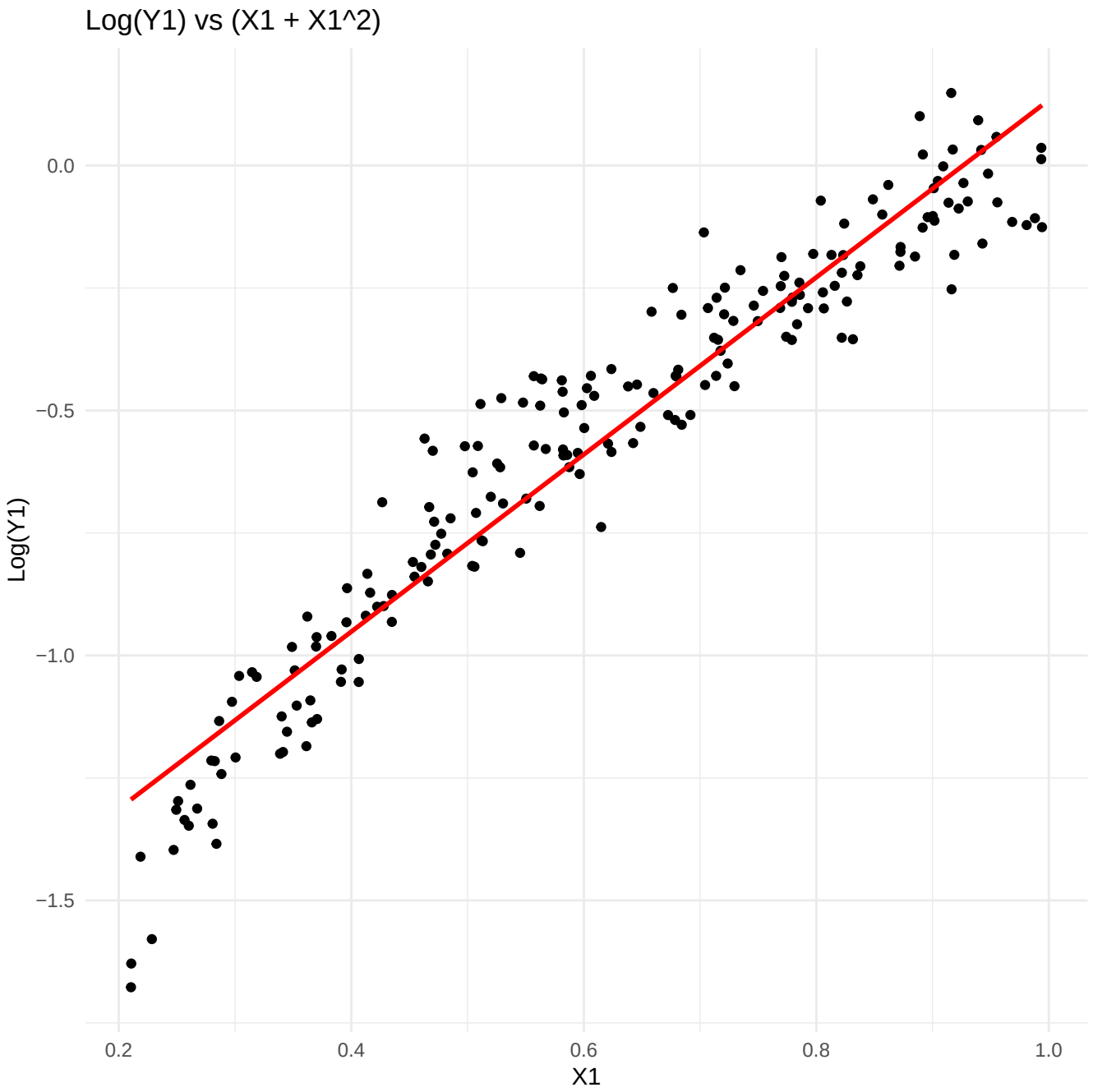
Going back to the residual plot of Y1 ~ X1 we see trumpet shape, meaning the bunching of the y's in the left and spreading out moving right, which implies that we have a nonconstant variance. This violates the assumption that all our error terms have the same variance. Additionally, the variance of the residuals is increasing along the x axis, and so this violates the assumption that the error terms are independent of x as well.

Now looking at the residual plot of Y2 ~ X2 we clearly see a negative polynomial curvature throughout the residuals moving along the x axis. This violates linearity assumption in that the relationship between X2 and Y2 are not linear. The data will need to first be transformed by a log function since the long function has a concave shape.

d.)

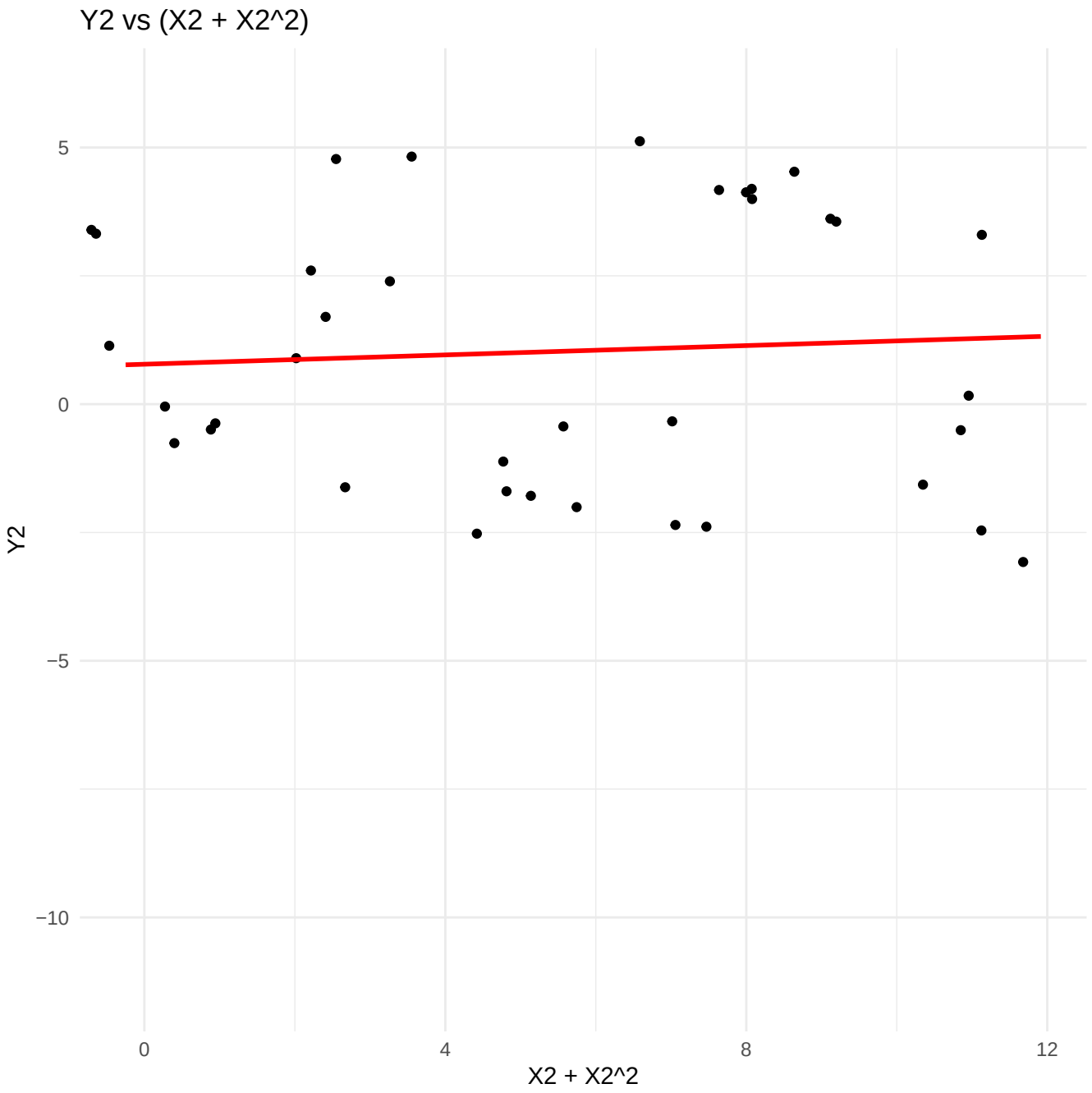
First to transform Y1 ~ X1. I'll take log(Y1) as Y1 only consists of positive values. Additionally I will introduce a polynomial term to capture any potential nonlinear relationship in the data. Looking at the scatterplot we can see a much better linear fit.

```
log_model_1 <- lm(log(Y1) ~ X1 + I(X1^2), data = nonlinear_data)

ggplot(nonlinear_data, aes(x = X1, y = log(Y1))) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, color = "red") +
  labs(title = "Log(Y1) vs (X1 + X1^2)",
       x = "X1",
       y = "Log(Y1)") +
  theme_minimal()
```



Next to transform Y2 ~ X2

```
log_model_2 <- lm(Y2 ~ X2 + I(X2^2), data = nonlinear_data)

ggplot(nonlinear_data, aes(x = X2 + I(X2^2), y = Y2)) +
  geom_point() +
  geom_smooth(method = "lm",formula = y ~ x, se = FALSE, color = "red") +
  labs(title = "Y2 vs (X2 + X2^2)",
       x = "X2 + X2^2",
       y = "Y2") +
  theme_minimal()
```

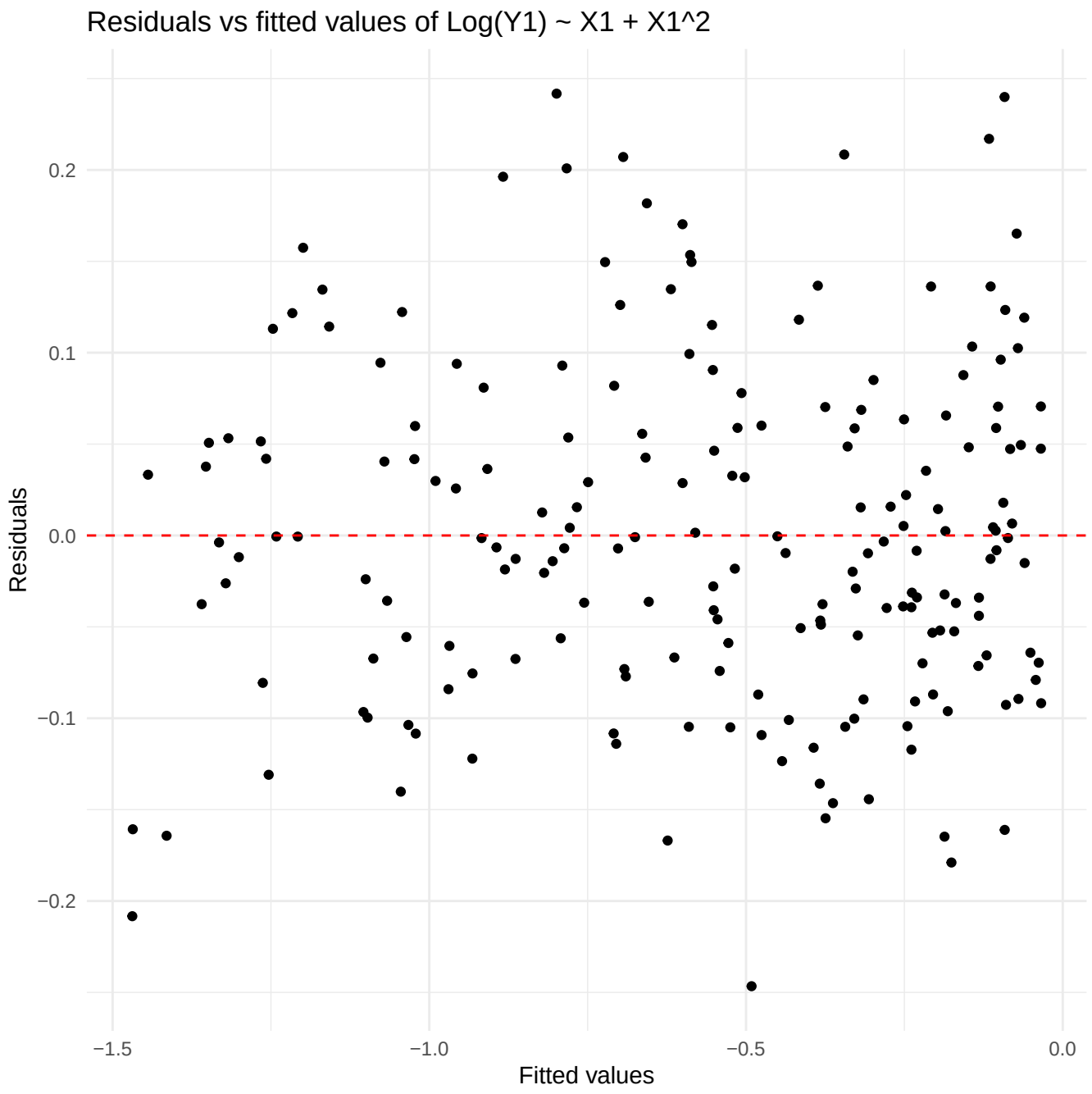


e.)

Here is the residual plot for revised model of Log(Y1) ~ X1 + I(X1^2)

```
plot_data <- data.frame(
  fitted = fitted(log_model_1),
  residuals = residuals(log_model_1)
)

# Create the correct residual plot
ggplot(plot_data, aes(x = fitted, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(x = "Fitted values",
       y = "Residuals",
```

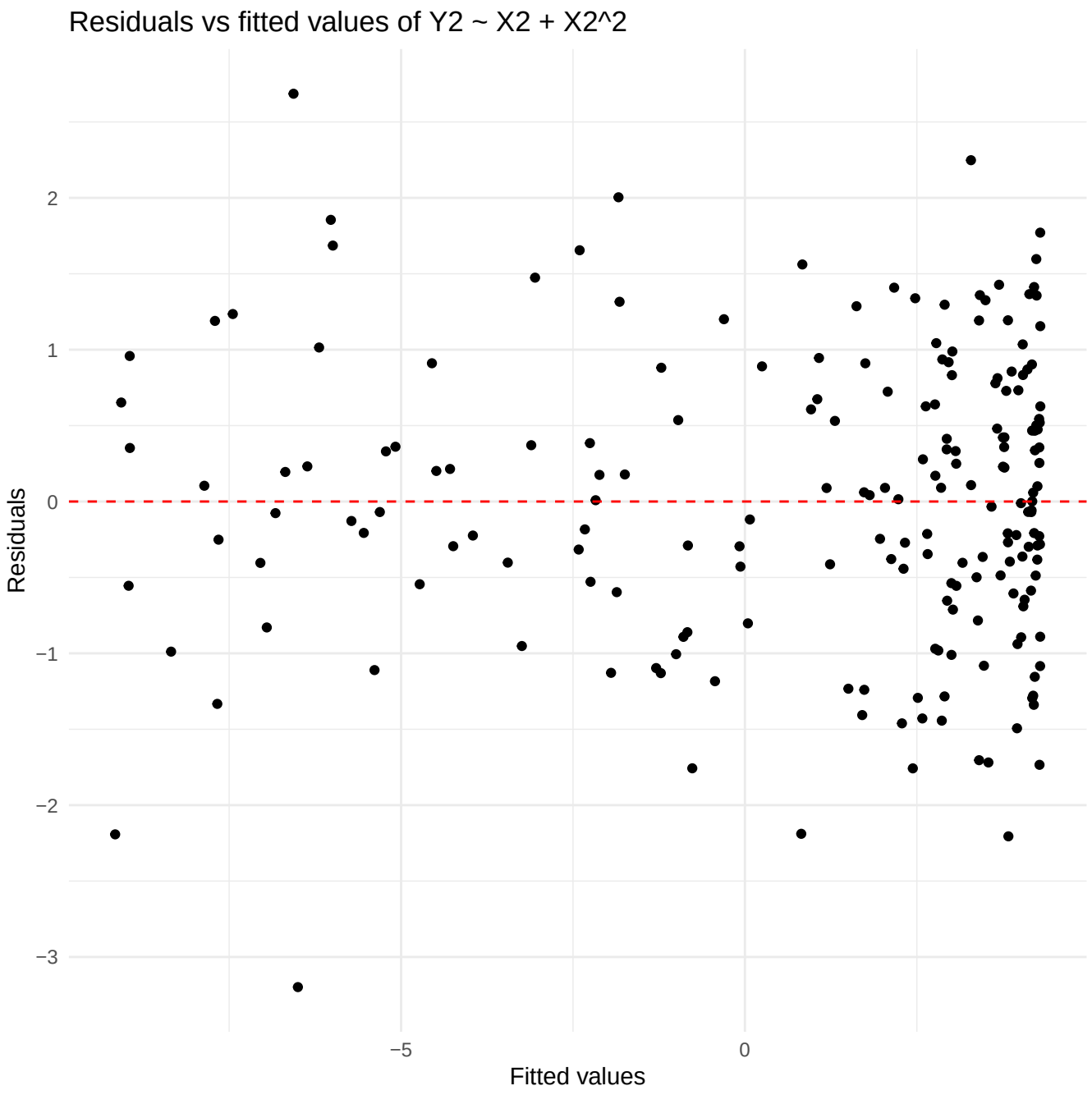
```
      title = "Residuals vs fitted values of Log(Y1) ~ X1 + X1^2") +
  theme_minimal()
```



Here is the residual plot for revised model Y2 ~ X1 + I(X1^2)

```
plot_data <- data.frame(
  fitted = fitted(log_model_2),
  residuals = residuals(log_model_2)
)

# Create the correct residual plot
ggplot(plot_data, aes(x = fitted, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(x = "Fitted values",
       y = "Residuals",
```

```
     title = "Residuals vs fitted values of Y2 ~ X2 + X2^2") +
theme_minimal()
```



As we can see, both residual plots display no apparent curvature to the data. Therefore, a lack of a curve suggests that it is likely that the linear assumptions of the model now hold.