Homework 3: Harder Simple Linear Regression

Ruonan Zhao

September 12, 2024

Instructions:

The purpose of this homework assignment is to look more into assessing conditions, looking for outliers/influential points, and performing transformations. Make sure you read each question carefully. In each question, I will give you a task to do, and I will tell you what I want you to output. You can write as much code as you want in each code chunk, but make sure you complete the task and only print the output I asked you to print. You should still include all your code in your knitted PDF file (do not hide your code). Don't sort the data unless you are told to sort the data. You should remove the "#" sign in each code chunk before writing your code. Also, if you see the comment "#DO NOT CHANGE", then I don't want you to make any modifications to that code. **You should knit your RMD file to a PDF after you answer every question.**

After you are done, knit the RMarkdown file to PDF and submit the PDF to Gradescope under Homework 3.

Questions

Q1 (2 Points)

Simulate data in R using the code provided below. We sample random values for X from a uniform distribution between 0 and 10, and compute Y using a fixed slope and intercept. We add an error that is randomly sampled from a gamma distribution to ensure that there is not a perfect linear relationship between X and Y. The gamma distribution is a theoretical distribution for a variable that is nonnegative. Therefore, all of the errors would tend to be positive, unless multiplied by a negative number. Thus, we use the sample function to modify the positive error to be negative 15% of the time.

Create a dataframe called **DATA1** using the two vectors: **X** and **Y**. Ensure that the column names of **DATA1** are "X" and "Y" so we know which variable is the response and which variable is the predictor. Display the first 12 rows of the dataframe. This should be the only output.

I would recommend learning about the function named `data.frame()`. The website https://statisticsglobe.com/data-frame-function-r would be helpful reading.

The `set.seed(120)` code hopefully will ensure that every student in the class has the same sample when the entire code chunk is run.

```r
set.seed(120) #DO NOT CHANGE

X = runif(60, 0, 10) #DO NOT CHANGE
beta0 = -3 #DO NOT CHANGE
beta1 = 6 #DO NOT CHANGE
```

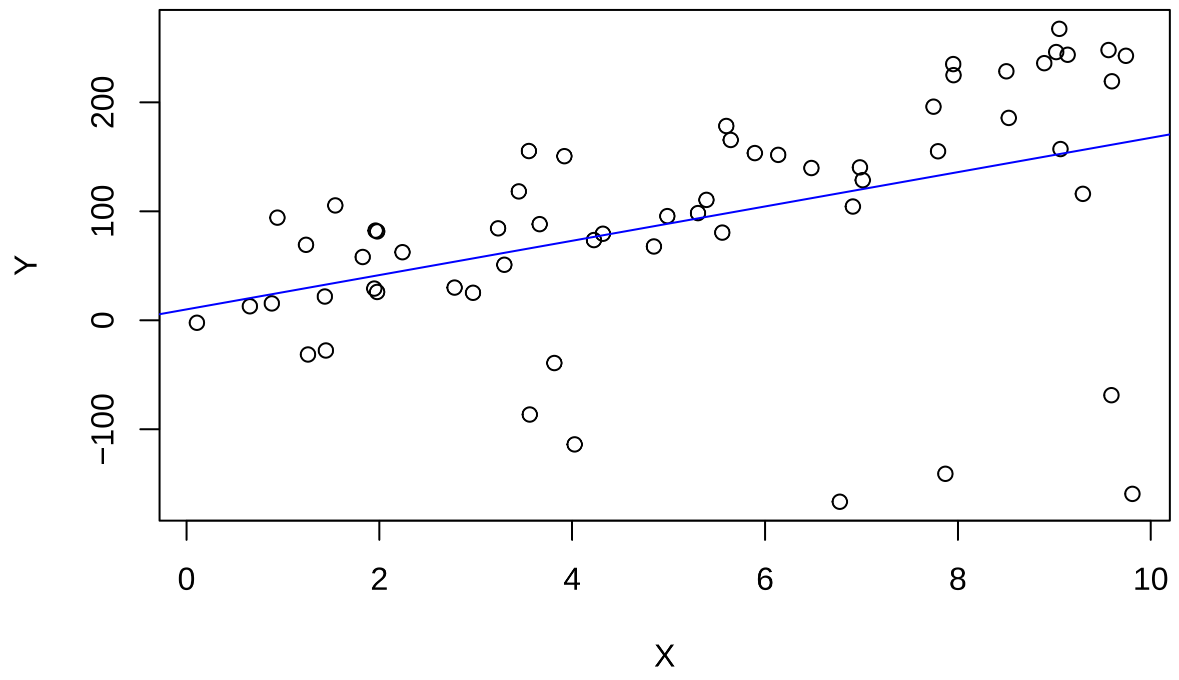
```r
Y = beta0 + beta1 * X +
    rgamma(60, shape = 1 * X, scale = 20) * sample(c(-1, 1), size = 60,
                                              replace = T, prob = c(0.15, 0.85)) #DO NOT CHANGE

DATA1 = data.frame(X, Y)
head(DATA1, 12)
```

```
##            X          Y
## 1  3.9190768  150.60408
## 2  1.2601679  -31.38987
## 3  3.4461308  118.25927
## 4  7.9549613  224.93908
## 5  1.9601551   82.30093
## 6  4.8471608   67.73085
## 7  9.8101690 -159.29968
## 8  0.9426554   94.18180
## 9  8.5275468  185.71398
## 10 9.1381900  243.65963
## 11 5.8938510  153.43707
## 12 9.5972095  219.30577
```

Q2 (2 Points)

Fit a linear regression model using Y as the dependent variable and X as the independent variable. Then, create a scatter plot of Y vs X with the fitted regression line added to the plot. The only output should be the graphic.

```r
model = lm(Y ~ X)
plot(X, Y)
abline(model, col="blue")
```

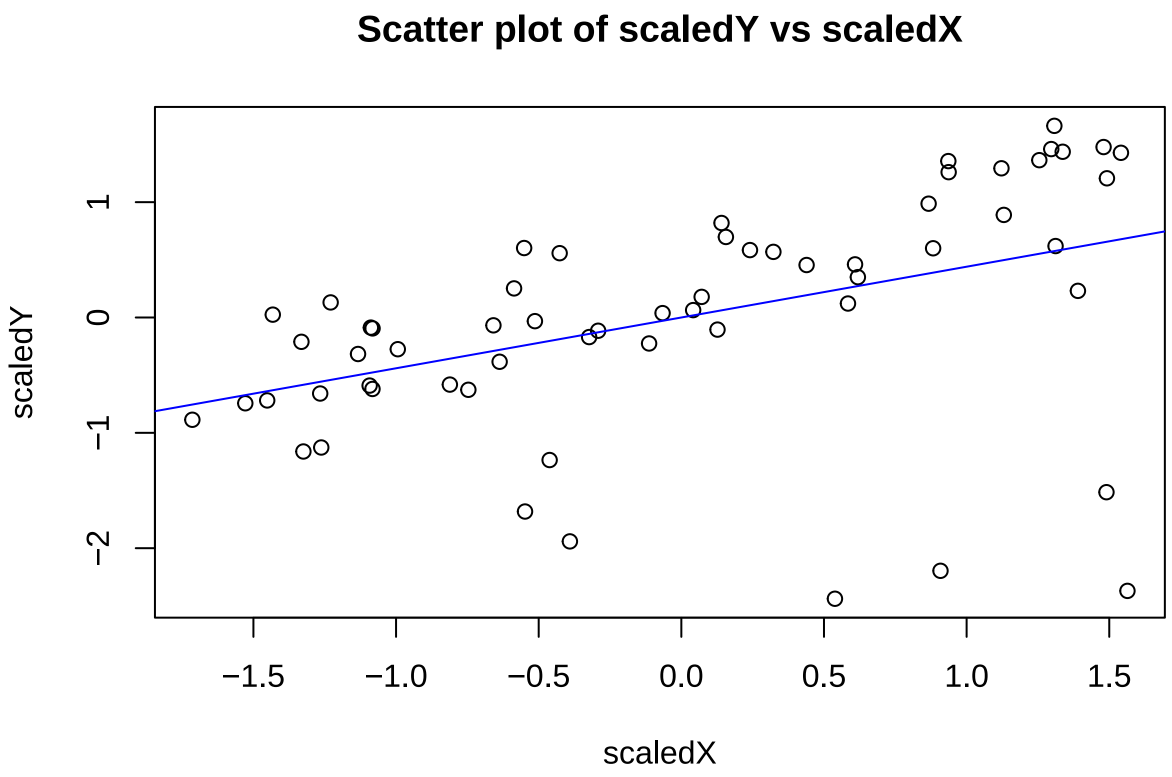


Q3 (3 Points)

Create standardized versions of the variables "X" and "Y", named "scaledX" and "scaledY", respectively. Standardize by subtracting the mean (centering) and then dividing by the sample standard deviation (scaling) for each variable. You can look into using the `scale()` function in R to do this.

After creating these variables, plot a scatter plot of *scaledY* vs *scaledX* with the fitted regression line. Reflect on how standardization affects the scatter plot and the regression line. What are the differences between your plot in Q2 and your plot in Q3? How did standardizing the data impact the scatter plot and regression line? Provide your answer in the designated response area below the code chunk.

```r
scaledX = scale(DATA1$X)
scaledY = scale(DATA1$Y)

plot(scaledX, scaledY, main="Scatter plot of scaledY vs scaledX")
abline(lm(scaledY ~ scaledX), col="blue")
```



Response in Complete Sentences: Different: more clearly lable of x axis and y axis, the scales on both axes are consistent, ranging roughly between -2 and 2. Standardization does not change the relationship between data points or the slope of the regression line, but after standardization, the range of X and Y becomes consistent, making it easier to compare data on different scales, and more clear for someone who need this plot.

Q4 (2 Points)

Calculate the natural logarithm of each value of the variable Y in the data. Print the vector of log-transformed Y values. Your only output should be a vector of the log of each of the 60 values for Y in the simulated dataset.

If you did this correctly, you will see "NaN" for some values. In the appropriate space below, explain why you got "NaN" for some calculations.

Note, you can run the `log()` function on a full vector of numbers. For example, try running the code `log(c(1,2,3,4))` and notice you get the vector `c(log(1),log(2),log(3),log(4))`.

```
log_Y = log(DATA1$Y)
```

```
## Warning in log(DATA1$Y): NaNs produced
```

```
print(log_Y)
```

```
##  [1] 5.014654      NaN 4.772879 5.415830 4.410382 4.215542      NaN 4.545227
##  [9] 5.224208 5.495772 5.033291 5.390467 4.061650 3.371020 5.513427 3.401620
## [17] 4.704866 4.387837      NaN 4.401357 4.939670 4.298775      NaN 5.431498
## [25] 4.648243 3.930682 2.738986 4.435237 3.227589 5.045464      NaN 4.588476
## [33]      NaN 4.753656      NaN      NaN 5.044084 4.374551 3.257863 4.943901
## [41] 5.491857 4.237795      NaN 5.463608 4.560038 5.505846 4.479774 5.057019
## [49] 5.588891 5.183467 4.657771 5.022578 4.856669 5.460025 5.108773 5.278198
## [57] 2.552902 3.082177      NaN 4.134643
```

```r
log(c(1, 2, 3, 4))
```

```
## [1] 0.0000000 0.6931472 1.0986123 1.3862944
```

Response in Complete Sentences: The NaNs are coming from negative(NaN) or zero(-Inf) values in the original Y data.

Q5 (7 Points)

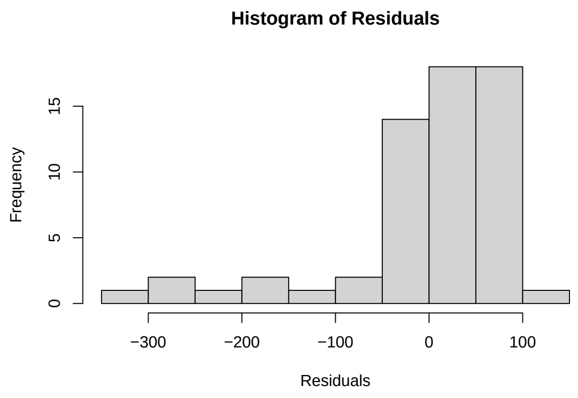
Using the linear regression model of Y on X from Q2, save the fitted values as a new column (variable) **fitted** and the residuals as a new column (variable) **resids** in DATA1. Create the following visuals based on this simple linear regression:

1. A histogram of the residuals with 15 breaks using the `hist()` function.
2. A scatter plot of residuals vs the predictor variable X with a horizontal reference line at 0.
3. A scatter plot of fitted values vs actual Y values. Add a 45-degree reference line with a y-intercept of 0 and a slope of 1 using the `abline()` function.
4. A normal quantile plot of the residuals with a reference line.

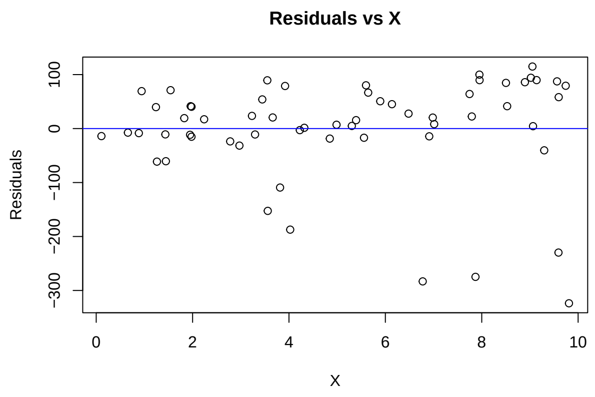
The only output from your code should be the four visuals.

After inspecting the plots, identify three potential issues with the regression model and explain each in a sentence. Write three sentences. Each sentence should be about a different potential problem. You should reference the figure (for example, "in figure 3 . . . " or "in the normal quantile plot"), you should reference the potential problem, and explain why that figure indicates that their may be a potential problem in our model.

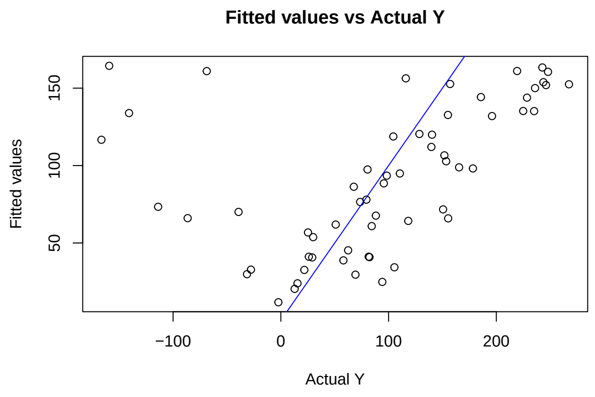
```r
DATA1$fitted = model$fitted.values
DATA1$resids = model$residuals

#1 A histogram of the residuals
hist(DATA1$resids, breaks = 15, main = "Histogram of Residuals", xlab = "Residuals")
```



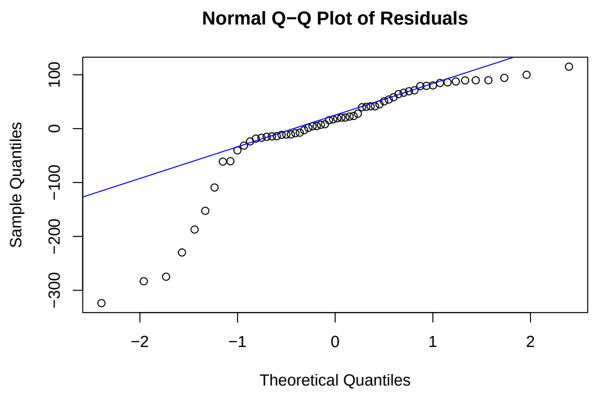
```
#2 A scatter plot of residuals vs the *X*
plot(DATA1$X, DATA1$resids, main = "Residuals vs X", xlab = "X", ylab = "Residuals")
abline(h = 0, col = "blue")
```



```
#3 A scatter plot of fitted values vs actual *Y*
plot(DATA1$Y, DATA1$fitted, main = "Fitted values vs Actual Y", xlab = "Actual Y", ylab = "Fitted value
abline(0, 1, col = "blue")
```



```
#4 A normal quantile plot of the residuals
qqnorm(DATA1$resids, main = "Normal Q-Q Plot of Residuals")
qqline(DATA1$resids, col = "blue")
```



Response in Complete Sentences: In figure 1, it is skewed, some residuals in the left are as low as -300, indicating that the residuals are not symmetrically distributed, this could indicate non-normal residuals.

In figure 2, there seems to be more variation in the residuals at lower values of X, with residuals becoming less spread out as X increases, this could indicate heteroscedasticity or non-linearity in the data.

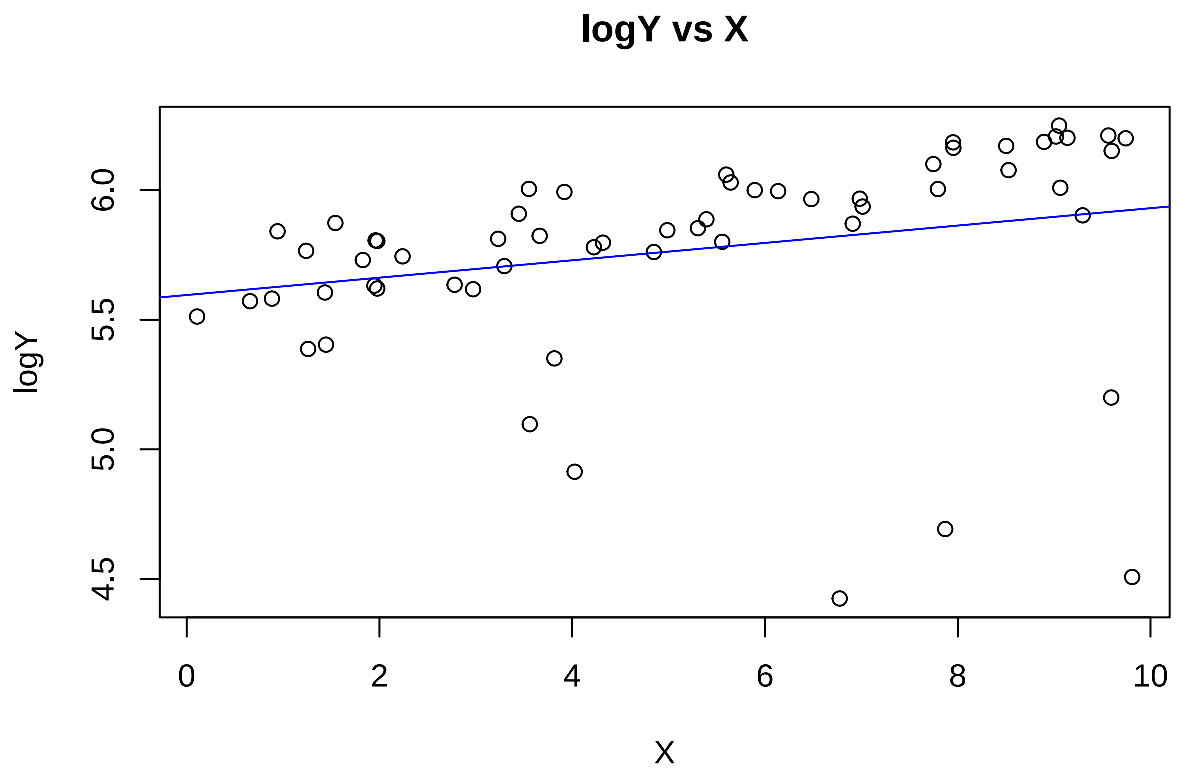
In figure 4, the residuals deviate from the straight line, particularly at the lower (left) and upper (right) ends of the plot, indicate non-normality of the residuals.

Q6 (3 Points)

To address the issues from Q5 in the simple linear regression model, we can try a log transformation on Y. However, we cannot just calculate `log(DATA1$Y)` due to problems seen in Q4. Instead, we will add a constant C=*250* to Y before taking the log.

In DATA1, create a new variable "logY" by adding a constant of C=*250* to Y before taking the logarithm. Then, fit a linear regression model of $logY$ on X and create a scatter plot with a regression line similar to Q2 and Q3 using $logY$ instead of Y.

```r
DATA1$logY = log(DATA1$Y + 250)

model_logY = lm(logY ~ X, data = DATA1)

plot(DATA1$X, DATA1$logY, main = "logY vs X", xlab = "X", ylab = "logY")
abline(model_logY, col = "blue")
```

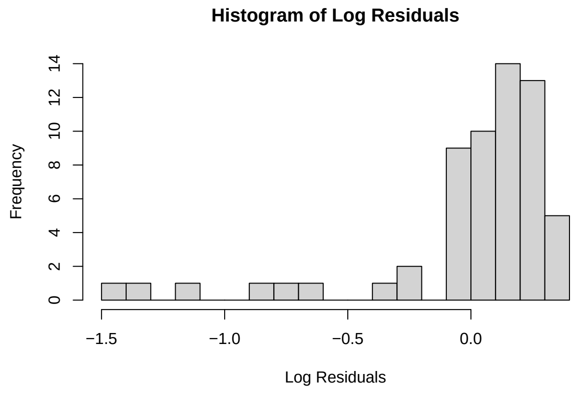


Q7 (4 Points)

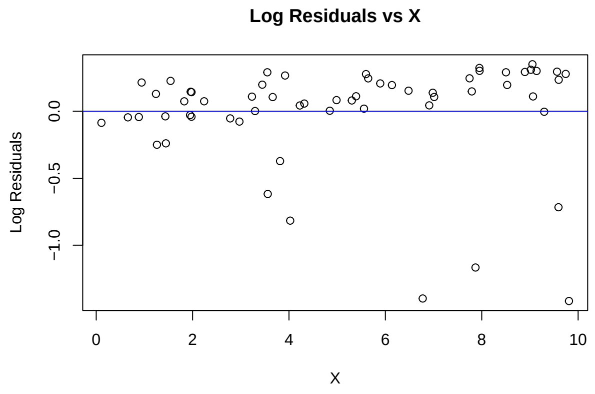
Store the fitted values and residuals into DATA1 from the log-transformed regression model as "logFitted" and "logResids", respectively. Recreate the four plots from Q5 using these new these fitted values and residuals. Think about whether or not this transformation helped in modifying the data so we have less problems with the outliers and the conditions of a simple linear regression.

Think about the fact that "logFitted" is a prediction for the log transformed Y and not the original variable Y. We will address this later.

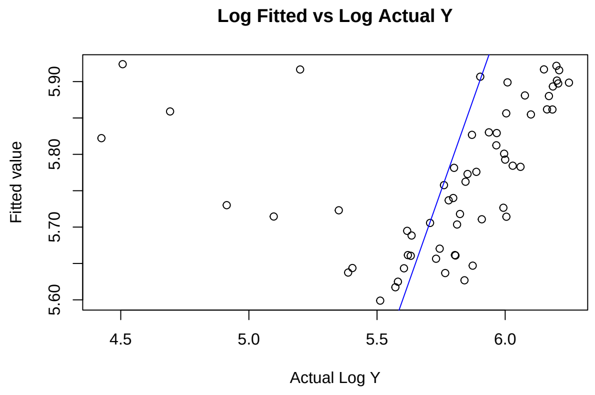
```
DATA1$logFitted = model_logY$fitted.values
DATA1$logResids = model_logY$residuals

hist(DATA1$logResids, breaks = 15, main = "Histogram of Log Residuals", xlab = "Log Residuals")
```



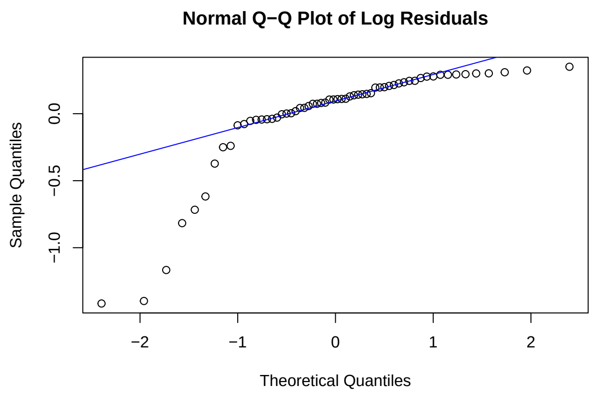
```r
plot(DATA1$X, DATA1$logResids, main = "Log Residuals vs X", xlab = "X", ylab = "Log Residuals")
abline(h = 0, col = "blue")
```



```r
plot(DATA1$logY, DATA1$logFitted, main = "Log Fitted vs Log Actual Y", xlab = "Actual Log Y", ylab = "Fi
abline(a = 0, b = 1, col = "blue")
```



```r
qqnorm(DATA1$logResids, main = "Normal Q-Q Plot of Log Residuals")
qqline(DATA1$logResids, col = "blue")
```



Q8 (3 Points)

Calculate and display the sum of squared errors (SSE) for both the original regression model of Y on X and the log-transformed model of $logY$ on X. Reverse the log transformation before calculating the SSE for the log-transformed model to make a fair comparison.

Note, we cannot compare the two numbers fairly since in one model we are predicting Y and in the other model we are predicting *log(Y+250)*. Hence, we need adjust or "untransform" the predictions so that we are still predicting Y. Make this adjustment and then compute the residuals manually by comparing these predictions of Y to the actual Y. This will allow you to compute SSE so that we can find out if there is evidence that use of the log transformation led to a smaller (better) SSE.

In your output, I only want to see the two numbers: SSE from the simple linear regression of Y on X and SSE from simple linear regression of $logY$ on X that is calculated after reversing the transformation.

```r
orig_residuals = DATA1$Y - model$fitted.values
SSE_orig = sum(orig_residuals^2)

log_fitted = DATA1$logFitted
log_fitted_untransformed = exp(log_fitted) - 250

log_residuals_untransformed = DATA1$Y - log_fitted_untransformed

SSE_log = sum(log_residuals_untransformed^2)

SSE_orig
```

```
## [1] 532928.5
```

```r
SSE_log
```

```
## [1] 569528.6
```