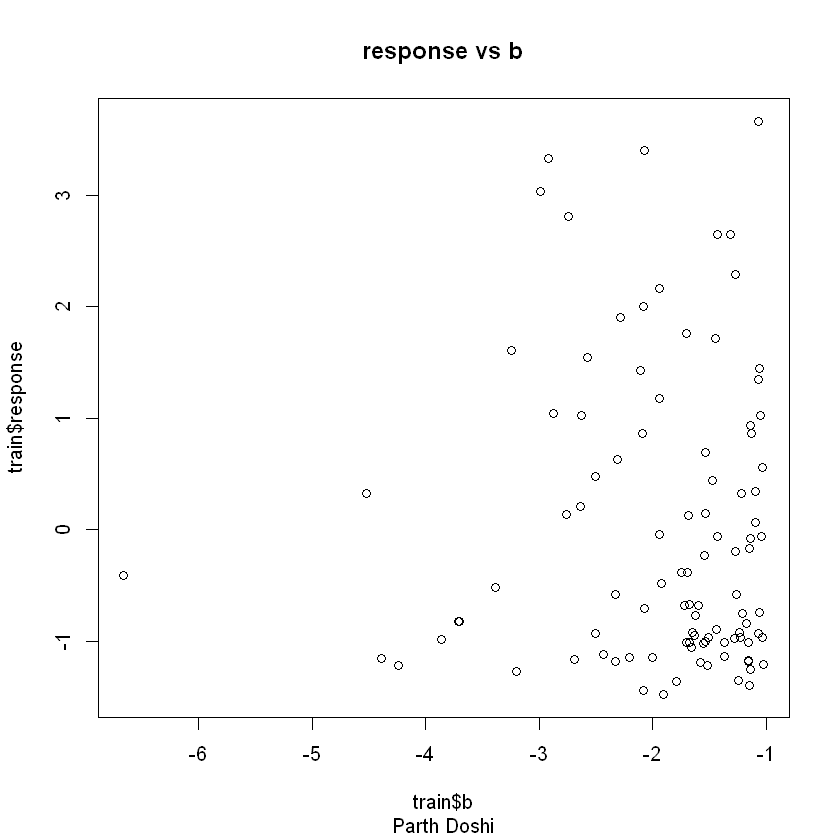
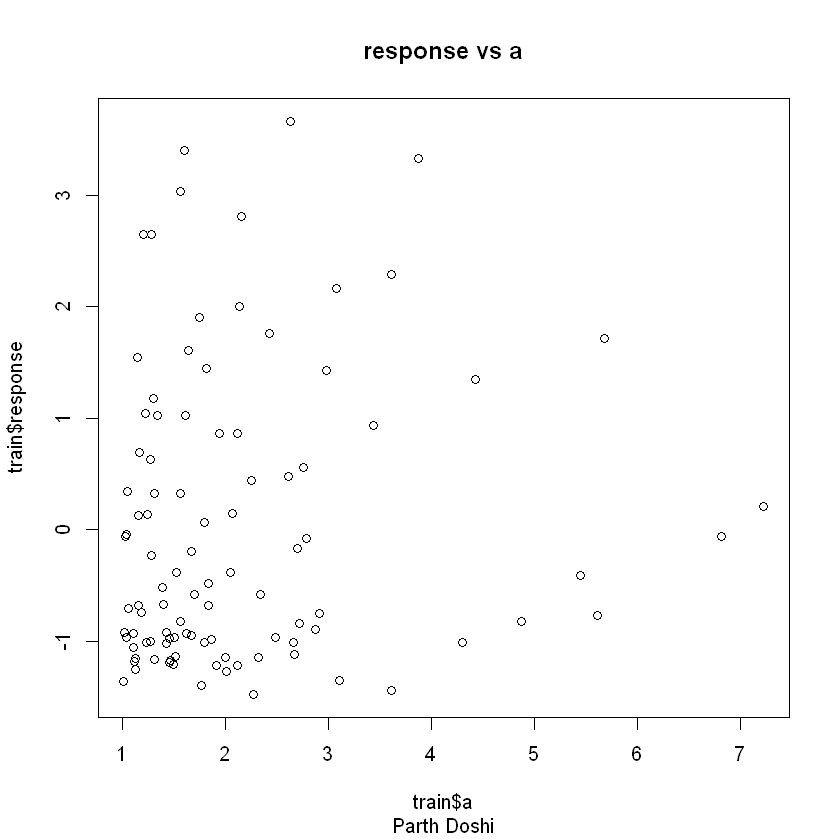
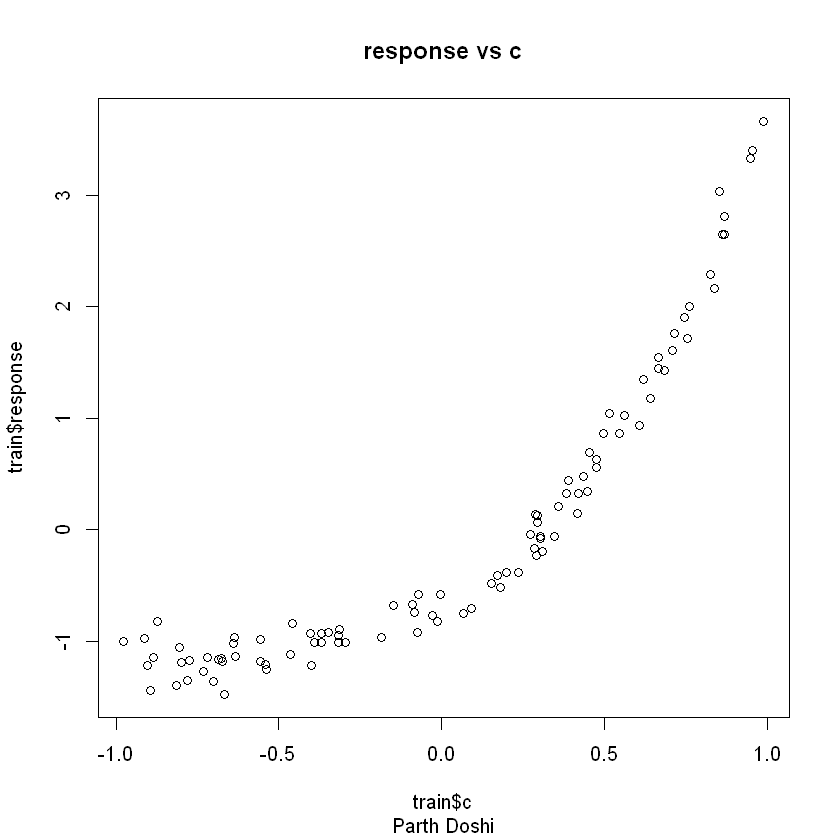
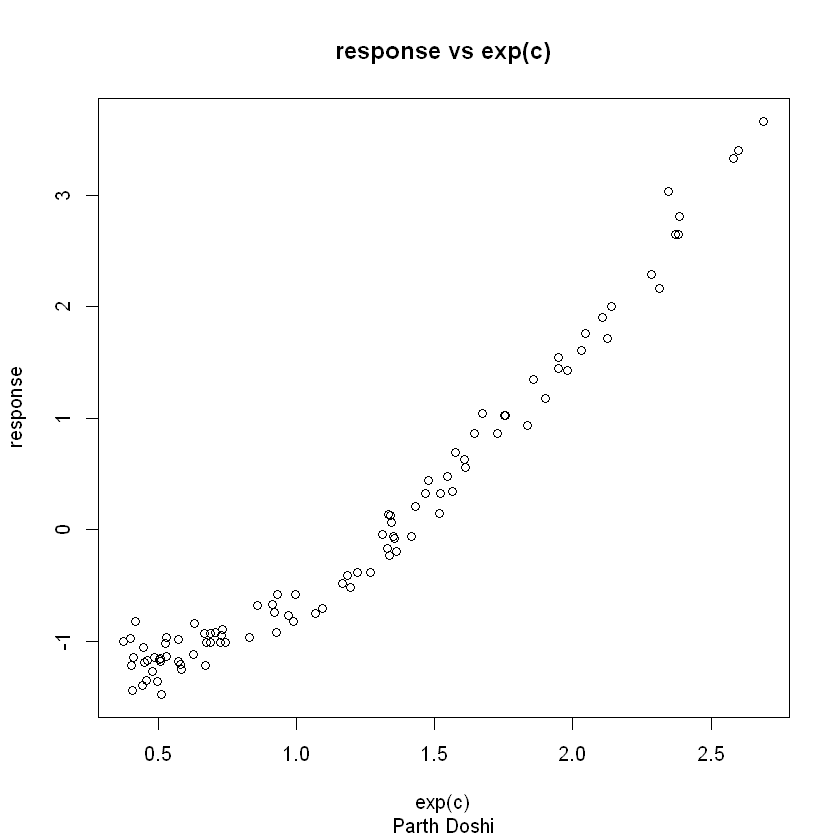
**Dataset 1:**

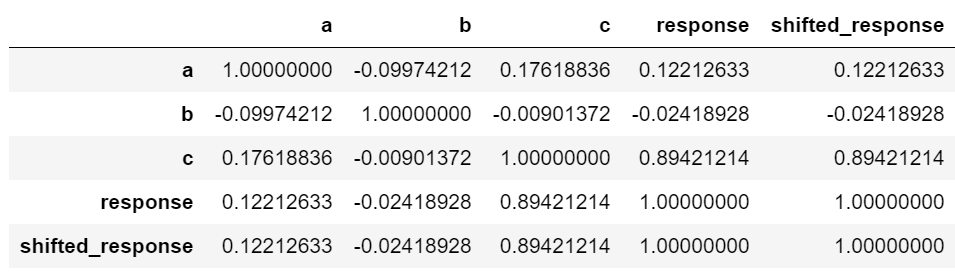
I will name all features a, b, c, etc. and response is just response.

I first went ahead and plotted all the features against the response variable to find any obvious hints.   


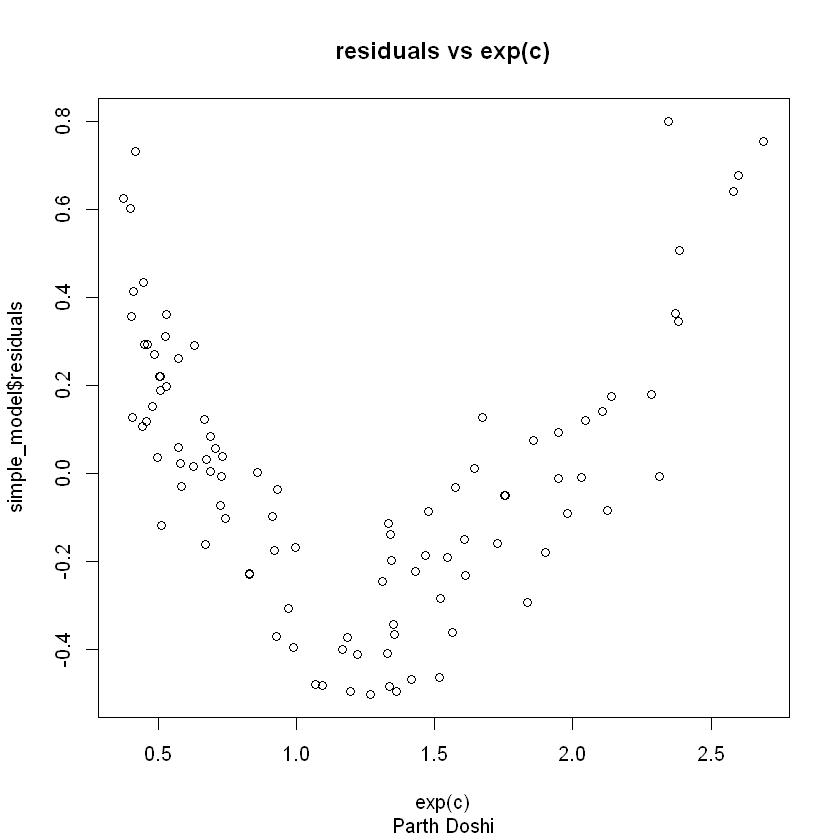
Instantly, we see that c has the largest correlation with our response variable, but transformations need to be done to make it a linear fit. Looking at the plot of c against our response, I can tell I need to run the exponential function transformation since the numbers are between -1 and 1. Plotting exp(c) against response gives me:



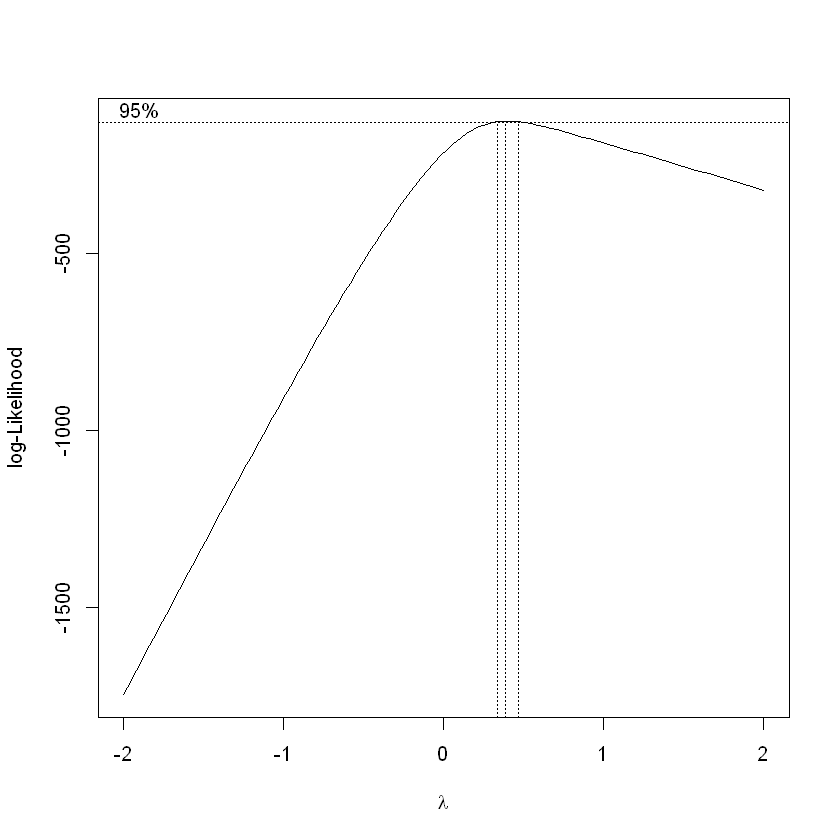
This looks much better.  
  
 I now found the correlation between a, b, c, exp(c), and the response variable:



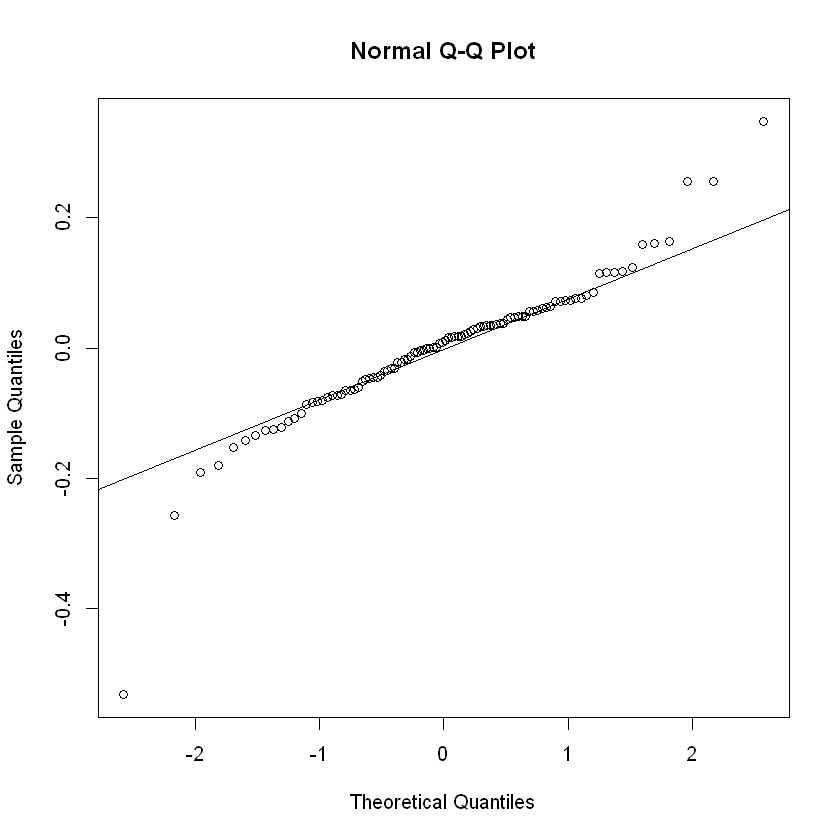
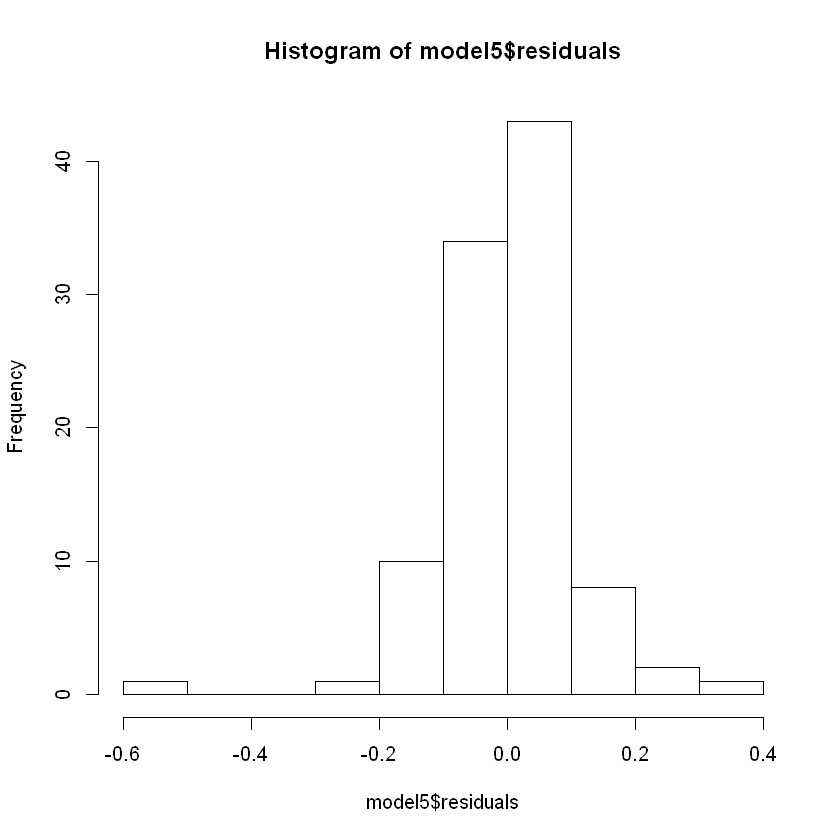
This helped to affirm my theory that c has high correlation with the response and the other variables are barely correlated. After running linear regression with the formula: , I now need to check if my assumptions hold. After plotting a histogram of the residuals, they seemed mostly normal if a little skewed to the right. However, plotting the residuals against exp(c) showed that our assumption of constant variance is not met:



To fix this, I now need to transform our response variable. I wanted to run boxcox but found that the response variable had negative values; I then shifted the response over by the absolute value of the minimum response value and added a very small epsilon to have only positive values. I now ran boxcox on these shifted response values and it returned that the square root transformation would have a high log-likelihood:



I now trained the model with the formula: . I then checked for normality and constant variance on the residuals and they seemed much better after the Y transformation.



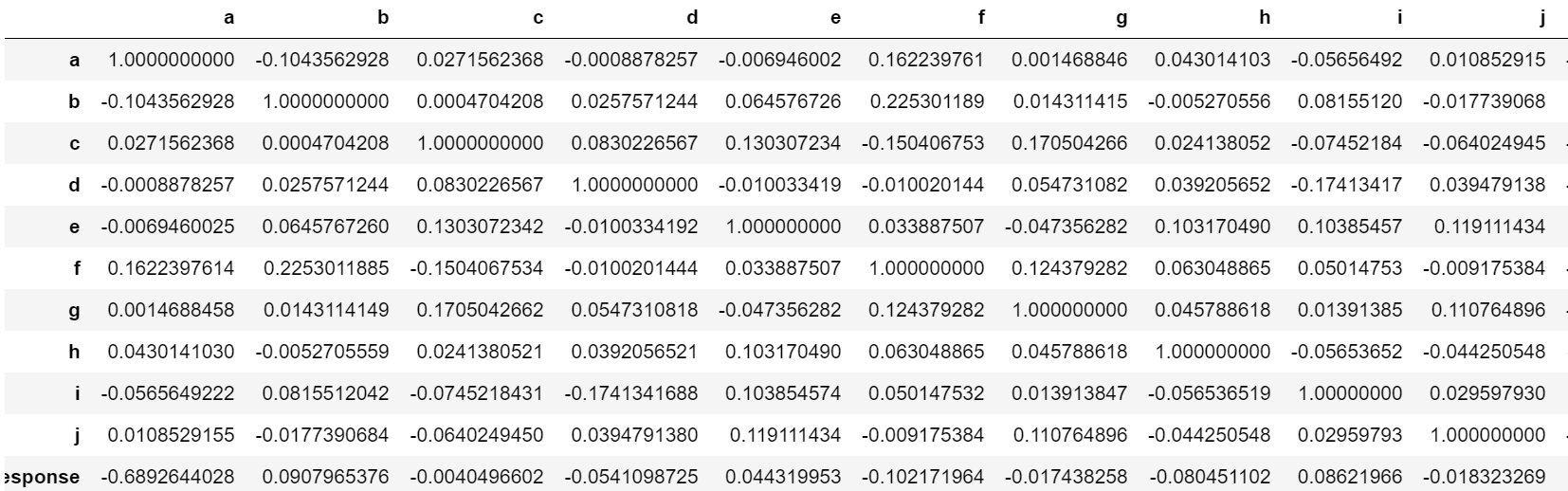
I then checked the SSE and MSE of the model manually by running predict on the training data and checking against the training response. To predict, I had to undo all my transformations that I previously did by subtracting the shift then taking the square of the predictions since the model uses the square root of the response. The SSE was appropriately low at 2.391 and the MSE at 0.024. Additionally, I plotted the model against the transformed variables and it seemed like a very good fit on the training data.

Now to check on the testing data, I had to call predict and then grab the real prediction values performing the same inverse transformations as discussed before. I then calculated SSE and MSE of the predictions by subtracting my predictions from the test response values, squaring those values, summing the values up, and then dividing by the number of observations. The SSE of our predictions is 1.436 and the MSE is 0.029. The MSE of the model (from the training data) and the MSE found from our predictions are very close which generally means the model generalized from our training data to our testing data fairly well.

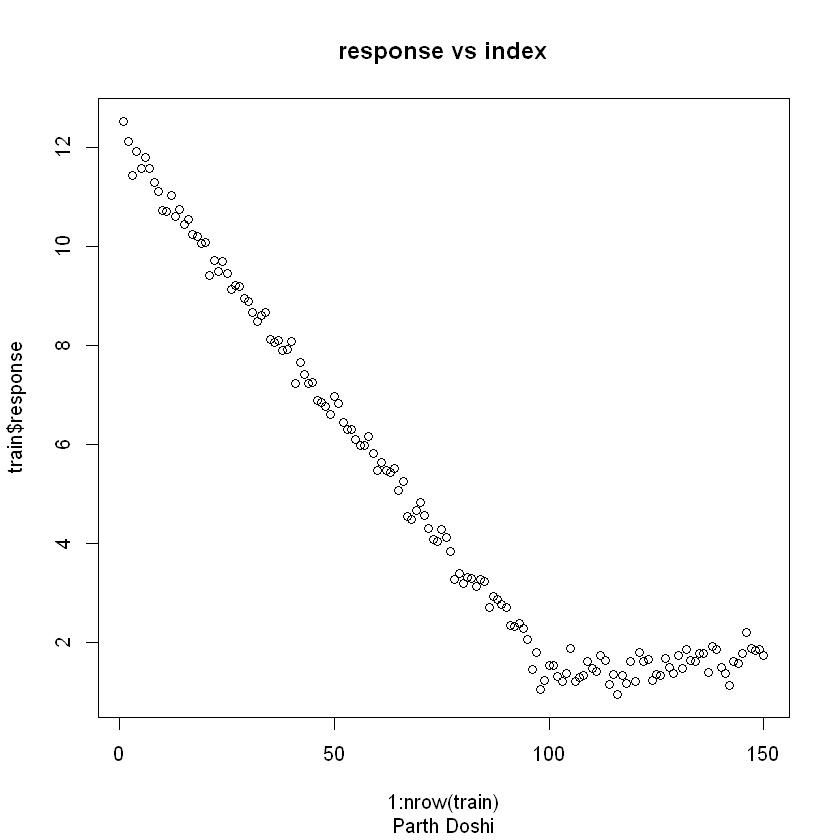
**Dataset2:**

I will name all features a, b, c, etc. in order of its column in the dataset and response is just response.

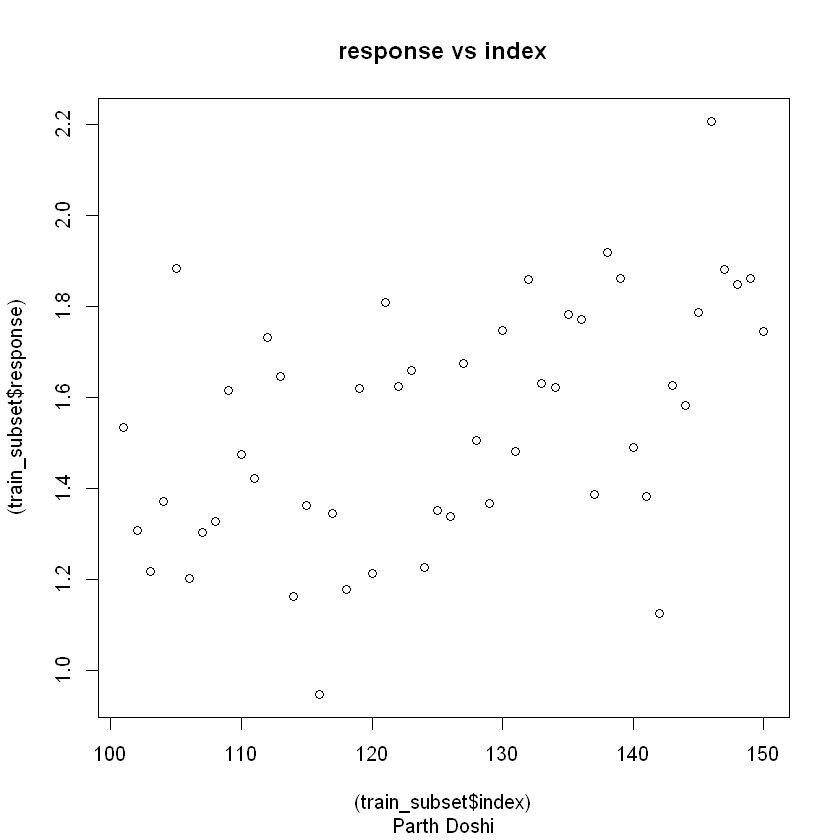
First thing I did was to plot all features against the response variable. This proved not very helpful since all the plots just looked like noise with no real correlation between any feature and the response variable. Even running correlation between all features and response basically told me the same thing:



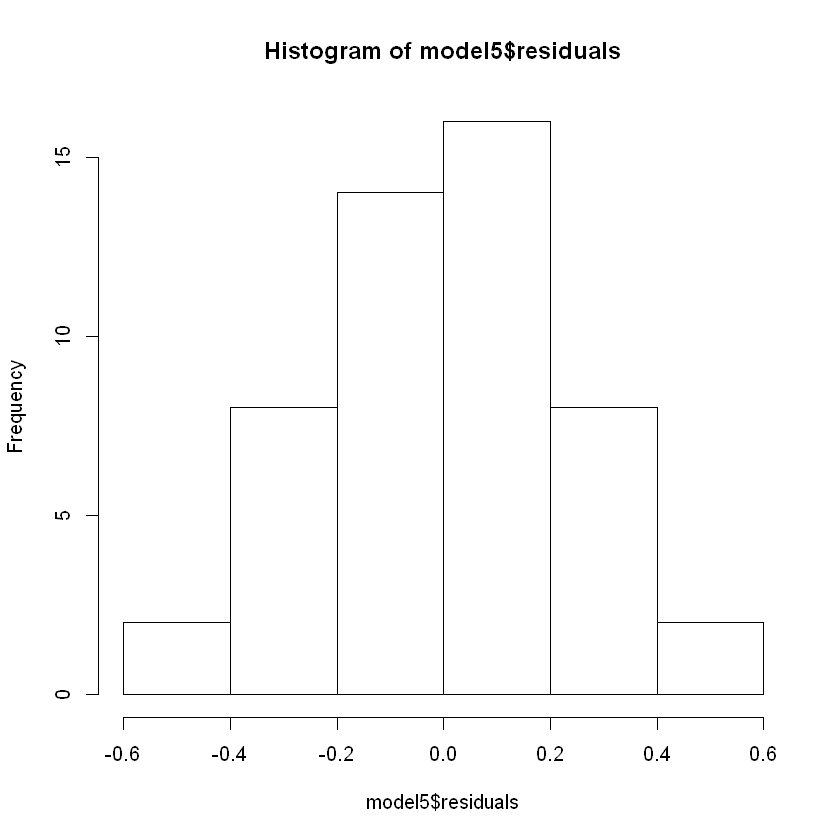
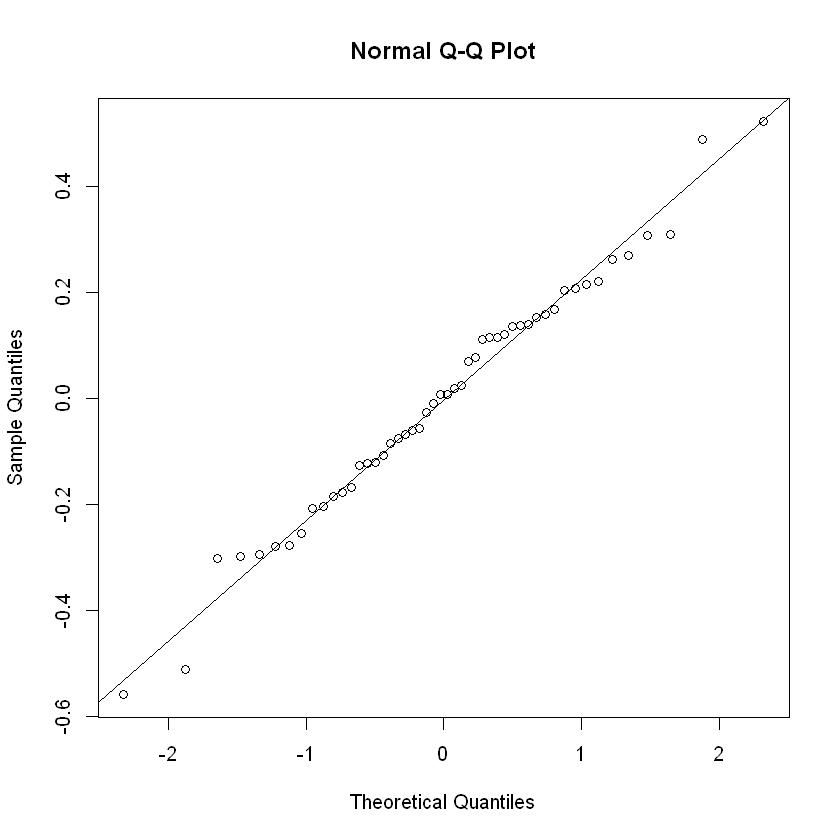
I then used the professor’s hint to use the index and then plotted the index of the observation against the response and found a mostly linear plot with a weird cutoff at around index 100:



This made me start plotting index against my other features to check if there was any correlation with that specific index. I immediately saw that if feature a was equal to 1, the response followed a whole different model. This prompted me to check the values of feature a in the test dataset and found feature a was always equal to 1. This meant that I could throw away all the observations where feature a was not equal to 1. This happened to be the first 100 observations. Additionally, I threw away feature a since this is now always 1 in the remaining training and test data. I then plotted the rest of the features (including the index) against the response values on the subset of the training data. These still looked like noise, but I could still see a slight correlation against the index:



I calculated correlation once more and found that my hunch was right, and that index was still highly correlated. Running an F-test to check if its coefficient was 0 or not came out statistically significant and so we reject the null hypothesis that index’s coefficient is 0. Thus, I now fitted a model using the formula . I then checked to see if the model’s residuals were normal and if the assumption of constant variance held:



They do and so I moved onto checking the model’s SSE/MSE against the predictions’ SSE/MSE.

Calculating SSE/MSE with this model is much simpler than the previous model since no transformations were needed. We plug the features’ observations in from the test data, subtract the prediction from the real value of y, square it, sum them all up, and then divide by the number of observations to find our SSE/MSE of the predictions. To get the MSE of the model, we do the same thing, but call predict on the training data. The SSE of the model is 2.5371514 and the MSE is 0.05285732. The SSE of the predictions is 2.10882314084606 and the MSE is 0.0421764628169211. The MSE of the model (from the training data) and the MSE found from our predictions are very close which generally means the model generalized from our training data to our testing data fairly well.