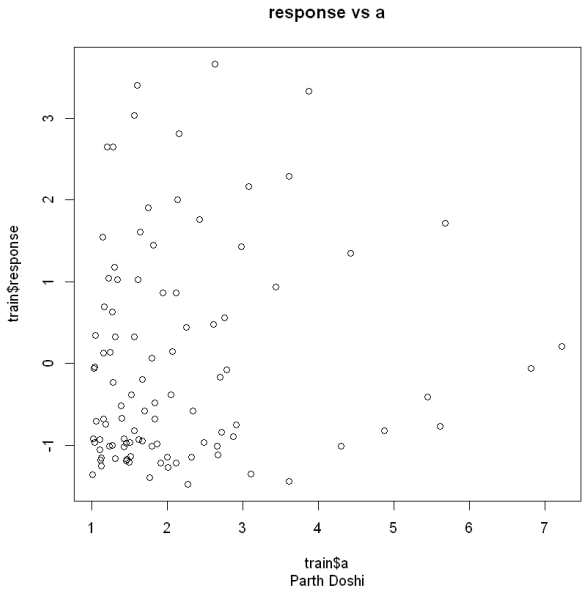
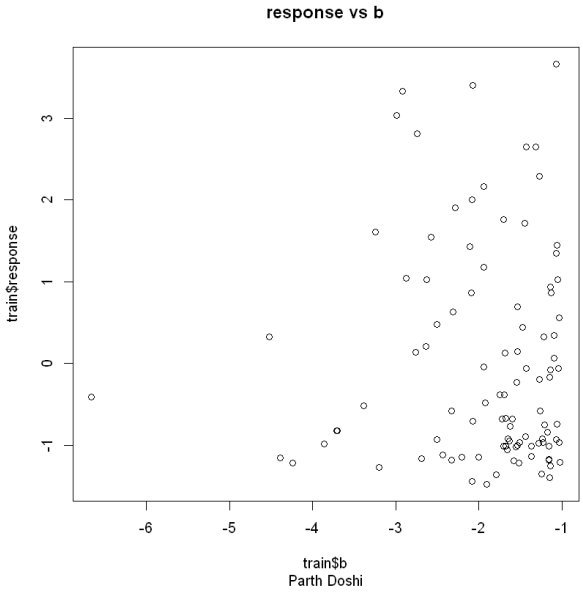
Parth Doshi, Max Lin, Abdulrahman Alabdulkareem, Austin Jin

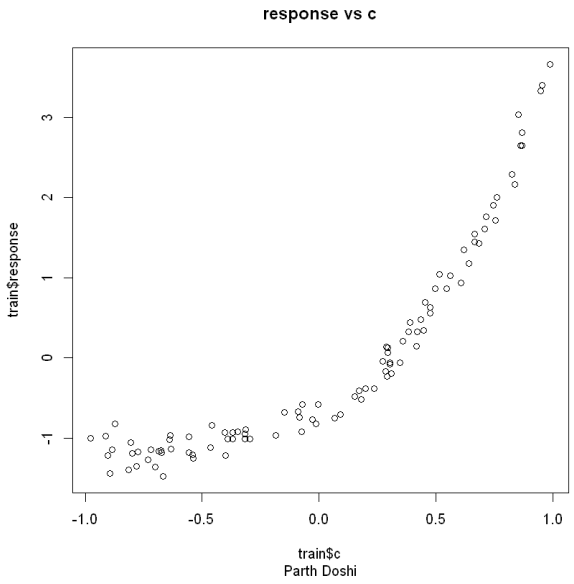
Dataset 1:

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

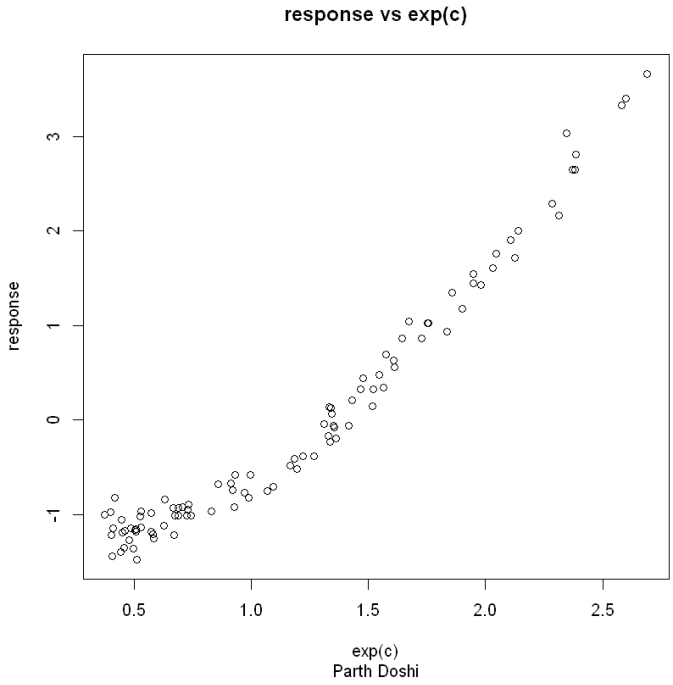
I first went ahead and plotted all the features against the response variable to find any obvious hints.







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:

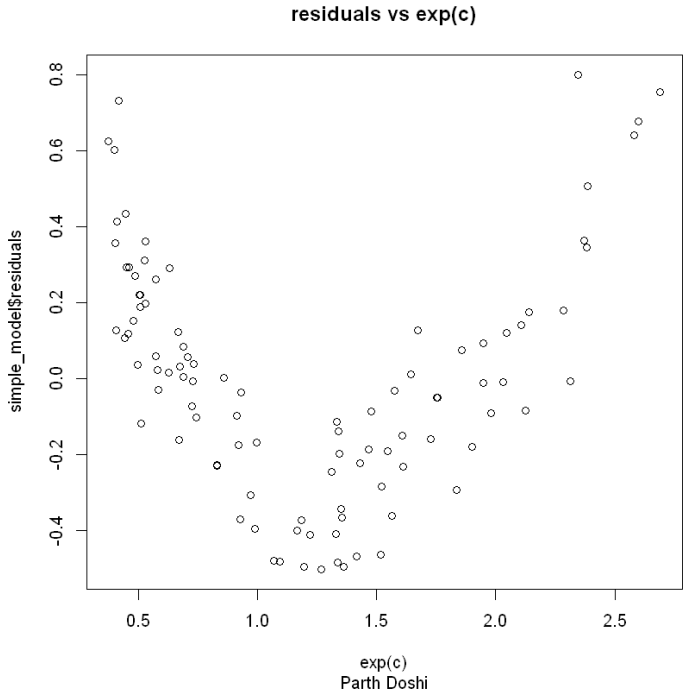


This looks much better.

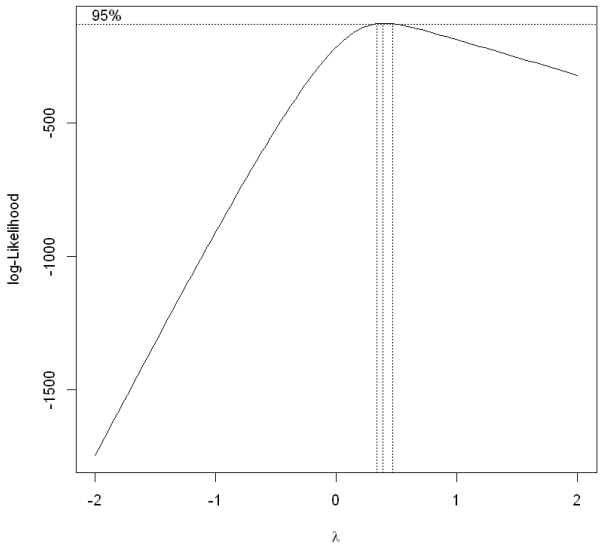
I now found the correlation between a, b, c, exp(c), and the response variable:

	a	b	c	response	shifted_response
a	1.00000000	-0.09974212	0.17618836	0.12212633	0.12212633
b	-0.09974212	1.00000000	-0.00901372	-0.02418928	-0.02418928
c	0.17618836	-0.00901372	1.00000000	0.89421214	0.89421214
response	0.12212633	-0.02418928	0.89421214	1.00000000	1.00000000
shifted_response	0.12212633	-0.02418928	0.89421214	1.00000000	1.00000000

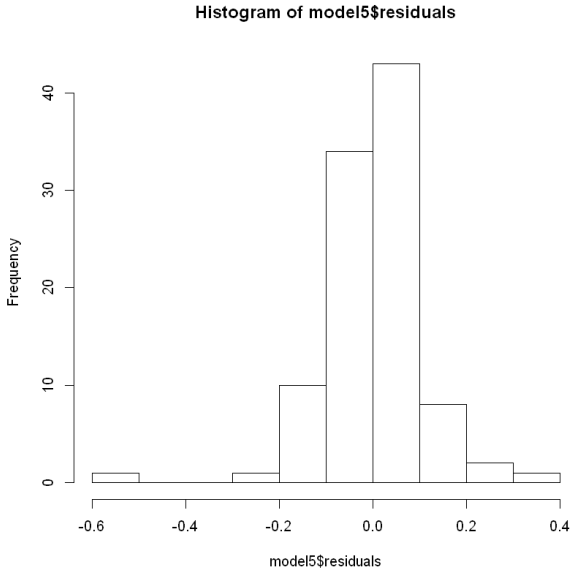
This helped to affirm my theory that c has high correlation with the response and the other variables are barely correlated. After running linear regression with the formula: $response \sim e^c$, I now need to check if my assumptions hold. After plotting a histogram of the residuals, they seemed mostly 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:

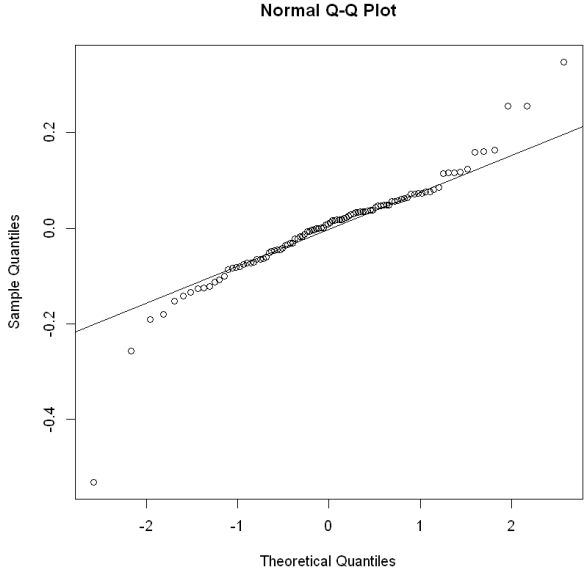


To fix this, I now need to transform our response variable. I wanted to run boxcox but found that the response variable had negative values; I then shifted the response over by the absolute value of the minimum response value and added a very small epsilon to have only positive values. I now ran boxcox 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: $\sqrt{response + shift} \sim e^{c}$. 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.

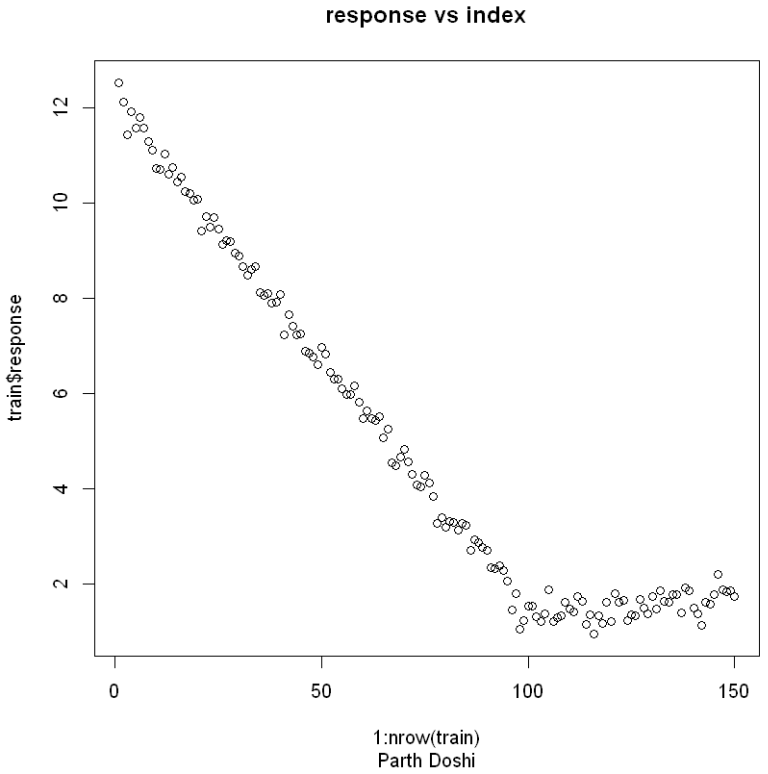
Dataset 2:

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:

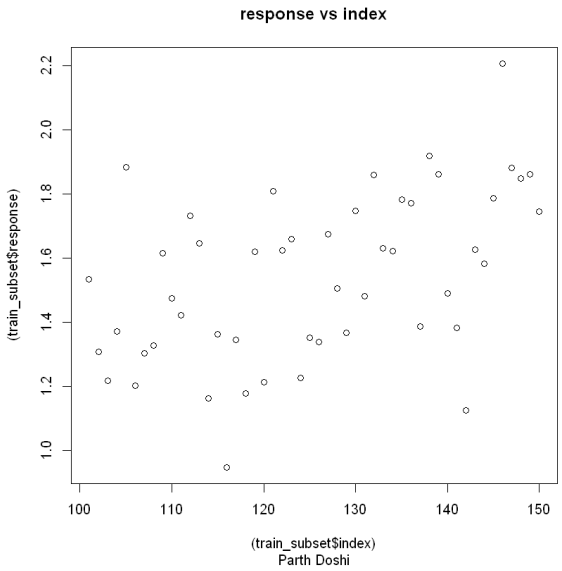
	a	b	c	d	e	f	g	h	i	j
a	1.0000000000	-0.1043562928	0.0271562368	-0.0008878257	-0.006946002	0.162239761	0.001468846	0.043014103	-0.05656492	0.010852915
b	-0.1043562928	1.0000000000	0.0004704208	0.0257571244	0.064576726	0.225301189	0.014311415	-0.005270556	0.08155120	-0.017739068
c	0.0271562368	0.0004704208	1.0000000000	0.0830226567	0.130307234	-0.150406753	0.170504266	0.024138052	-0.07452184	-0.064024945
d	-0.0008878257	0.0257571244	0.0830226567	1.0000000000	-0.010033419	-0.010020144	0.054731082	0.039205652	-0.17413417	0.039479138
e	-0.0069460025	0.0645767260	0.1303072342	-0.0100334192	1.000000000	0.033887507	-0.047356282	0.103170490	0.10385457	0.119111434
f	0.1622397614	0.2253011885	-0.1504067534	-0.0100201444	0.033887507	1.000000000	0.124379282	0.063048865	0.05014753	-0.009175384
g	0.0014688458	0.0143114149	0.1705042662	0.0547310818	-0.047356282	0.124379282	1.000000000	0.045788618	0.01391385	0.110764896
h	0.0430141030	-0.0052705559	0.0241380521	0.0392056521	0.103170490	0.063048865	0.045788618	1.000000000	-0.05653652	-0.044250548
i	-0.0565649222	0.0815512042	-0.0745218431	-0.1741341688	0.103854574	0.050147532	0.013913847	-0.056536519	1.00000000	0.029597930
j	0.0108529155	-0.0177390684	-0.0640249450	0.0394791380	0.119111434	-0.009175384	0.110764896	-0.044250548	0.02959793	1.000000000
esponse	-0.6892644028	0.0907965376	-0.0040496602	-0.0541098725	0.044319953	-0.102171964	-0.017438258	-0.080451102	0.08621966	-0.018323269

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:

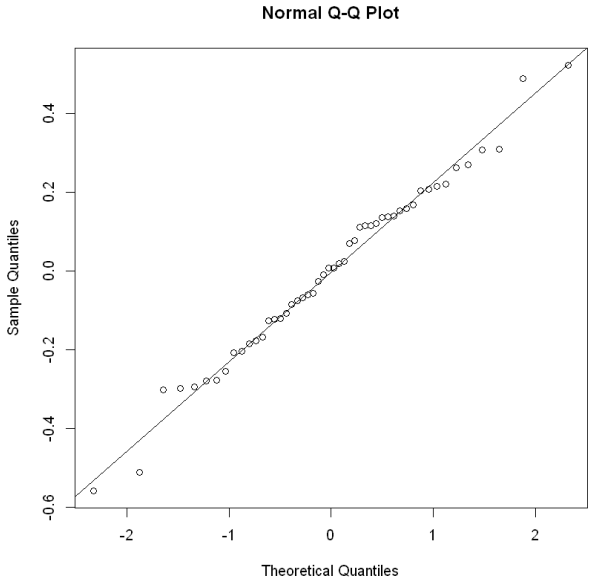


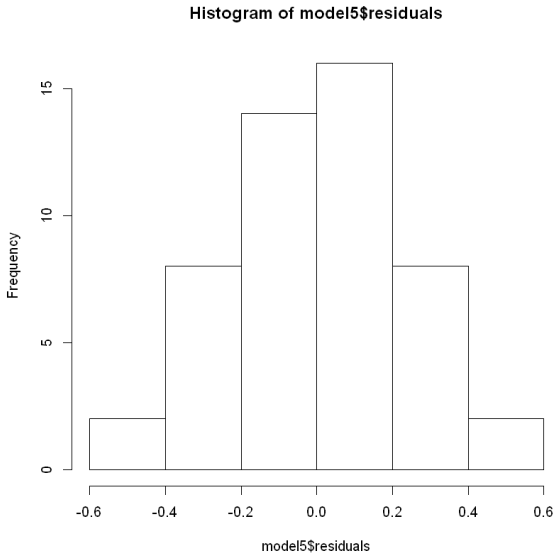
This made me start plotting index against my other features to check if there was any correlation with that specific index. I immediately saw that if feature a was equal to 1, the response followed a whole different model. This prompted me to check the values of feature a in the test dataset and found 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 $train\_subset\$response \sim train\_subset\$index$. I then checked to see if the model's residuals were normal and if the assumption of constant variance held:





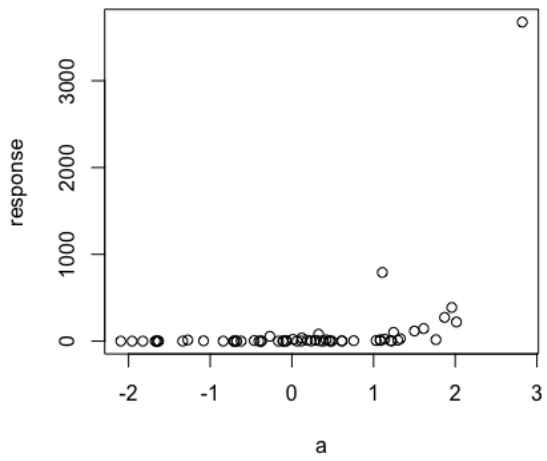
They do and so I moved onto checking the model's SSE/MSE against the predictions' SSE/MSE.

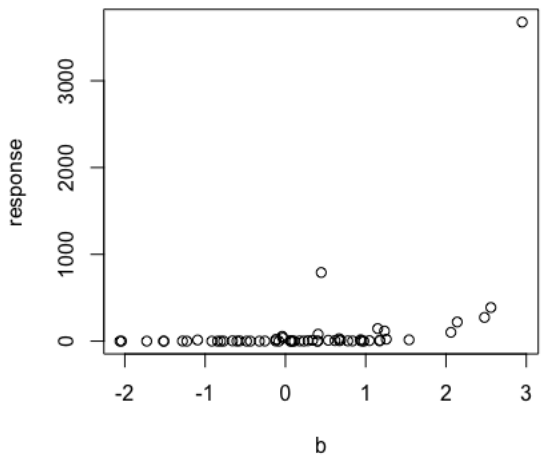
Calculating SSE/MSE with this model is much simpler than the previous model since no transformations were needed. We plug the features' observations in from the test data, subtract the prediction from the real value of y, square it, sum them all up, and then divide by the number of observations to find 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.

Dataset 3:

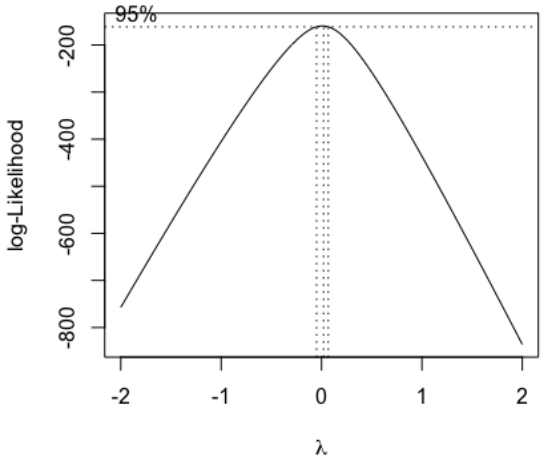
I will name all features a, b, c, d and response is just response.

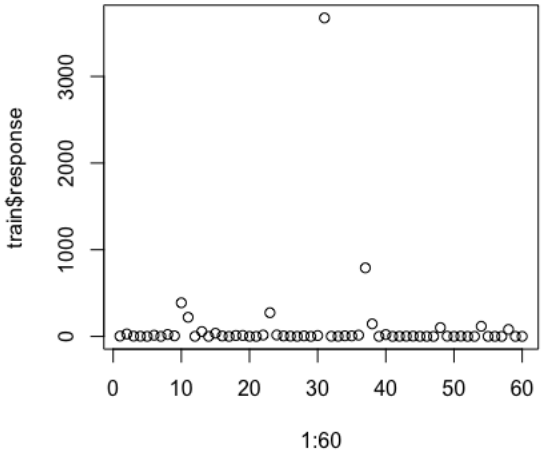
I first went ahead and plotted all the features against the response variable to find any obvious hints. We see that all a,b,c, and d are very hard to interpret as they all fall within a small range with some big 'outlier' on the graph.





The graph of response vs A seemed to show that some type of exponential function. I ran boxcox which indicated that a $\lambda = 0$, would make a good choice for the transformation on response.

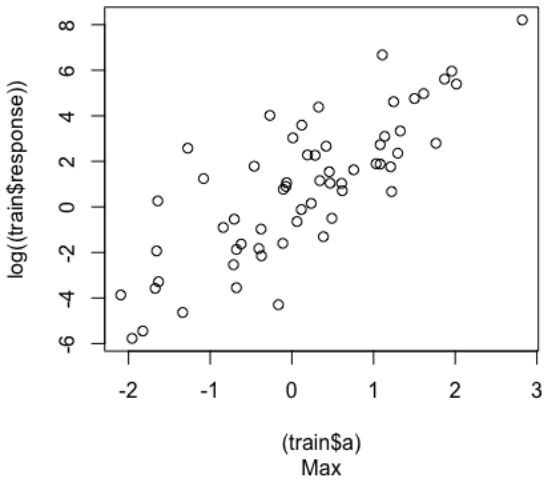


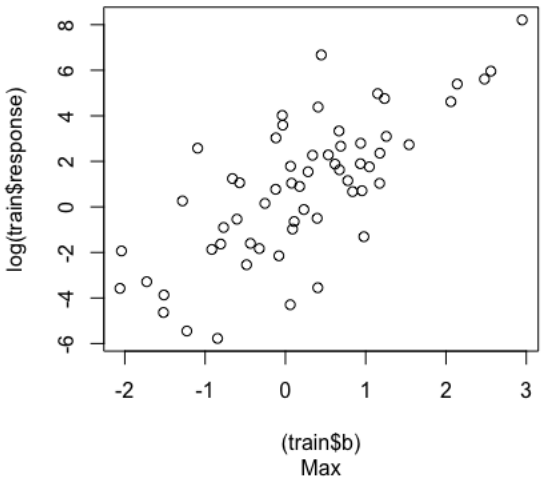


I transformed the response into log(response) and repeated the steps of plotting a,b,c,d vs log(response). This time, the data became much more interpretable. We could see some clear linear relationships on both A and B. C and D however, looked more like random scatter and noise, so we believed that to be unhelpful in explaining the data.

I also made sure to check the index vs response plot to check any anomalies in the data. It seems like

that point near 3000 and near 1000 may be outliers.

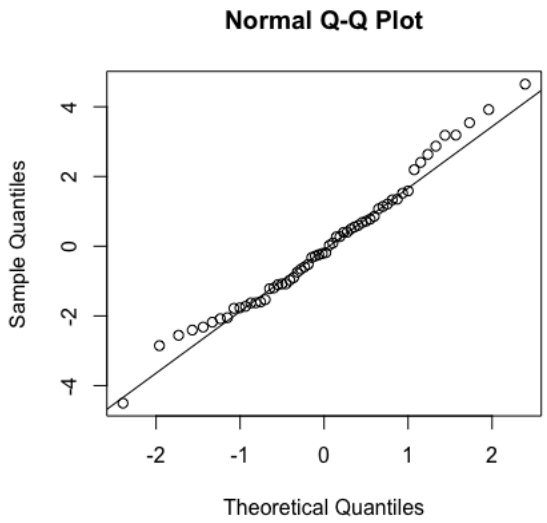




I ran a simple linear model including a,b,c,d. The anova showed that A was statistically significant whereas B,C, and D were not. Upon further inspection, I calculated the pairwise correlation between the variables and noticed that A and B were very correlated.

```
> cor(train)
                    a           b           c           d  response log_response
a           1.00000000  0.92898436 0.31048529 -0.03658013 0.4009296   0.80080462
b           0.92898436  1.00000000 0.05269316 -0.10999749 0.4049804   0.74103923
c           0.31048529  0.05269316 1.00000000  0.09901290 0.1816143   0.29829042
d          -0.03658013 -0.10999749 0.09901290  1.00000000 0.1946926  -0.06096714
response    0.40092960  0.40498043 0.18161434  0.19469258 1.0000000   0.43082568
log_response 0.80080462 0.74103923 0.29829042 -0.06096714 0.4308257   1.00000000
```

I checked the VIF and found a value of ~7 which is not above 10, but is moderate to high levels of multicollinearity so I removed B from the data. I then checked my assumptions on the residuals of the data through the histogram and qqplot. The qqplot followed the qqline and the histogram seemed pretty normal, with a very slight right skew indicating an overall normal distribution of the residuals.



I went further to plot residuals vs a,b,c,d, all of which seemed to show constant variance. In order to estimate SSE, we predict the test data, then we exponentiate to transform back to the original response variable scale. We then calculate SSE with the (predicted – target)^2. We estimate SSE = 304667.6 on the test dataset.

Dataset 4:

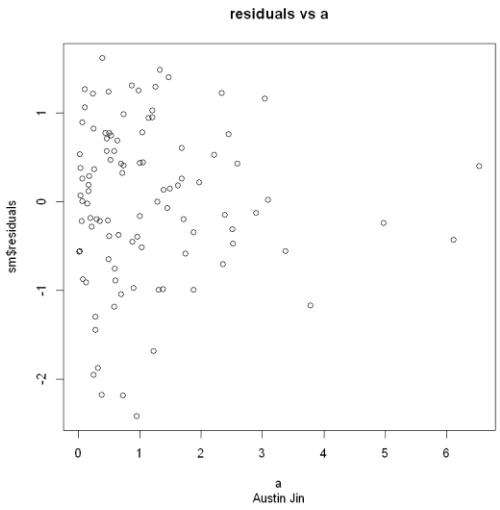
First thing to notice about this dataset is that it contains only one explanatory variable, I tried to run a simple linear regression model on the single variable and I got an SSE of 7.6. So then I followed it up and tried to use the indices as another variable, so first I plotted the indices against the response variable but it was complete noise, I still tried to print a correlation table and saw that the correlation between the response variable and the explanatory variable is 0.66 while the correlation between the response variable and the index is 0.03 so it's safe to assume that the index is not useful, just in case I ran another simple model with both the explanatory variable and the index and got an SSE very similar to the original model and the p-value of the index is 0.67 so its safe to assume that we can throw out the index.

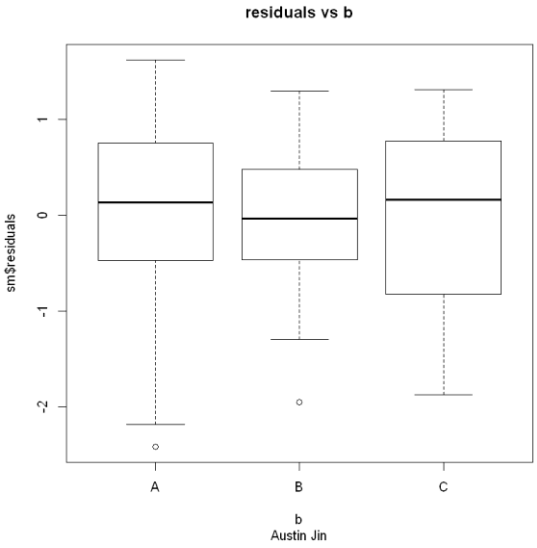
Next I did the obvious thing to do which is to look at the plot of the response variable against the explanatory variable, and the plot looks very similar to the log function or the sqrt function. So I tried training two models one against log(x) and one against sqrt(x) and got SSE's of 7.5 and 5.9. after looking at the plot of log(x) against the response variable we see that there is a clear outlier in the x-axis because in the original x data, one of the points is very near 0 thus log(x) makes it an outlier, an easy fix is to do log(x+1) which gave me an SSE of 5.47. So we can figure out that log is a bit better than sqrt althout the SSE of log is so high due to an outlier. So I performed a boxplot on x and saw that the outliers are c(1, 10, 56) thus I removed those points and retrained to get an SSE of 5.43.

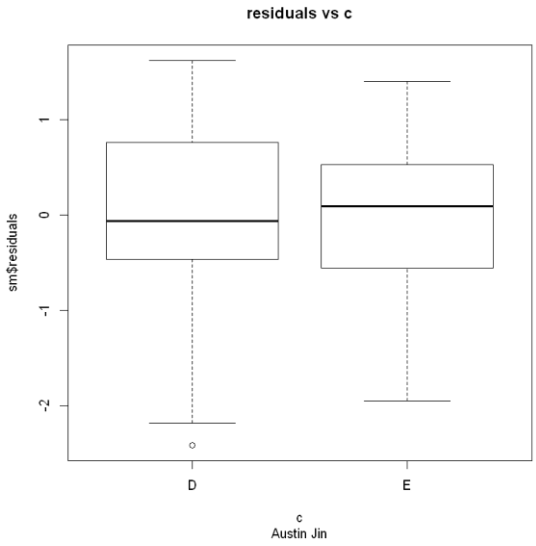
Finally I looked at the three models I currently have which are sqrt(x), log(x), and log(x+1) and calculated the SSE for the test set and saw that I get SSE's of 4, 9.3, 6.18 which might make it seem like sqrt is the best representation however when we print the plot of the residuals we can see a single point that is a large outlier, and we can see that log(x) fits the data much better than sqrt(x). the different between log(x) and log(x+1) is very small that we just chose to go with the simpler model and choose log(x) for an SSE of 5.4346 on the training set without including the points c(1, 10, 56) and an SSE of 9.309673 on the test set however we get an SSE of 2.90391 on the test set without the obvious outlier.

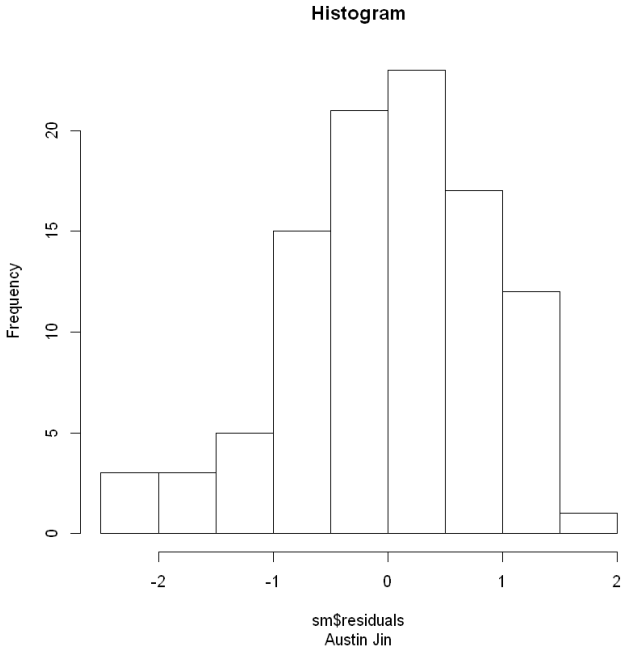
Dataset 5:

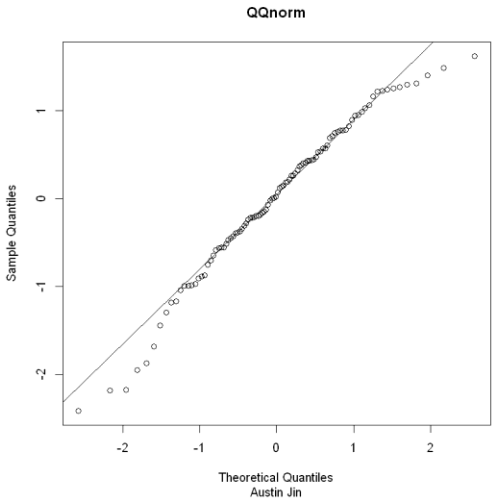
Relevant Plots:







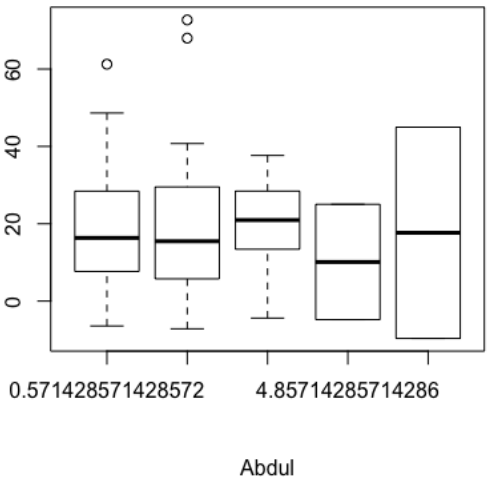


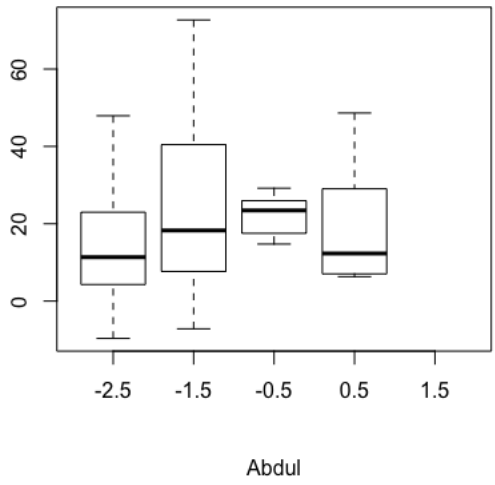


I took in the data and saw 3 features a, b, and c. I saw that b and c were factors. I plotted the response vs the index and saw a few different models. After tukey I saw that there was a relation between the 2 factors b and c as well as a relation with a. I created a linear model $response \sim a * b * c$. I checked the residuals as seen above and saw that it was normal. I checked the variances and saw it was mostly constant variance. My MSE and SSE were easy to predict since there were no real transformations done to the data here.

Dataset 6:

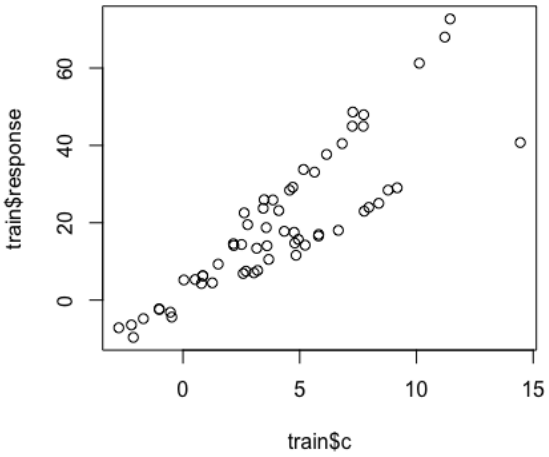
We use the data as 4 explanatory variables a,b,c,d and the response.
First, we alter the data at index 28 and set at 0.5 because we never see that factor, so it causes problems when converting to factors.
We plot a,b,c,d vs response to gain insight on the data.

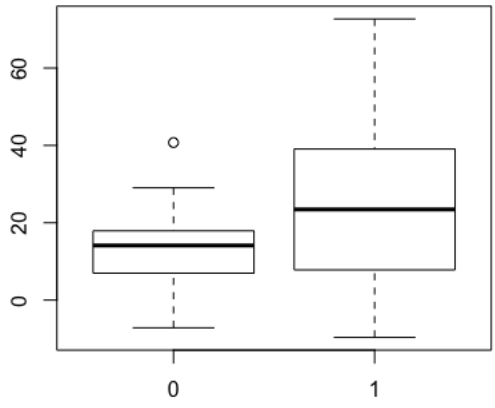




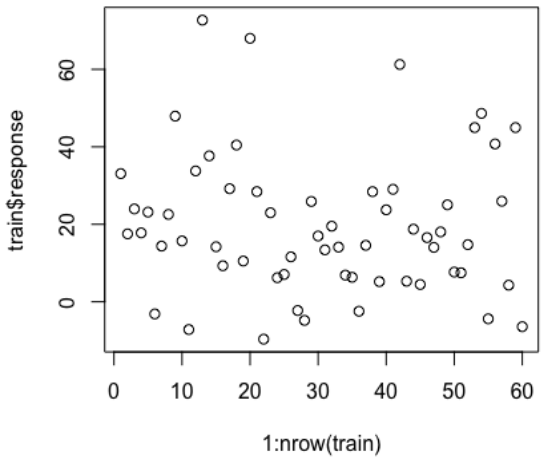
On left: response vs. A

On Right: response vs B



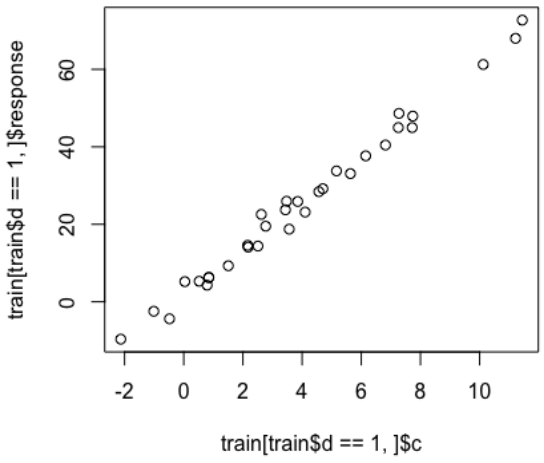


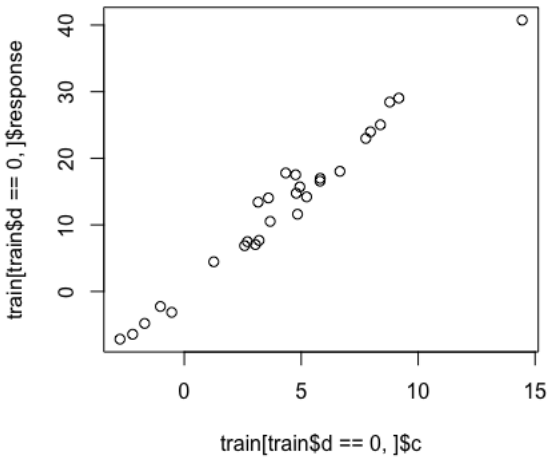
We plot C and see that it is very promising. Has a good a linear shape. Seems to have 2 lines.
The plot of D with 2 factors might explain the 2 lines of C.



The index plot seems to be randomly scattered, which is not very helpful.

Training 2 different models, for d = 0 and d = 1 gives us great lines.

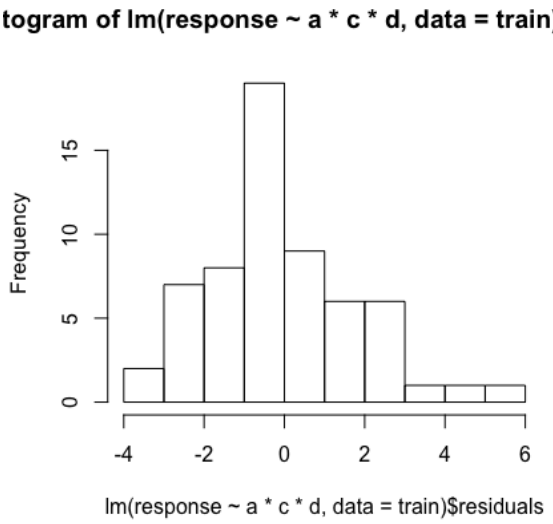


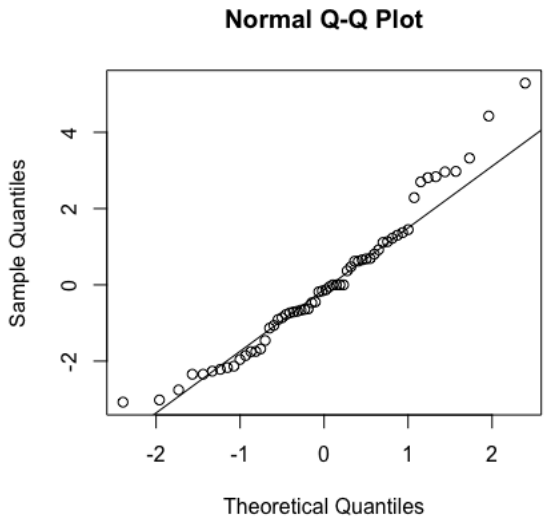


With this improvement we decrease our SSE from 2000 to 250 with only using c and d and its interaction. Using all combinations of a*b*c*d with all factors can drop our SSE to 120. Using a tukey test, we see that none of the levels of A and B are significantly different. Only D differs using contrast.

We end up using A*C*D as our model.

We check our assumptions for normality of residuals:





We arrive at an SSE of 129.9422 for the test data.