Dr. Azencott- Spring 2016  
Machine Learning – Project 3

**Linear and Non-Linear Regression**

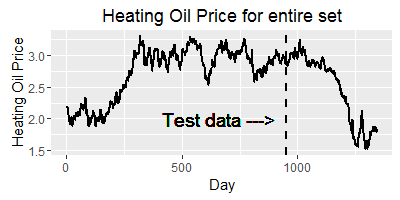
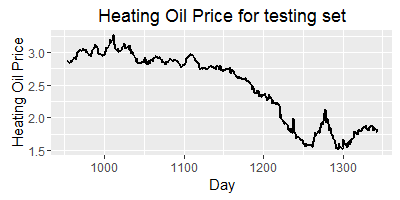
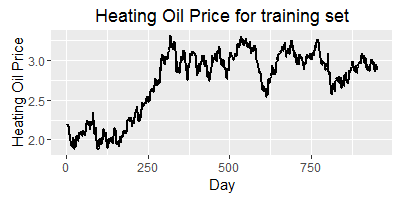
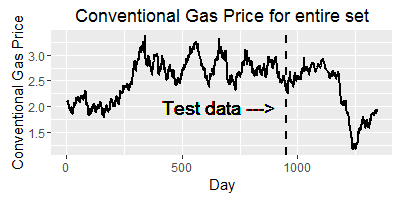
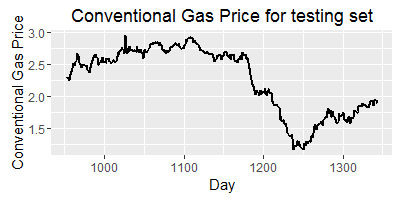
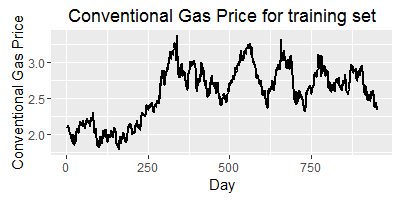
It is well-known that predicting the future price of commodities is a sought after endeavor. I employ three techniques throughout this paper to predict the future price of Conventional Gasoline, Heating Oil and the next-day price of Aluminum. First, standard linear regression is used to predict both the price of Conventional Gasoline and Heating Oil. The explanatory variables for both of these regressions are: Crude Oil Prices in West Texas (B), Henry Hub Natural Gas Spot Price (G), Propane Prices in Mont Belvieu, Texas (F) and the Crude Oil Price in Brent Europe (H). Next I use a more sophisticated regression technique to predict the next-day price of Aluminum. The explanatory variables are: steel price, nickel price, copper price and aluminum price. I employ non-linear regression with polynomial kernel degree 2 for this. I do not attempt to optimize my offset or scale of this function. However, I do recruit kernel ridge regression as a mathematically elegant way to improve the invertibility of our kernel matrix and to improve the accuracy of the predictions. In all of these regressions I split my data into two sets. I create a training set and testing set, both of which I keep in sequential order. Randomizing points would be useless in predicting future prices. I outline the details in the subsections below. I use R as my programming language. Some of the algorithms were written from scratch yet some functions were called.

**Part 1: Standard Linear Regression:**

The parent data set in which I extracted my variables had many other day prices of commodities. There are 1411 days recorded starting from 01/04/2010 through 06/01/2015. First, I import the data set and immediately change the names for easier manipulation. I set the four explanatory variables and two response variables (separate regressions) to have the following names:

* B = Crude Oil Prices: West Texas
* G = Henry Hub Natural Gas Spot Price
* F = Propane Prices: Mont Belvieu, Texas
* H = Crude Oil Prices: Brent Europe
* E = Conventional Gasoline Prices (Response)
* I = Heating Oil Prices (Response)

I then split my data into training and testing. I allocate the first 1000 days to training and the remaining 411 days to testing. I recognize that there are missing values. I opt to remove these points before I perform regression. This leaves me with 952 for training and 391 for testing. Below I plot the response variables over the training, testing and full data set.

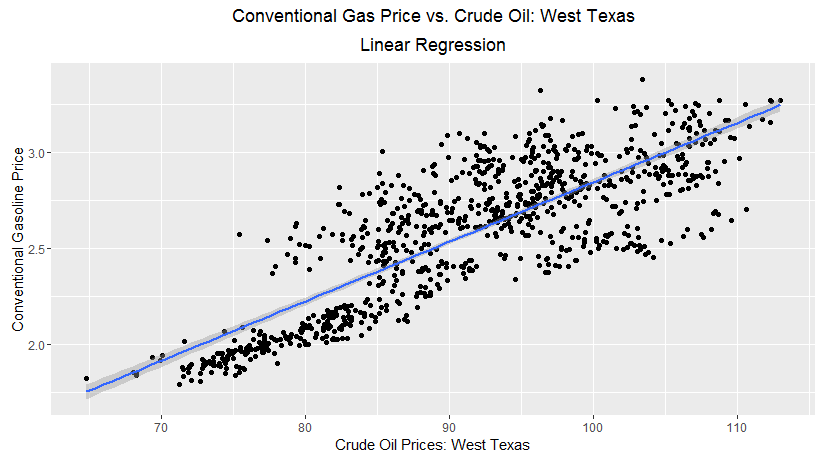


After studying the above plots, we can hope that our model can account for the sharp decline in price around day 1150-1175.

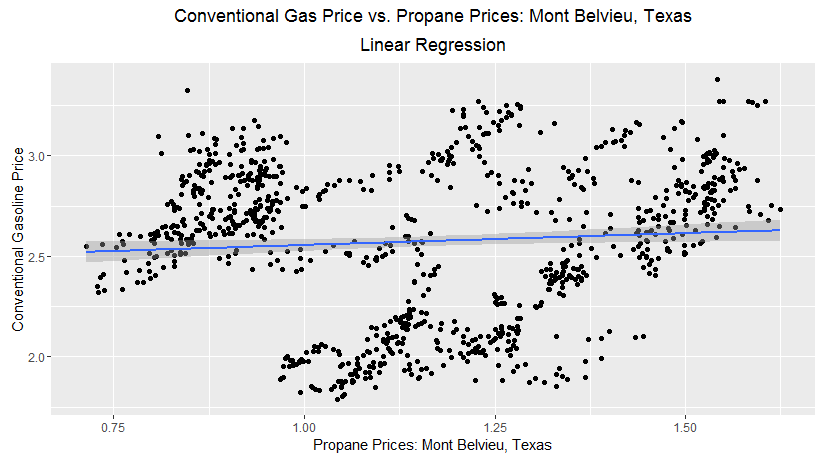
**Plotting**

I have chosen to plot each explanatory variable against the response variable to better understand the relationship between the two. I use stat\_smooth(method = ‘lm’) to show the best fit line between that particular explanatory variable and the response.

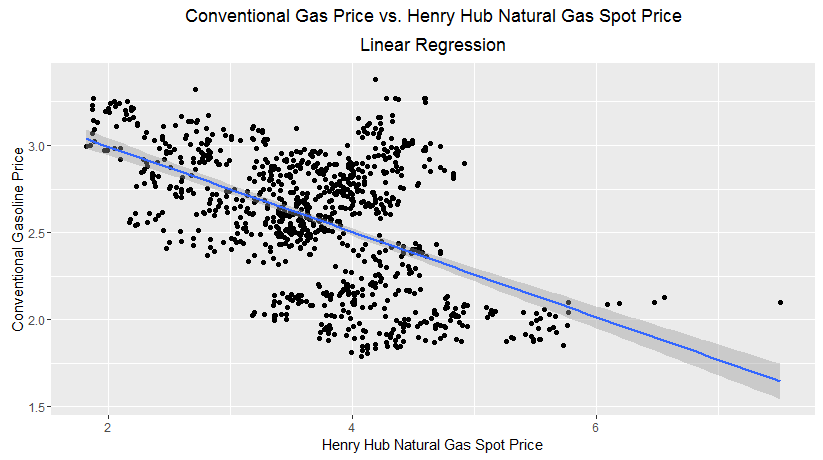
**Conventional Gas Price**



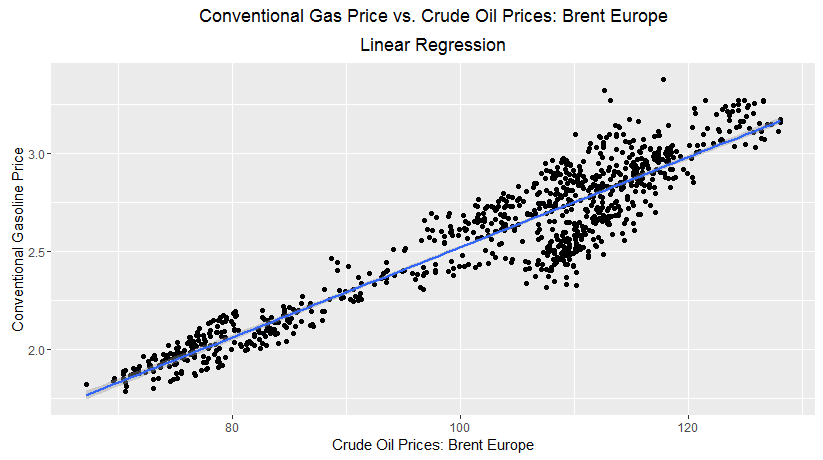
We can see that there is a noticeable positive relationship.



There is a very heterogeneous mixture of values here. Linear regression clearly is not capturing this relationship.

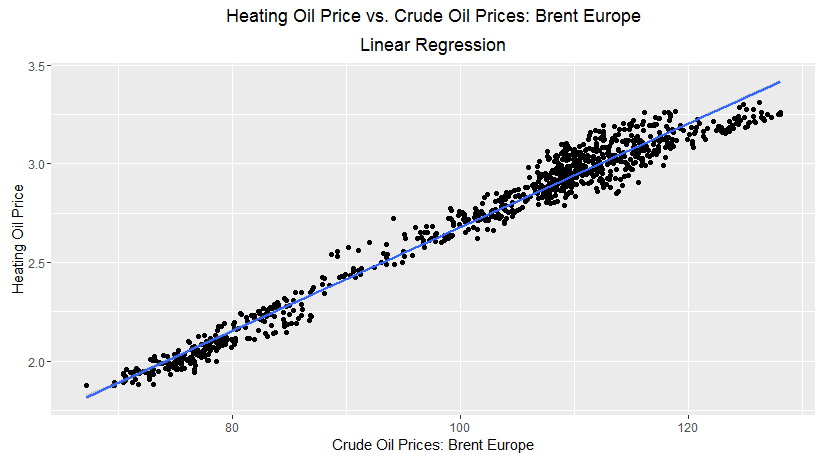
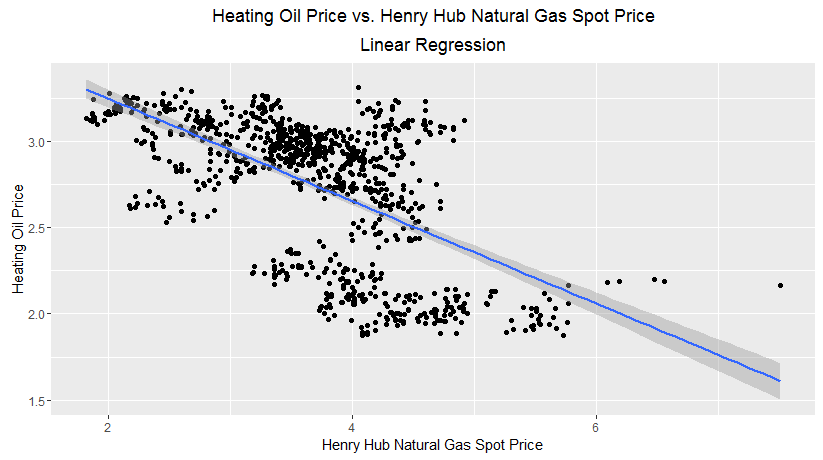
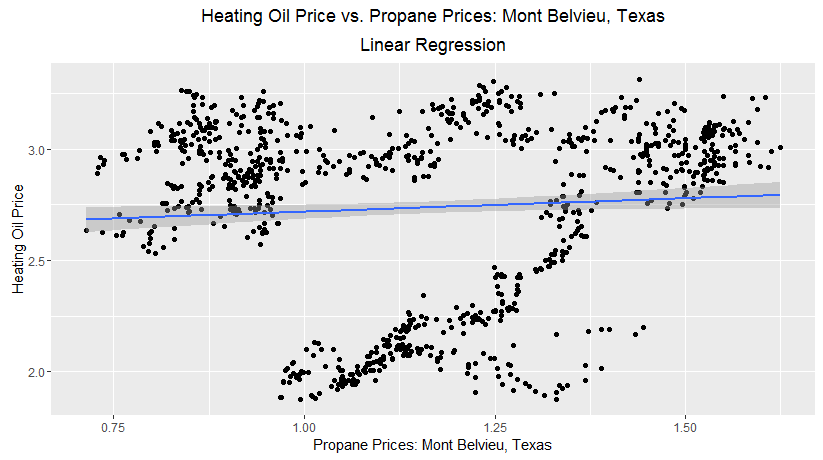
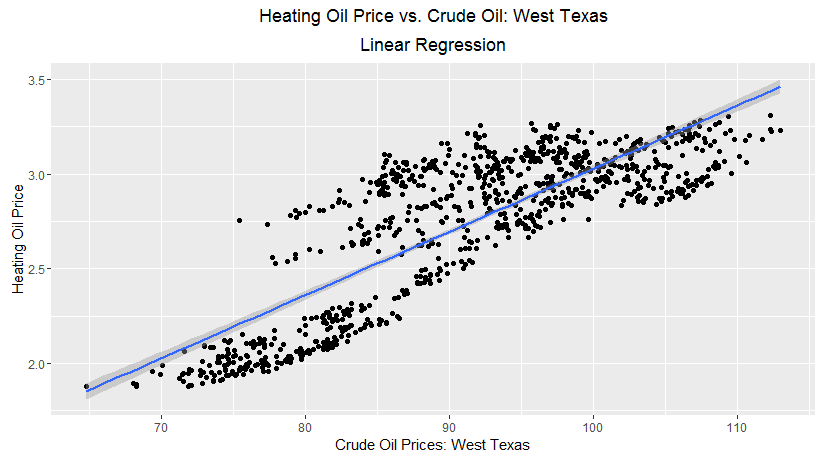


The best fit line has resulted in a downward trend but the relationship is not too clear given the distribution of points.



This response variable has the tightest pattern of the four. We know Crude Oil prices and gasoline to in fact have a linear relationship.

**Heating Oil Price**



The plots for Heating Oil Price vs. the explanatory variables have similar patterns to the simple regressions of Conventional Gas Price.

**Simple Linear Regression**

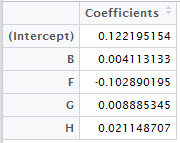
Simple linear regression has a very simple function call, ‘lm’, that I used for predicting both Gas Price (E) and Oil Price (I). ‘lm’ solves the following equation for the regression coefficients :



Where , explanatory variables, is the response variable.

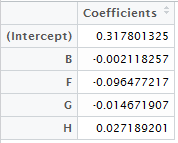
The regression coefficients are as follows:

**Regression Coefficients for Conventional Gas Prices**



We can see that the two biggest contributors to price are F and H. F refers to propane prices. These are competing energy sources so it makes sense that F is negative. H refers to Brent crude oil. We see that as Brent crude (the main factor of gasoline) increases so does the price.

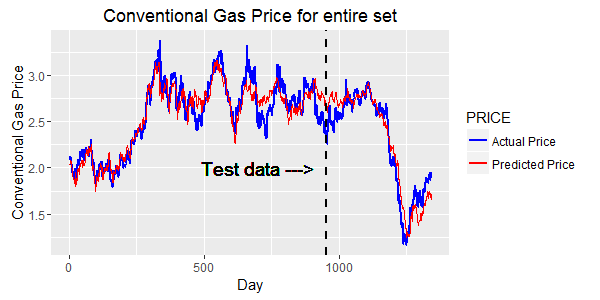
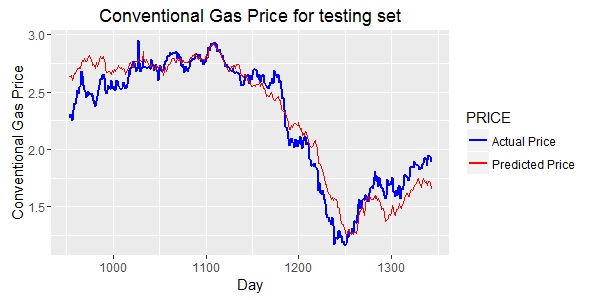
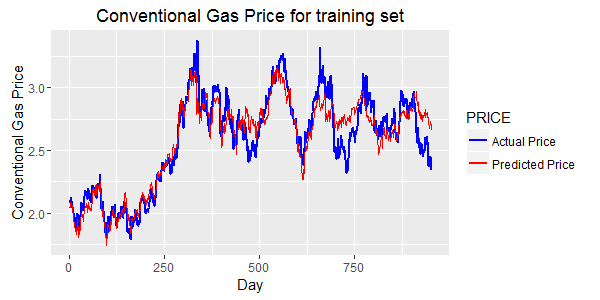
**Regression Coefficients for Heating Oil Prices**



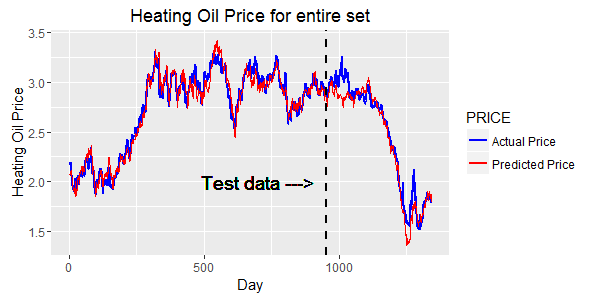
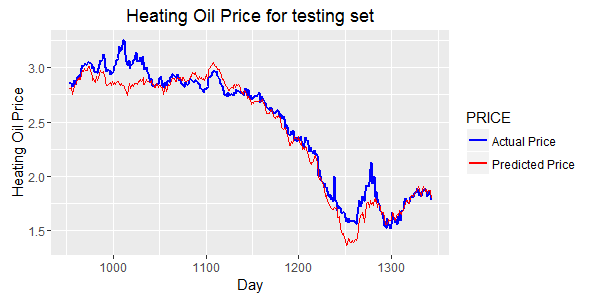
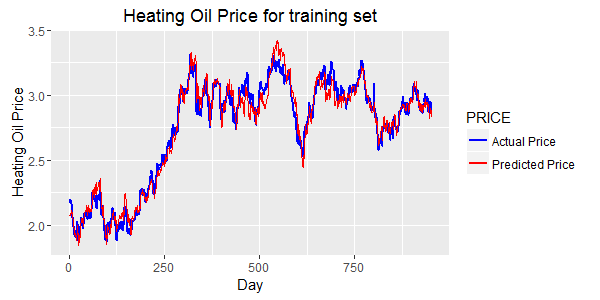
The above table illustrates that all but H have a negative coefficient and F has the largest impact on the response. This makes sense as F represents propane prices. As the price of propane increases the price of heating oil goes down. It is interesting that B does not have a positive coefficient. However, it does contribute the least.

**Predicted Plots**

**Conventional Gasoline Prices with Predictions**

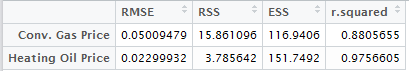


**Heating Oil Prices with Predictions**

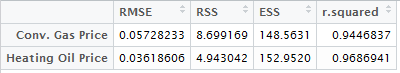


After studying the above plots we can see that the predicted curve follows most closely to the actual price for the training set. This is obviously the case as the learning algorithm learned from these values. The predicted values for the test set follow reasonably well. The following table will help us better understand just how well it predicted.

**Performance Training Set**



**Performance Testing Set**



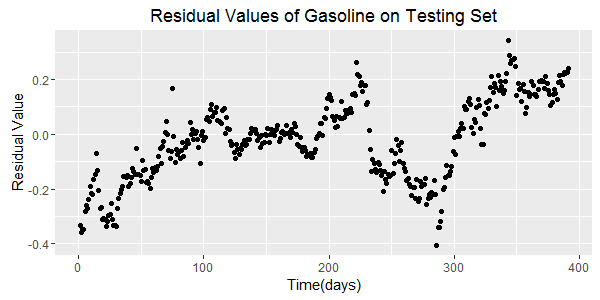
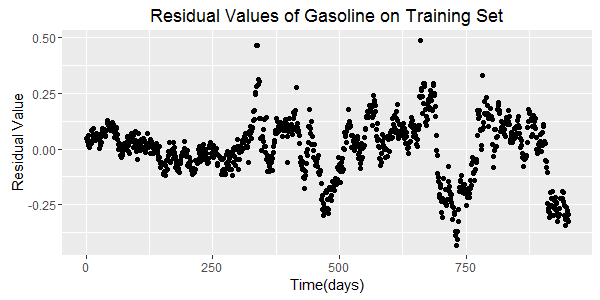
The abbreviations in the above table are defined as such:

* **RMSE: Root Mean Square Error**
  + RMSE represents the sample standard deviation of the differences between the predicted values and the observed values. In other words, RMSE is the measure of spread of data around the regression line. The lower the better.
* **RSS: Residual Sum of Squares**
  + RSS measures how much variation there is in the residuals of the modelled data.
* **ESS: Explained Sum of Squares**
  + ESS measures how much variation there is in the modelled values.
* **r.squared: or the coefficient of determination**
  + indicates the proportion of the variance in the dependent variable that is predictable from the independent variable.

We can see that is high ~95% in both predicting Gasoline price and Heating Oil price. This means that 95% of variation in Gas and Oil price is accounted for by the chosen explanatory variables. The RMSE error is proportionally low which tells us that the residual errors are very small. Finally, I share and explain the residual plots.

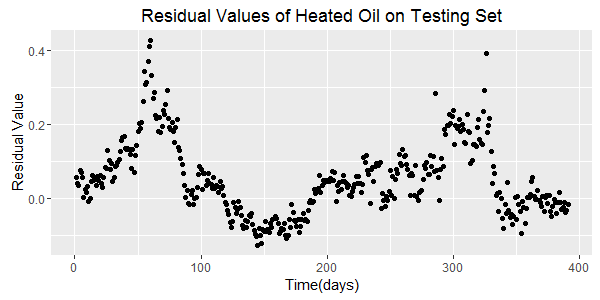
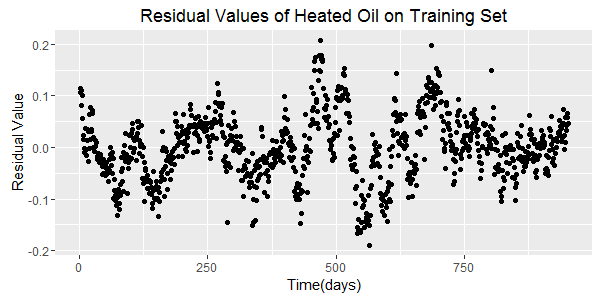
**Residual Plots**

**Conventional Gasoline Residuals**



Both of these residual plots suggest that a linear model is not suitable for predicting because we see a definite pattern. The residual plots should not indicate any bias by systematically over or under predicting the values.

**Heating Oil Residuals**



As with Conventional Gasoline Prices, we see that Heating Oil Prices have similar issues with their residual plots, namely that there is an obvious, discernable pattern.

**Conclusion**

Based on studying the regression plots and comparing the performance statistics I believe that simple linear regression did a fine job predicting the price of both Conventional Gasoline Prices and Heating Oil Prices. That being said, I believe that a nonlinear regression technique would yield higher accuracy.

**Part 2 and Part 3**

**Nonlinear Regression with a degree 2 Polynomial Kernel:**

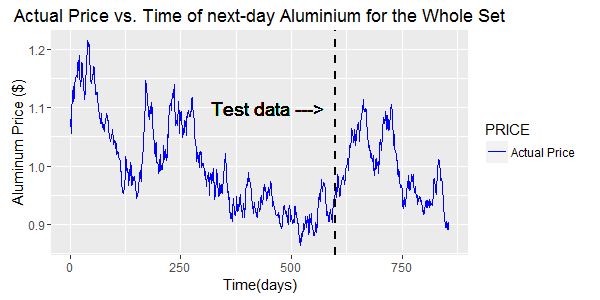
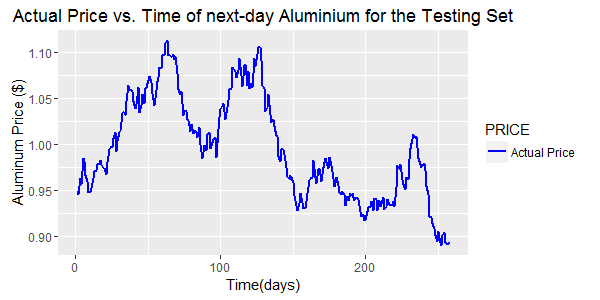
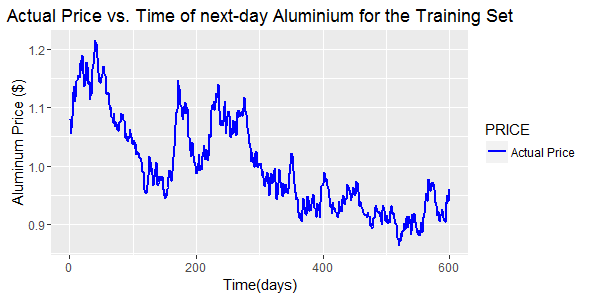
**with and without Kernel Ridge Regression**

The parent data set in which I extracted my variables had many other day prices of other commodities. There are 838 days recorded starting from 01/03/2012 through 06/08/2015. First, I import the data set and immediately change the names for easier manipulation. I set the four explanatory variables and response variable to have the following names:

* D = Steel Price
* E = Nickel Price
* I = Copper Price
* J = Aluminum Price
* JJ = Next-day Aluminum Price (Response)
* Predict = Predicted price without KRR
* Predict.krr = Predicted price with KRR

I then split my data into training and testing. I allocate the first 600 days to training and the remaining 258 days to testing. I recognize that there are no missing values. I calculate the mean of each response variable in the training set and save them as mD, mE, mI, and mJ. I use these values to scale my explanatory variables for all corresponding values in training and testing. Next I create a new column named ‘JJ’ in both training and testing. I fill this column with the next-day price of aluminum. I choose to define the last day (600th in training and 258th in testing) in each set as the one before it. I could have used a value more fitting of the general trend of the price but felt that repeating the 600th and 258th day would not affect my regression coefficients too much. Below I plot the response variable against time for the training period, testing period and then the full period.

**Next-day price of Aluminum**



**Non-Linear Kernel Regression**

Nonlinear kernel regression is very similar to the above formulation. The matrix of explanatory variables, and need to be replaced by the kernel function, , so that solving for the regression coefficients is equivalent to solving the following:

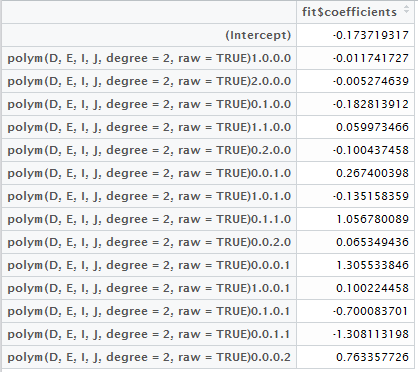
, where is the kernel matrix.

Non-linear regression with a user-specified choice of kernel is the function call, ‘lm’ and defining the formula appropriately. The below code extracts the regression coefficients:

*fit <- lm(data = training, JJ ~ polym(D, E, I, J, degree=2, raw=TRUE))*

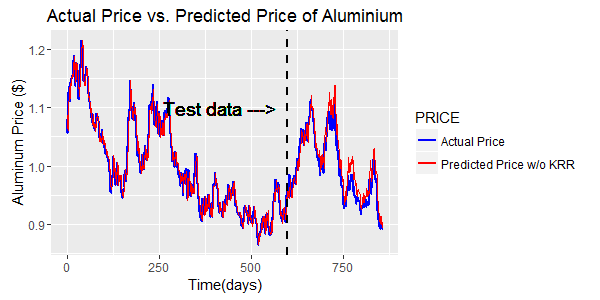
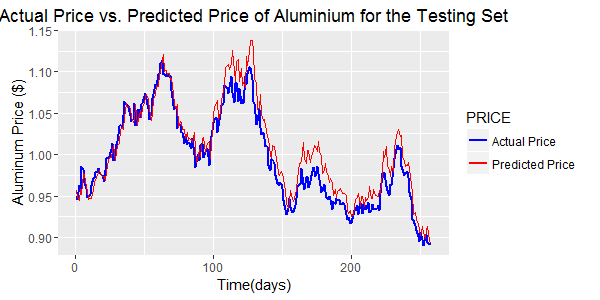
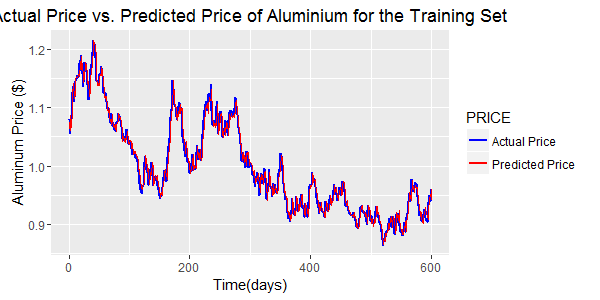
*View(as.data.frame(fit$coefficients))*:

**Regression Coefficients**



It is difficult looking at the above table and making claims that are evidently true. But we can see that Aluminum price and Copper price have the biggest impact on next-day Aluminum Price. Conversely, it seems that Steel price and Nickel price have less of an impact, particularly Steel price. To see how these values result in predictions, let’s inspect the plots below.

**Actual vs. Predicted Plotting**



The above plots show that the training period was more accurately predicted. This is expected since our algorithm trained on this set. We can see that the testing set predictions follow nicely until about the 100th day. Before I include the residual plots or performance tables, I would like to run the same regression but introduce a ridge. The technique is known as kernel ridge regression.

**Kernel Ridge Regression (KRR)**

Kernel ridge regression is similar to solving , but adding an identity matrix scaled by a choice in lambda, as follows:

, where is the kernel matrix, is the ridge scalar and is the response variable.

I was not able to use the same function call for KRR so I wrote the computations out manually. I will include the code below.

The kernel type is a polynomial of degree 2. I experimented with offset = (1,10,100,1000). Offset = 100 gave the best results...by mere hundredths of a percent. Thus, these comparisons are not reported. The library {kernlab} was used to call ‘polydot’ and ‘kernelMatrix’ I save my kernel function in kern:

*kern <- polydot(degree = 2, offset = 100)*

'kernelMatrix' requires the vectors to be matrices:

*x.training <- as.matrix(training[,c(2,3,4,5)])  
x.testing <- as.matrix(testing[,c(2,3,4,5)])*

Kernel Matrix K for training set:

*kern.train = kernelMatrix(kern,x.training)*

Kernel Matrix k for testing set:

*kern.test = kernelMatrix(kern,x.training,x.testing)*

The above code does just what *fit = lm(…)* did previously. I introduce the ridge, , to kern.train before we solve for our coefficients. I wrote a for-loop to test for an optimal lambda. The Lambda which gave the smallest RMSE was chosen. I only looked at the RMSE for the testing set since accuracy in the testing set is what we are ultimately after. Lambda = .81 was the most optimal. I combed through .001-10. I will include tables after this code.

*lambda = seq(.01, 1, .005)  
findlambda <-data.frame(1:258)  
error <- data.frame(Lambda = 1:length(lambda), Error = 1:length(lambda))*

***for****(i in 1:length(lambda)){  
 beta\_hat\_temp = solve(kern.train+lambda[i]\*diag(length(training$JJ)), tol = 1e- 23)%\*%as.matrix(training$JJ)*

*findlambda[,i] <- t(kern.test)%\*%beta\_hat\_temp*

*error[i,1] <- lambda[i]  
 error[i,2] <- sd(testing$JJ - findlambda[,i])/mean(training$JJ)*

***}***

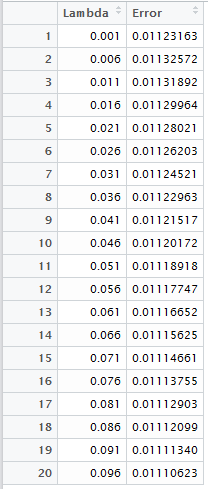
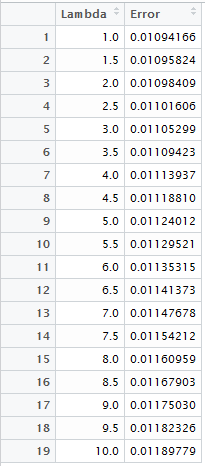
*best.lambda = error[which(error[,2] == min(error[,2])),]  
lambda = best.lambda$Lambda #assigns best lambda found*

Now that I have found the most optimal lambda, I can carry on with the KRR.

*beta\_hat = solve(kern.train+lambda\*diag(length(training$JJ)))%\*%as.matrix(training$JJ)*

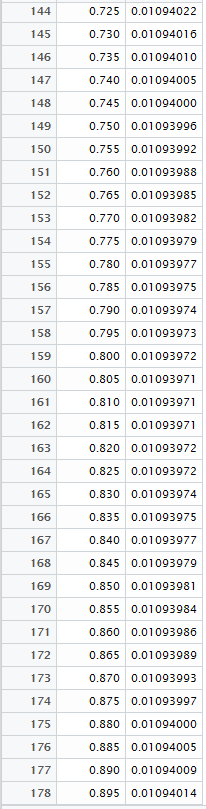
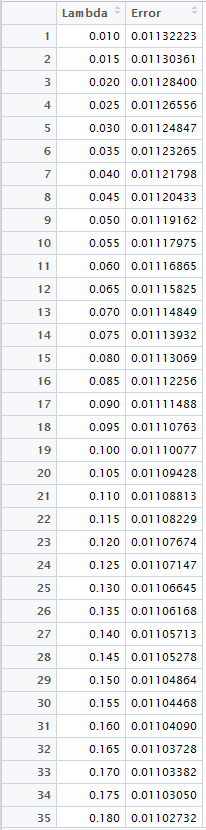
# Computes the predictions of KRR

*training$Predict.krr <- t(kern.train) %\*% beta\_hat  
testing$Predict.krr <- t(kern.test)%\*%beta\_hat*



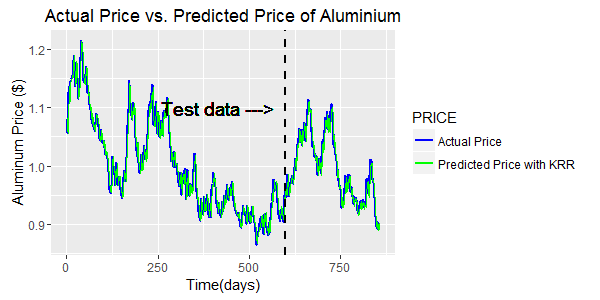
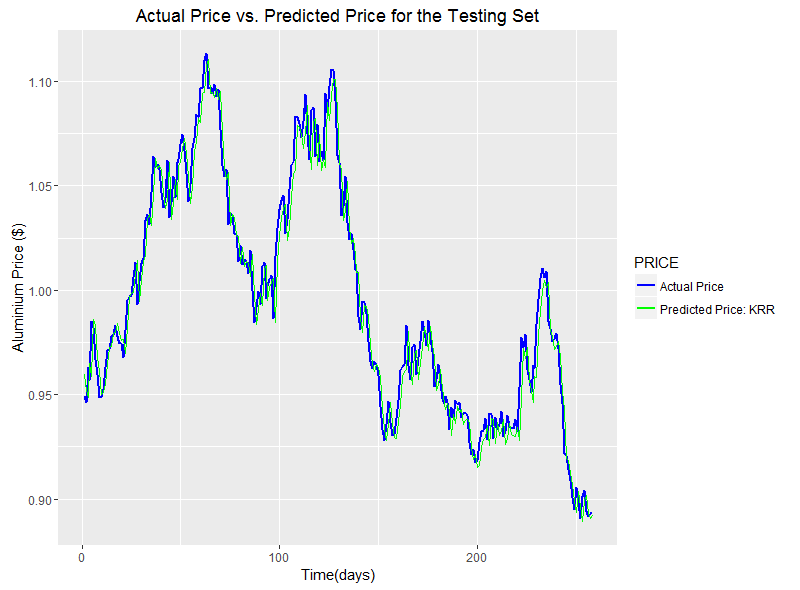
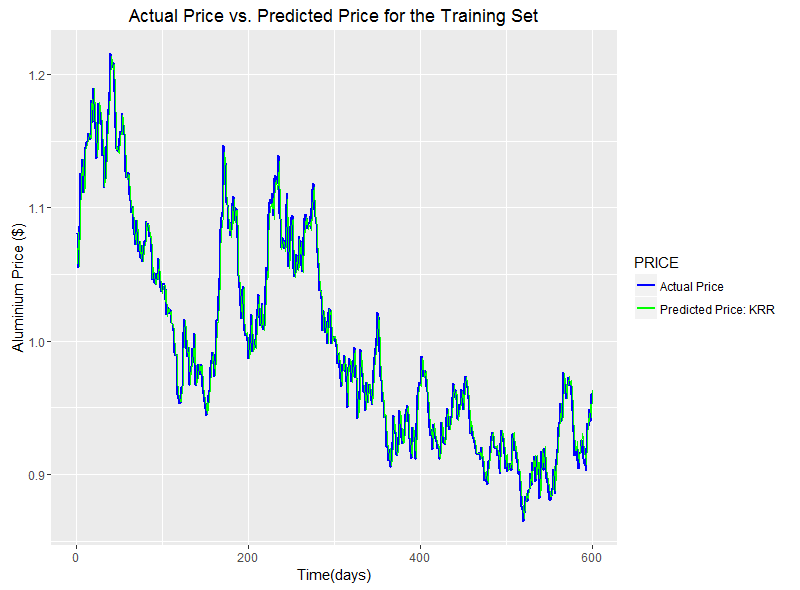
Negligible difference to be sure, but modern computing makes it so that wringing out a few 1000ths of a percent takes virtually no extra time.

…

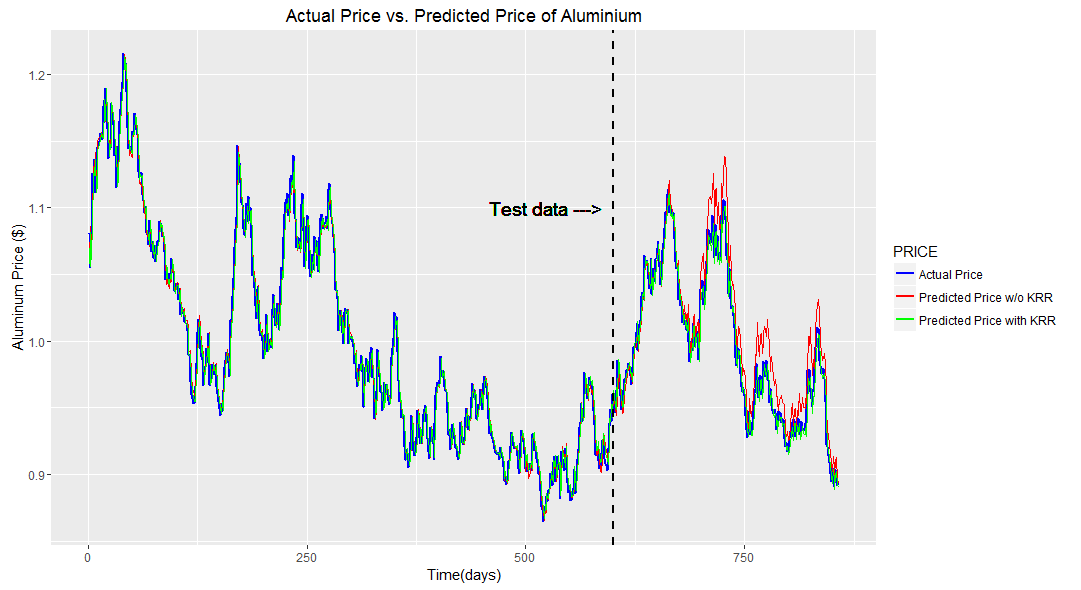


We see at the 161st entry that the lowest RMSE error we achieve is with lambda = .81. This gives RMSE.test = .01093971

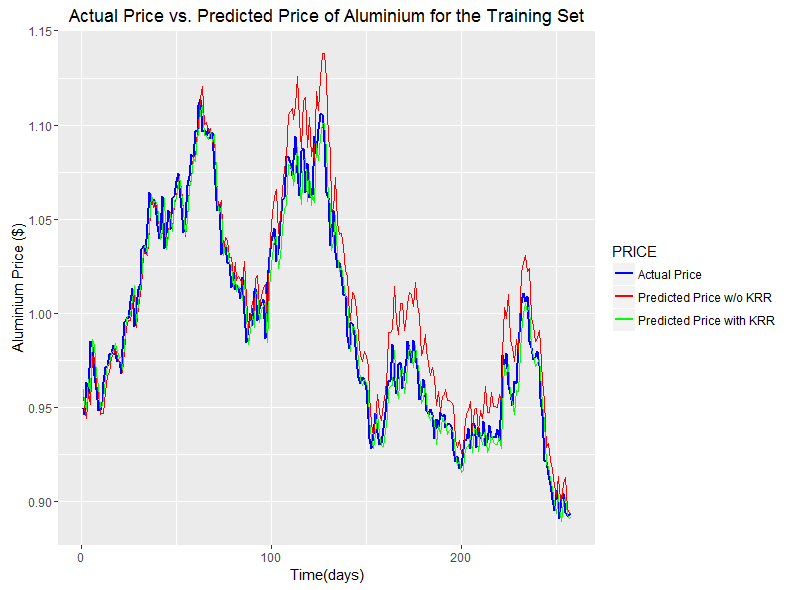
**Plotting predicted values with KRR**



We can already see how much more closely the predicted curve with KRR follows the actual price compared to without KRR. Let’s look at a plot with both predictions.

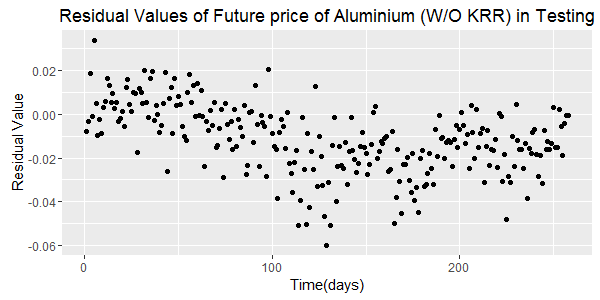
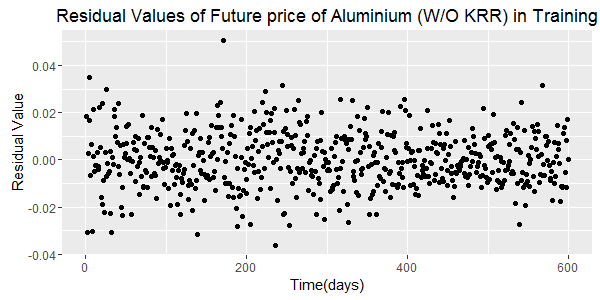


Training predictions look very similar but we see much improvement in the testing set. Let’s take a closer look.

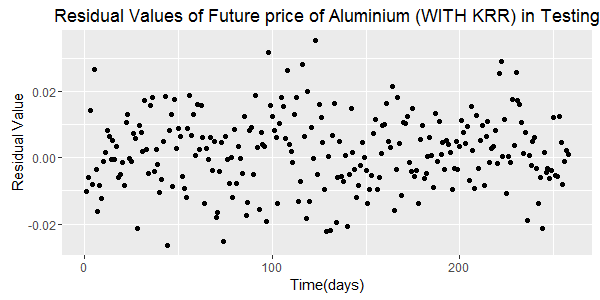
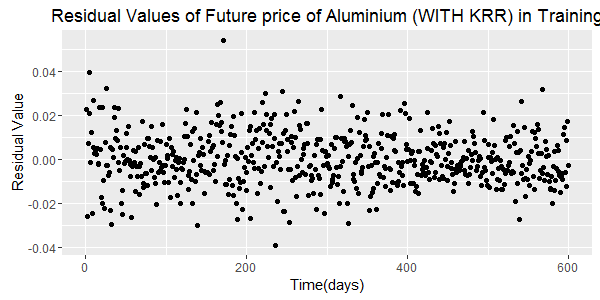


Notice the 100th day into testing where non-KRR had trouble keeping the predictions close. KRR does a markedly better job at keep the predictions close in value. I will now include performance tables and residual plots to verify these claims.

**Residual Plots:  
with and without KRR**



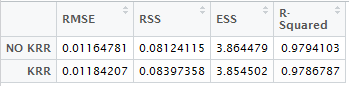
The pattern in the testing period without KRR is not random.  
 following plots illustrate how KRR ‘cleans up’ the bias seen in the residuals.



There is now very little pattern associated with the testing period’s residual plot. We can now feel much more confident that our model is not introducing too much bias.

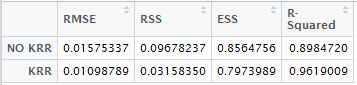
Finally, below I include the performance tables so that we can inspect the RMSE, RSS, ESS and r-squared values.

**Training Performance**



The training period does not appear to benefit from KRR.

**Testing Performance**



However, the testing period undergoes a considerable increase in accuracy by introducing KRR.

**Conclusion**

After running through the motions of non-linear kernel regression with and without a ridge, I have come to the conclusion that the increase in accuracy (reduction in residual errors) is well worth the minute increase in computational cost. For improvement, one could try using a subset of explanatory variables, namely, removing steel. This could result in worse accuracy but of all the explanatory variables it has the least impact. I did not attempt to properly optimize the offset in my polynomial kernel. There is a good chance that a better choice of offset exists. But, I found offset = (1,10,100,1000) to have very similar results when performing no KRR and KRR with lambda = 0.1.