11.1.1 – Please review the included R file 11.1.1.R along with the explanation

First, I loaded and inspected the data. Unlike in some previous assignments using this dataset, I checked for multicollinearity, but decided not to remove any correlated predictors, as I wanted to see how each of the models would deal with that issue.

I then split the data into to training and test data, using 80% of the data for the training set and the remaining 20% for the test set.

I then created a basic model with only one predictor and ran a forward stepwise regression using the stepAIC function from the MASS package. This function can perform both forward and backward stepwise regression based on AIC. The final model from the forward step used the factors M + Po1 + Ineq + Ed + Prob + Wealth and had an AIC of 403.13. The predictor added at each step and the resulting model quality is shown below.

Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
1			35	4260568	435.1978
2	1	1831892.71	34	2428676	416.4017
3	1	269794.94	33	2158881	414.0448
4	1	428184.44	32	1730696	407.8653
5	1	272778.27	31	1457918	403.5193
6	1	90998.64	30	1366919	403.1346

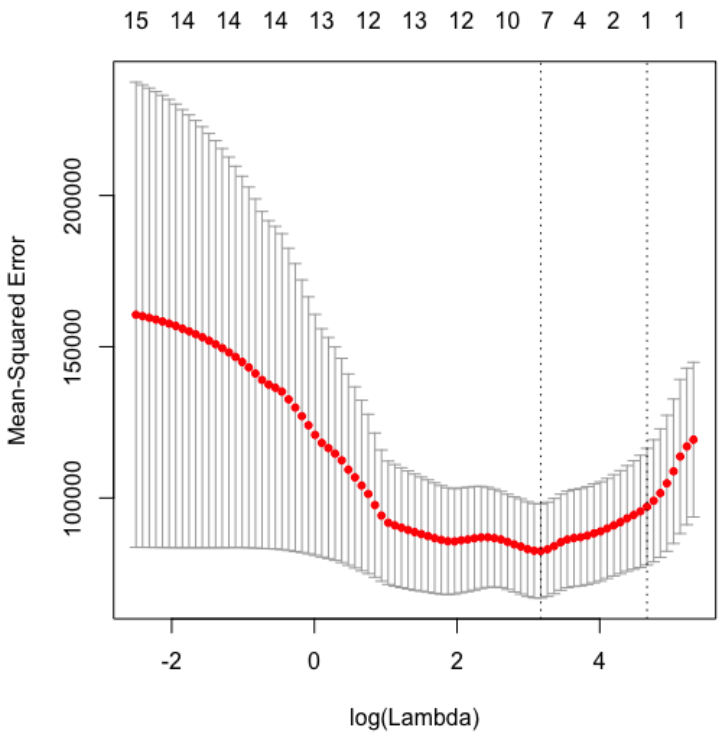
Running backward stepwise regression did not remove any of these predictors as doing so did not improve AIC.

I then used the test data and hypothetical test city from previous assignments to test the model. The R2 using the predictors listed above was .83, and the model predicted a crime rate of 940 in the test city.

11.1.2 – Please review the included R file 11.1.2.R along with the explanation

After loading the data, I first scaled it as is required when using constraint-based models like Lasso. I then split the data into separate data frames of input and response variables, split each into training and test sets, and converted each of the input and response training and test sets into matrices as required by glmnet. I created a Lasso model in GLM net using the training data. I used the packages cross validation feature to find the best value of lambda that minimized

mean squared error. The value of lambda that minimized MSE was around 3 and resulted in a model using 7 factors as shown in the chart
below. This is generally consistent with what I found with the stepwise model and manual variable selection a few weeks ago.

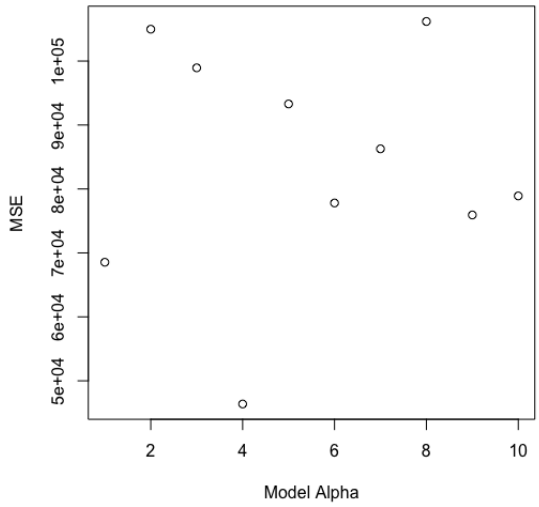


R2 with this model was .72 and the prediction on the hypothetical test city was a crime rate of 1003.

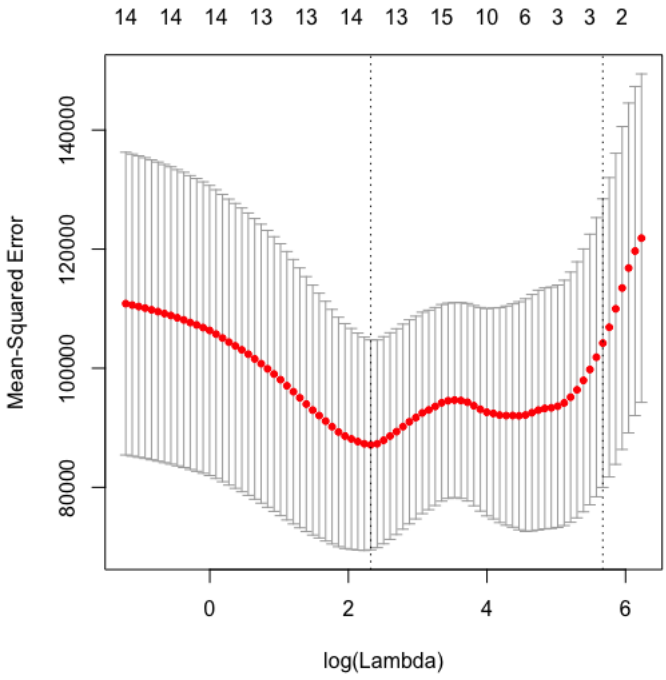
11.1.3 – Please review the included R file 11.1.3.R along with the explanation

After loading the data, I followed similar preparation, splitting, and scaling described in the lasso example above. However, rather than only splitting into training and test data I also created a validation data set to be used when selecting the best alpha value for the model.

I then initially created basic elastic net and ridge models on the training data, but decided to loop through alpha values between 0 (Ridge Regression) and 1 (LASSO), and use cross validation on the training data with each model to find the Lambda value that minimized MSE on the training data. With these 11 models I then calculated MSE for each model on the validation data and plotted it on the chart below.



Model 4 had the lowest MSE on the validation data set when using the lambda value that minimized MSE, so I decided to move forward with that model. Interestingly this model still includes most of the predictors, unlike the previous model found using stepwise, LASSO, and manual variable selection.



This model has an R2 of .79, and predicted a crime rate of 1142 on the hypothetical test city.

Additional: Choice of best Lambda Values of cross validated elastic net models.

While I was looking into the Lambda selection for problem 11.1.2 and 11.1.3 I noticed that the glmnet documentation includes two potential options for the Lambda value used when predicting with a cross validation fitted model: “Min” which uses the Lambda value which minimized Mean Squared Error on the training data, or “1se” which uses the simplest model possible that is within 1 standard error of the minimum MSE. Interestingly, this choice has a massive difference in R2 on the same model.

For example, the model I used in 11.1.3 which has an R2 of .79 when using the “min” Lambda has an R2 of .38 when using the “1se” Lambda. This is likely because the more complicated model is able to overfit the very limited number of data points we have in this data set while the simpler model does not. However, the simple model in these cases often only used 2-3 predictors, leading to poor performance. By manually entering Lambda values between these two extremes, I was able to find relatively simple models that still had fairly high R2 values.