Stat 4620 Project

Cameron Erdman, Colin Walsh, Maggie Miller, Rithika Annareddy, Zak Taylor

2023-12-08

Report

The Boston dataset contains the housing values of 506 suburbs in the Boston area. The dataset contains 13 predictors and 1 mystery response variable that we will try to predict statistical analysis. The 13 predictors in this dataset are as follows:
-crim: per capita crime rate by town.
-zn: Proportion of residential land zoned for lots over 25,000 sq.ft.
-indus: Proportion of non-retail business acres per town.
-chas: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
-nox: Nitrogen oxides concentration (parts per 10 million).
-rm: Average number of rooms per dwelling.
-age: Proportion of owner-occupied units built prior to 1940.
-dis: Weighted mean of distances to five Boston employment centers.
-rad: Index of accessibility to radial highways.
-tax: Full-value property-tax rate per $10,000.
-ptratio: Pupil-teacher ratio by town.
-lstat: Lower status of the population (percent).
-medv: Median value of owner-occupied homes in $1000s.

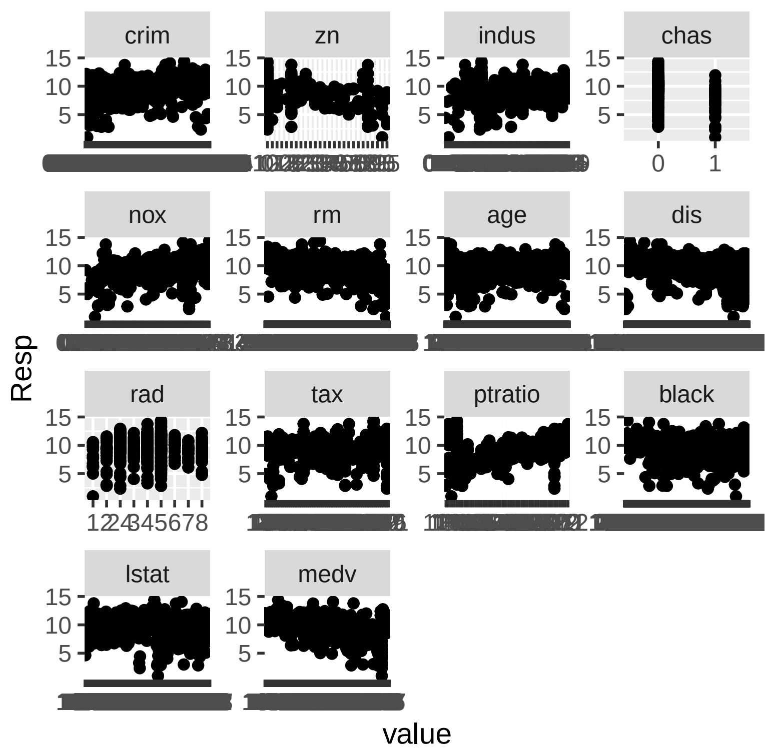
```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv  Resp
## 1 24.0 7.713
## 2 21.6 9.453
## 3 34.7 9.604
## 4 33.4 7.980
## 5 36.2 8.164
## 6 28.7    NA
```

Before we can do any model building, we must first clean up our dataset and explore the variables we will be using for said model building. In our exploration of data, we found 10 missing values within our dataset. All 10 of the missing values were found to be from our mystery response variables, so we compared the observations where the response was missing to those that contained values for the response to see if there was a rhyme or reason to omit these responses. To do this we compared the means and standard deviations of the predictor variables of responses that were missing to those that had a response recorded. As you can see from the analysis below, we concluded that response variables were likely omitted randomly, as we saw no significant difference between the means of observations with missing responses compared to those with a response variable. Since it seems to be random whether or not the response was not recorded, we decided to omit observations without a response, bringing our dataset to 496 total observations.

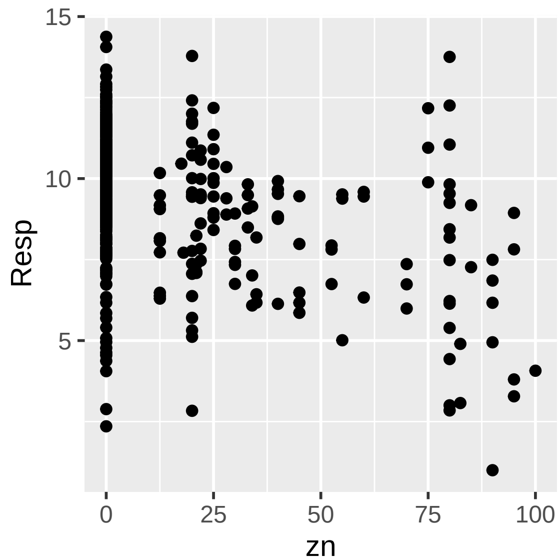
```
## [1] "Means:"

##         crim          zn       indus         nox          rm         age
##    3.6135236  11.3636364  11.1367787   0.5546951   6.2846344  68.5749012
##          dis         rad         tax     ptratio       black       lstat
##    3.7950427   9.5494071 408.2371542  18.4555336 356.6740316  12.6530632
##         medv
##   22.5328063

##       crim         zn      indus        nox         rm        age        dis
##   4.720968  12.500000  10.462000   0.568200   6.491600  63.760000   3.467570
##        rad        tax    ptratio      black      lstat       medv
##  12.200000 463.600000  18.030000 348.968000  12.177000  25.530000
```

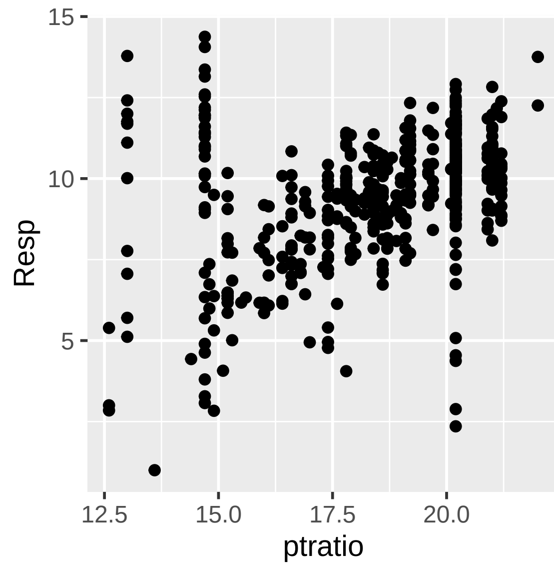
Now that we have cleaned up our dataset, we can look into the predictors and explore their relationship with the response. First, we plotted each predictor against the response variable to see if there were any concerning relationships between a certain predictor and the response, as can be seen in the graph below.

```
## Warning: attributes are not identical across measure variables; they will be
## dropped
```

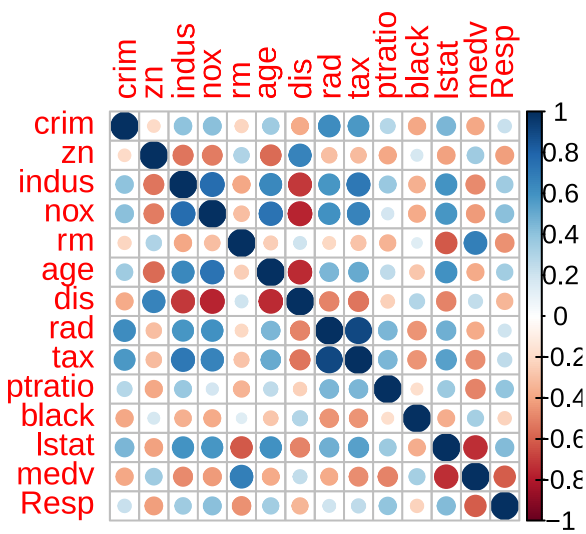


We see that there are a few potential similarities between zn and ptratio and the response. However, after further investigation (can be found in appendix) we found that this concern is not necessary and we can continue on with our exploration of the variables.





To further explore our data, we looked into the correlation of each variable through a correlation matrix, as can be seen in the table below.



Through this we can come to a few key conclusions about our data. We found that there are no variables with zero corollary effect with the response. Of all the predictors, the variable medv has the strongest correlation with a -0.6 implying that as the median value of the house decreases, the response variable increases. We also have to take note of the potential interaction effect between the variables dis, nox, indus, tax, rad, and age, as they are all decently correlated with each other. Keeping these takeaways in mind, we moved into our model building to try to predict the response.

First things first, we must randomly split our data into a training set and a testing set. We do this so we can build models on the training set and test their effectiveness on the testing set. We decided to start our model building with the most intuitive model, the linear model. However, with 13 predictors, our model could suffer from the overfitting and an increased variance from too many features. In order to dampen the effect of overfitting we selected a model using best subset selection. We took every possible model with k=0,..,13 predictors, taking the model with the lowest RSS from each model with k predictors. After we found the best model with k predictors, we compared the Cp of each model (since Cp is an unbiased estimate of test MSE) and took the model with the lowest Cp, which can be seen below.

```
## Least Squares Model Test MSE
##                     1.725316
```

```
## [1] "Model Variables Chosen and their coefficients: "
```

```
##  (Intercept)           zn        chas1          nox          tax      ptratio
##  7.265979027 -0.009863672 -1.790245548  5.613405527 -0.003893106  0.171698560
##         medv
## -0.101323276
```

Although the least squares model seems to perform well, we should still explore other possible models. We first started by looking at shrinkage through Ridge Regression and LASSO. These methods are very useful when trying to avoid overfitting, which is a concern when you have a large amount of predictors. Both are able to control the bias-variance tradeoff through a shrinkage parameter lambda, with Ridge Regression doing a better job at capturing a lot of variables providing small effects while LASSO is better at dimension reduction and variable selection. However, after creating both models with our training set and testing them against the testing set, we find that they have comparable, yet slightly worse test MSEs compared to our least squares model. Due to simplicity and interpretability of the least squares model, we chose to keep that model over the Ridge Regression and LASSO models.

```
## Ridge Model Test MSE Lasso Model Test MSE
##             1.733120             1.748026
```

Next we decided to look into more dimension reduction methods, PCR and PLS. PCR is unsupervised, so it does not have any information on the relationship of the response variable with the predictors. Since it is unsupervised, it can help uncover relationships within the data we did not know about. However, the PCR model performs poorly in our tests, so we decided to look into a supervised version of PCR, PLS. Although this model performed better, as seen by the lower test MSE, it still is not as good as our least squares model, so we continue to search for a better model.

```
## PCR Model Test MSE PLS Model Test MSE
##           2.353810           1.988015
```

We decided that regression trees could be a good place to look next. It could be helpful for us because it handles interactions well, which we pointed out as a potential issue when we explored our data. However, a regular tree is often poor at predicting due to it being prone to a larger variance. This can be remedied through algorithms such as bagging, random forest, and boosting. However, these methods really hurt the interpretability of our model. Bagging involves building many trees and averaging them out, and our model tested very well, with a test MSE lower than that of our least squares model. The same held true for our random forest model, which tries to de-correlate the trees gathered from bagging, although worse than the bagging model. Boosting performed worse than both other tree methods and will be disregarded. An interesting thing to note is the variable medv was the most important for all 3 regression tree models, which led us to look into the relationship between and response variable and medv by itself.

```
## Using 500 trees...
```

```
##      Boosting Model MSE Random Forest Model MSE       Bagging Model MSE
##            1.7732873559            0.0002884102            0.0015472589
```

```
## [1] "Bagging Importance"
```

```
##           %IncMSE IncNodePurity
## crim     4.858007      52.50145
## zn       1.736928      28.76410
## indus    2.911727      18.72019
## chas     5.000729      15.16808
## nox      8.365013     119.62643
## rm       6.830183     162.27975
## age      1.502414      35.96164
## dis      7.824878      82.79878
## rad      2.892727      10.71497
## tax      1.996962      13.87014
## ptratio 14.589038     159.37620
## lstat    3.085921      48.37970
## medv    10.865046     308.98955
```

To test the relationship between the response and medv, we decided to take some splines. However, all of our models seem to perform poorly in tests, so we will disregard these models.

```
## Warning in bs(medv, degree = 3L, knots = 21.2, Boundary.knots = c(5.6, 50: some
## 'x' values beyond boundary knots may cause ill-conditioned bases
```

```
## [1] 2.455012
```

```
## [1] 2.435072
```

```
## Warning in smooth.spline(Boston_train$medv, Boston_train$Resp, cv = T):
## cross-validation with non-unique 'x' values seems doubtful
```

```
## [1] 2.51081
```

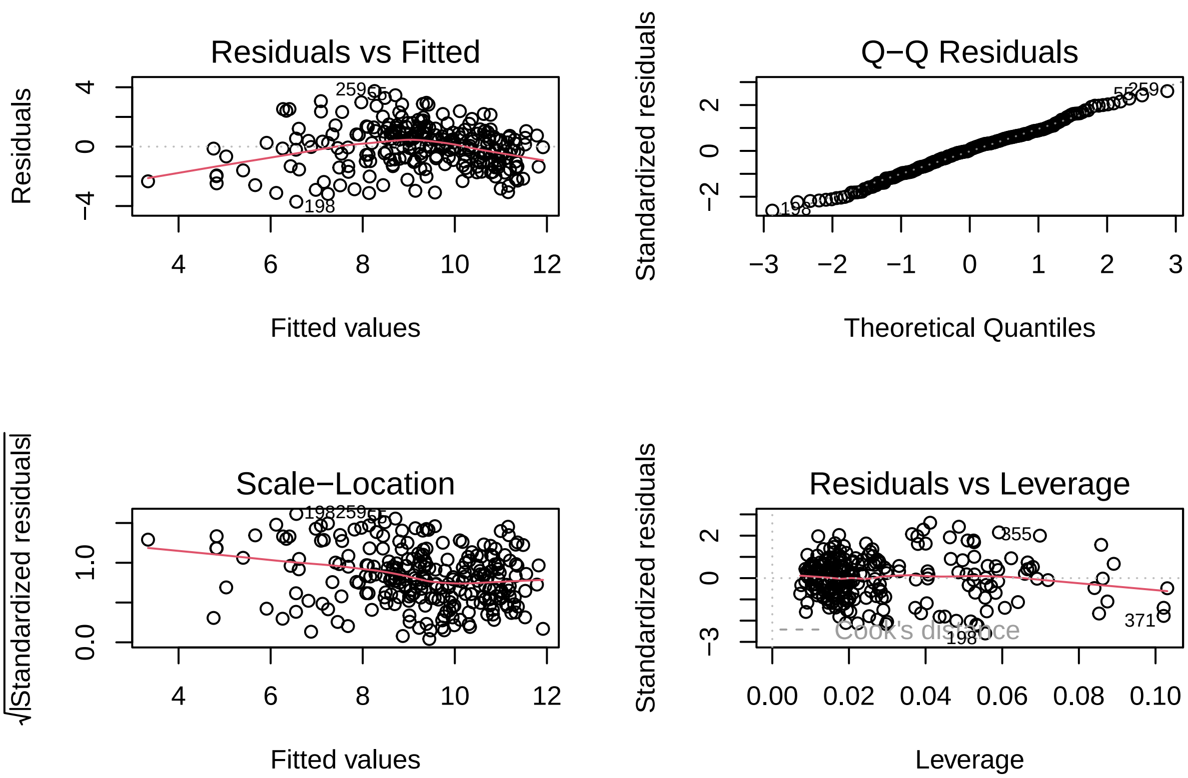
```
## Smoothing Splines MSE     Cubic Splines MSE   Natural Splines MSE
##              2.510810              2.455012              2.435072
```

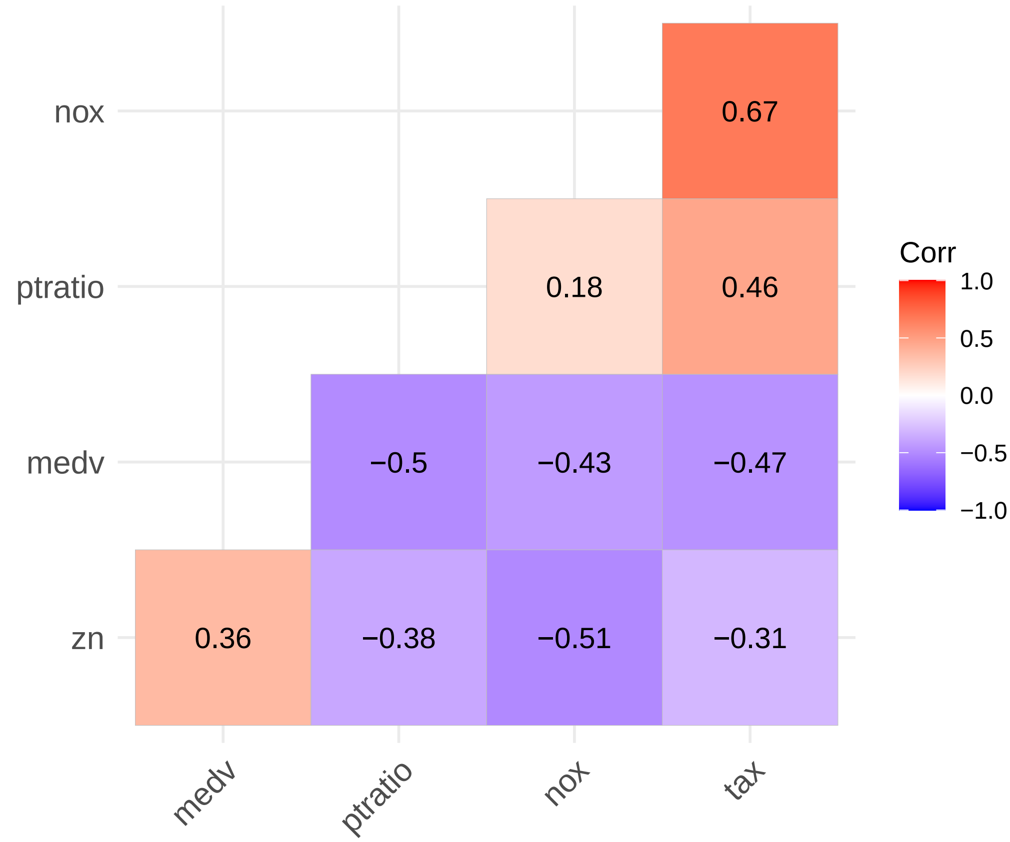
Now that we built some models and compared them, we must decide which model we think is best. Our decision came down to two models, the least squares model and the bagging model. The bagging model has the advantage of performing slightly better in our testing of predictive power. However, the least squares model is much easier to interpret. In the end, we decided that the interpretability of the least squares model outweighed the slightly better predictive prowess of the bagging model. On top of the concern of interpretability for bagging, we also run into the concern of correlated trees, which is addressed in random forest, however, random forest performed worse predictively and still runs into the issue of being hard to interpret. Due to these concerns, we have decided to select the least squares model as our model of choice. We can see the summary of our final model in the graph below.

```
##
## Call:
## lm(formula = Resp ~ zn + chas + nox + tax + ptratio + medv, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7122 -0.9803  0.0662  0.9351  3.7438
##
## Coefficients:
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.2659790  1.4837636   4.897 1.78e-06 ***
## zn          -0.0098637  0.0046650  -2.114  0.03551 *
## chas1       -1.7902455  0.3363955  -5.322 2.35e-07 ***
## nox          5.6134055  1.2977033   4.326 2.23e-05 ***
## tax         -0.0038931  0.0008491  -4.585 7.29e-06 ***
## ptratio      0.1716986  0.0569046   3.017  0.00282 **
## medv        -0.1013233  0.0118712  -8.535 1.58e-15 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 1.467 on 241 degrees of freedom
## Multiple R-squared:  0.5409, Adjusted R-squared:  0.5294
## F-statistic: 47.32 on 6 and 241 DF,  p-value: < 2.2e-16
```

Now that we have selected the least squares model, let’s take a look at our assumptions of a linear model and see if our data violates these assumptions. For the least squares model it was assumed that the model is linear in parameters, and when graphed the data does not look perfectly linear. However the other assumptions are that residual values are normally distributed and the variance of the residuals is approximately constant which can be seen in the normal qq plot and scale-location plot respectively. Then the multicollinearity assumption is also met which can be shown in the correlation plots, where none of the correlations are higher than .8 (chas has to be excluded in the charts since it is a factor variable).





So in conclusion, we have learned that our data has quite a few significant predictors for our response, however, not every predictor was useful. We learned that our response variable has a few factors it depends on heavily and a few that really do not have a big impact. Along the way we learned that our response is at least somewhat linear, as we can see from our graphs, models that tend to be less flexible tend to perform better. In our exploration and model building, digging into each model brought us to another potential model that could potentially perform better. Although in the end we decided our original model was our best one, we feel as if there was strong reasoning behind exploring other models.

Appendix

Section 2.1

Loading in the data

```
load("./Boston_Stat4620_2023.RData") #Loads as Boston.Stat4620
df <- Boston.Stat4620 #Copy data to df for manipulation
head(df) #taking a peak at the data to check load was successful
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
```

```
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv  Resp
## 1 24.0 7.713
## 2 21.6 9.453
## 3 34.7 9.604
## 4 33.4 7.980
## 5 36.2 8.164
## 6 28.7    NA
```

Checking the metadata

```r
##ran this to get below information
#library(MASS)
#?Boston
```

From the Boston metadata: This data frame contains the following columns:

crim: per capita crime rate by town.

zn: proportion of residential land zoned for lots over 25,000 sq.ft.

indus: proportion of non-retail business acres per town.

chas: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).

nox: nitrogen oxides concentration (parts per 10 million).

rm: average number of rooms per dwelling.

age: proportion of owner-occupied units built prior to 1940.

dis: weighted mean of distances to five Boston employment centres.

rad: index of accessibility to radial highways.

tax: full-value property-tax rate per $10,000.

ptratio: pupil-teacher ratio by town.

black: $1000(Bk - 0.63)^2$ where Bk is the proportion of black individuals by town.

lstat: lower status of the population (percent).

medv: median value of owner-occupied homes in $1000s.

Resp: specific to this data set, the response variable.

Describing the variables

```r
summary(df)
```

```
##       crim                zn             indus       chas         nox
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   0:471   Min.   :0.3850
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1: 35   1st Qu.:0.4490
##  Median : 0.25651   Median :  0.00   Median : 9.69           Median :0.5380
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14           Mean   :0.5547
```

```
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10           3rd Qu.:0.6240
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74           Max.   :0.8710
##
##        rm             age              dis              rad
##  Min.   :3.561   Min.   :  2.90   Min.   : 1.130   Min.   : 1.000
##  1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100   1st Qu.: 4.000
##  Median :6.208   Median : 77.50   Median : 3.207   Median : 5.000
##  Mean   :6.285   Mean   : 68.57   Mean   : 3.795   Mean   : 9.549
##  3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188   3rd Qu.:24.000
##  Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.000
##
##       tax           ptratio          black            lstat
##  Min.   :187.0   Min.   :12.60   Min.   :  0.32   Min.   : 1.73
##  1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38   1st Qu.: 6.95
##  Median :330.0   Median :19.05   Median :391.44   Median :11.36
##  Mean   :408.2   Mean   :18.46   Mean   :356.67   Mean   :12.65
##  3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23   3rd Qu.:16.95
##  Max.   :711.0   Max.   :22.00   Max.   :396.90   Max.   :37.97
##
##       medv            Resp
##  Min.   : 5.00   Min.   : 1.000
##  1st Qu.:17.02   1st Qu.: 8.582
##  Median :21.20   Median : 9.778
##  Mean   :22.53   Mean   : 9.435
##  3rd Qu.:25.00   3rd Qu.:10.712
##  Max.   :50.00   Max.   :14.376
##                  NA's   :10
```

```r
sapply(df, class)
```

```
##      crim        zn     indus      chas       nox        rm       age       dis
## "numeric" "numeric" "numeric"  "factor" "numeric" "numeric" "numeric" "numeric"
##       rad       tax   ptratio     black     lstat      medv      Resp
## "integer" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
```

All data is of type numeric with exception of the Charles river dummy variable being a factor and the rad index being an integer.

Checking fill levels

```r
sum(is.na(df))
```

```
## [1] 10
```

We see there are 10 NA's in our data frame.

```r
sapply(df, function(x) sum(is.na(x)))
```

```
##    crim      zn   indus    chas     nox      rm     age     dis     rad     tax
##       0       0       0       0       0       0       0       0       0       0
## ptratio   black   lstat    medv    Resp
##       0       0       0       0      10
```

They are all in the Resp variable.

```r
df_na <- df[rowSums(is.na(df)) > 0,]
df_na
```

```
##         crim zn indus chas   nox    rm   age    dis rad tax ptratio  black
## 6    0.02985  0  2.18    0 0.458 6.430  58.7 6.0622   3 222    18.7 394.12
## 116  0.17134  0 10.01    0 0.547 5.928  88.2 2.4631   6 432    17.8 344.91
## 193  0.08664 45  3.44    0 0.437 7.178  26.3 6.4798   5 398    15.2 390.49
## 194  0.02187 60  2.93    0 0.401 6.800   9.9 6.2196   1 265    15.6 393.37
## 258  0.61154 20  3.97    0 0.647 8.704  86.9 1.8010   5 264    13.0 389.70
## 383  9.18702  0 18.10    0 0.700 5.536 100.0 1.5804  24 666    20.2 396.90
## 455  9.51363  0 18.10    0 0.713 6.728  94.1 2.4961  24 666    20.2   6.68
## 470 13.07510  0 18.10    0 0.580 5.713  56.7 2.8237  24 666    20.2 396.90
## 480 14.33370  0 18.10    0 0.614 6.229  88.0 1.9512  24 666    20.2 383.32
## 496  0.17899  0  9.69    0 0.585 5.670  28.8 2.7986   6 391    19.2 393.29
##     lstat medv Resp
## 6    5.21 28.7   NA
## 116 15.76 18.3   NA
## 193  2.87 36.4   NA
## 194  5.03 31.1   NA
## 258  5.12 50.0   NA
## 383 23.60 11.3   NA
## 455 18.71 14.9   NA
## 470 14.76 20.1   NA
## 480 13.11 21.4   NA
## 496 17.60 23.1   NA
```

These are the 10 rows containing NA values in the Resp variable.

```r
summary(df)
```

```
##       crim                zn             indus       chas         nox
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   0:471   Min.   :0.3850
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1: 35   1st Qu.:0.4490
##  Median : 0.25651   Median :  0.00   Median : 9.69           Median :0.5380
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14           Mean   :0.5547
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10           3rd Qu.:0.6240
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74           Max.   :0.8710
##
##        rm             age              dis              rad
##  Min.   :3.561   Min.   :  2.90   Min.   : 1.130   Min.   : 1.000
##  1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100   1st Qu.: 4.000
##  Median :6.208   Median : 77.50   Median : 3.207   Median : 5.000
##  Mean   :6.285   Mean   : 68.57   Mean   : 3.795   Mean   : 9.549
##  3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188   3rd Qu.:24.000
##  Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.000
##
##       tax           ptratio          black            lstat
##  Min.   :187.0   Min.   :12.60   Min.   :  0.32   Min.   : 1.73
##  1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38   1st Qu.: 6.95
##  Median :330.0   Median :19.05   Median :391.44   Median :11.36
##  Mean   :408.2   Mean   :18.46   Mean   :356.67   Mean   :12.65
```

```
##  3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23   3rd Qu.:16.95
##  Max.   :711.0   Max.   :22.00   Max.   :396.90   Max.   :37.97
##
##       medv            Resp
##  Min.   : 5.00   Min.   : 1.000
##  1st Qu.:17.02   1st Qu.: 8.582
##  Median :21.20   Median : 9.778
##  Mean   :22.53   Mean   : 9.435
##  3rd Qu.:25.00   3rd Qu.:10.712
##  Max.   :50.00   Max.   :14.376
##                  NA's   :10
```

```r
summary(df_na)
```

```
##       crim               zn           indus          chas        nox
##  Min.   : 0.02187   Min.   : 0.0   Min.   : 2.180   0:10   Min.   :0.4010
##  1st Qu.: 0.10781   1st Qu.: 0.0   1st Qu.: 3.572   1: 0   1st Qu.:0.4803
##  Median : 0.39526   Median : 0.0   Median : 9.850          Median :0.5825
##  Mean   : 4.72097   Mean   :12.5   Mean   :10.462          Mean   :0.5682
##  3rd Qu.: 9.43198   3rd Qu.:15.0   3rd Qu.:18.100          3rd Qu.:0.6388
##  Max.   :14.33370   Max.   :60.0   Max.   :18.100          Max.   :0.7130
##
##        rm             age              dis             rad
##  Min.   :5.536   Min.   :  9.90   Min.   :1.580   Min.   : 1.0
##  1st Qu.:5.767   1st Qu.: 35.77   1st Qu.:2.079   1st Qu.: 5.0
##  Median :6.330   Median : 72.80   Median :2.647   Median : 6.0
##  Mean   :6.492   Mean   : 63.76   Mean   :3.468   Mean   :12.2
##  3rd Qu.:6.782   3rd Qu.: 88.15   3rd Qu.:5.253   3rd Qu.:24.0
##  Max.   :8.704   Max.   :100.00   Max.   :6.480   Max.   :24.0
##
##       tax           ptratio          black            lstat
##  Min.   :222.0   Min.   :13.00   Min.   :  6.68   Min.   : 2.870
##  1st Qu.:296.5   1st Qu.:16.15   1st Qu.:384.92   1st Qu.: 5.143
##  Median :415.0   Median :18.95   Median :391.89   Median :13.935
##  Mean   :463.6   Mean   :18.03   Mean   :348.97   Mean   :12.177
##  3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:393.93   3rd Qu.:17.140
##  Max.   :666.0   Max.   :20.20   Max.   :396.90   Max.   :23.600
##
##       medv            Resp
##  Min.   :11.30   Min.   : NA
##  1st Qu.:18.75   1st Qu.: NA
##  Median :22.25   Median : NA
##  Mean   :25.53   Mean   :NaN
##  3rd Qu.:30.50   3rd Qu.: NA
##  Max.   :50.00   Max.   : NA
##                  NA's   :10
```

Too much information, I'm going to look specifically at the means and standard deviations.

```r
print("Means:")
```

```
## [1] "Means:"
```

```r
sapply(df[,-c(4, 15)], function(x) mean(x))
```

```
##        crim          zn       indus         nox          rm         age
##   3.6135236  11.3636364  11.1367787   0.5546951   6.2846344  68.5749012
##         dis         rad         tax     ptratio       black       lstat
##   3.7950427   9.5494071 408.2371542  18.4555336 356.6740316  12.6530632
##        medv
##  22.5328063
```

```r
sapply(df_na[,-c(4, 15)], function(x) mean(x))
```

```
##       crim         zn      indus        nox         rm        age        dis
##   4.720968  12.500000  10.462000   0.568200   6.491600  63.760000   3.467570
##        rad        tax    ptratio      black      lstat       medv
##  12.200000 463.600000  18.030000 348.968000  12.177000  25.530000
```

```r
print("Standard Deviations:")
```

```
## [1] "Standard Deviations:"
```

```r
sapply(df[,-c(4, 15)], function(x) sd(x))
```

```
##        crim          zn       indus         nox          rm         age
##   8.6015451  23.3224530   6.8603529   0.1158777   0.7026171  28.1488614
##         dis         rad         tax     ptratio       black       lstat
##   2.1057101   8.7072594 168.5371161   2.1649455  91.2948644   7.1410615
##        medv
##   9.1971041
```

```r
sapply(df_na[,-c(4, 15)], function(x) sd(x))
```

```
##        crim          zn       indus         nox          rm         age
##   6.0451317  22.2673154   7.0783988   0.1079247   0.9487810  32.5849386
##         dis         rad         tax     ptratio       black       lstat
##   1.9672980  10.2610374 186.2162423   2.5802885 121.2483256   7.1413849
##        medv
##  11.4417413
```

From a quick check at the means and standard deviations, it seems as though the NA data in Resp is random.

```r
df_no_na <- na.omit(df)
sapply(df_no_na, function(x) sum(is.na(x)))
```

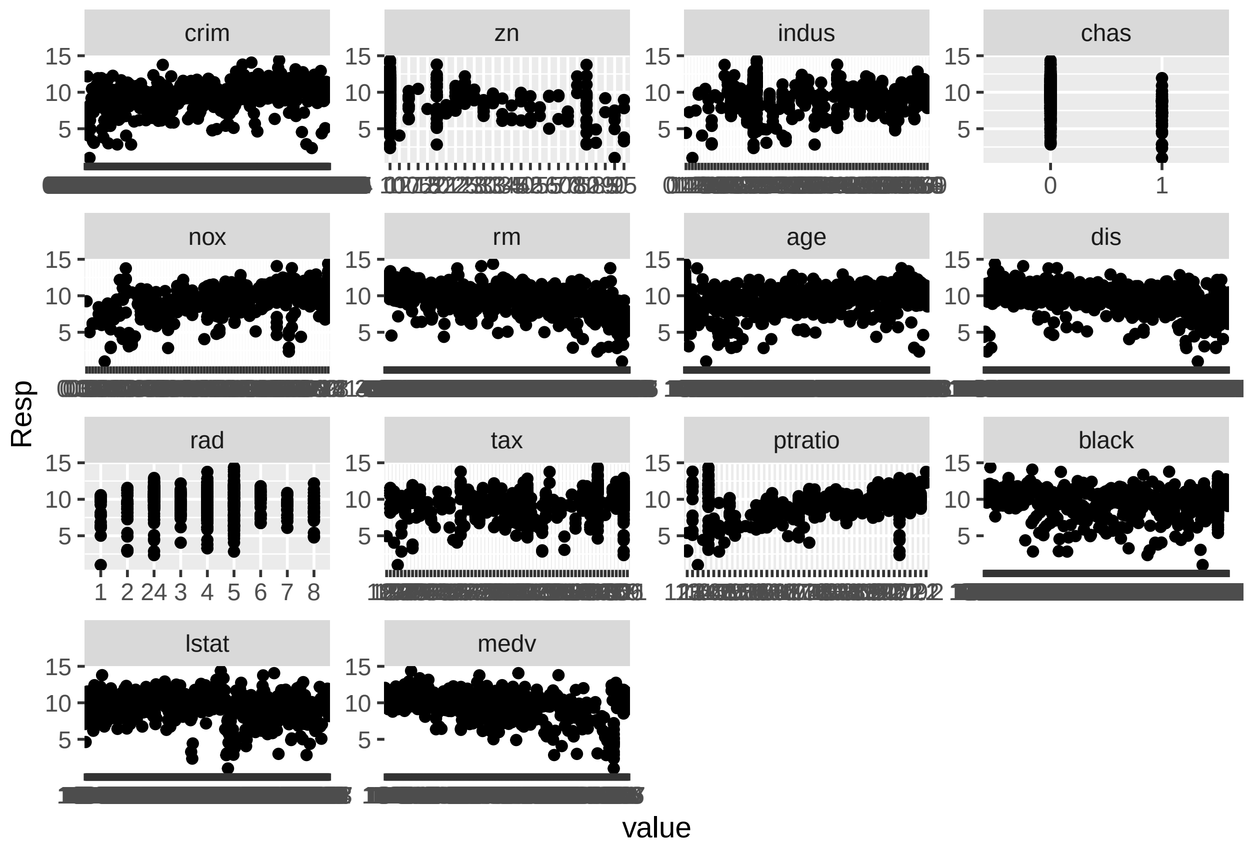
```
##    crim      zn   indus    chas     nox      rm     age     dis     rad     tax
##       0       0       0       0       0       0       0       0       0       0
## ptratio   black   lstat    medv    Resp
##       0       0       0       0       0
```

So we remove the NA values and double check our fixed data frame has no missing values.

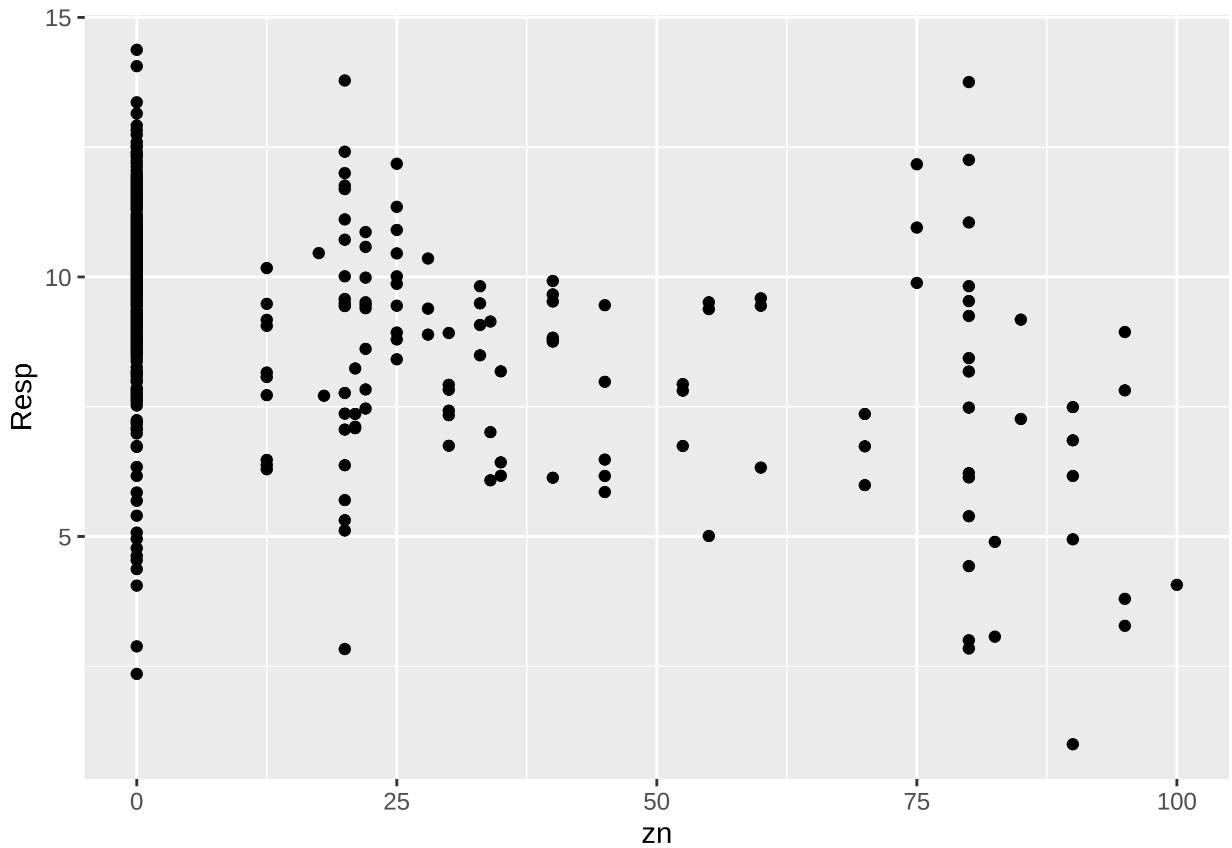
Exploratory Analysis

```
# plot each feature against Resp
ggplot(melt(df_no_na, id="Resp"), aes(x=value, y=Resp))+
  facet_wrap(~variable, scales="free")+
  geom_point()
```

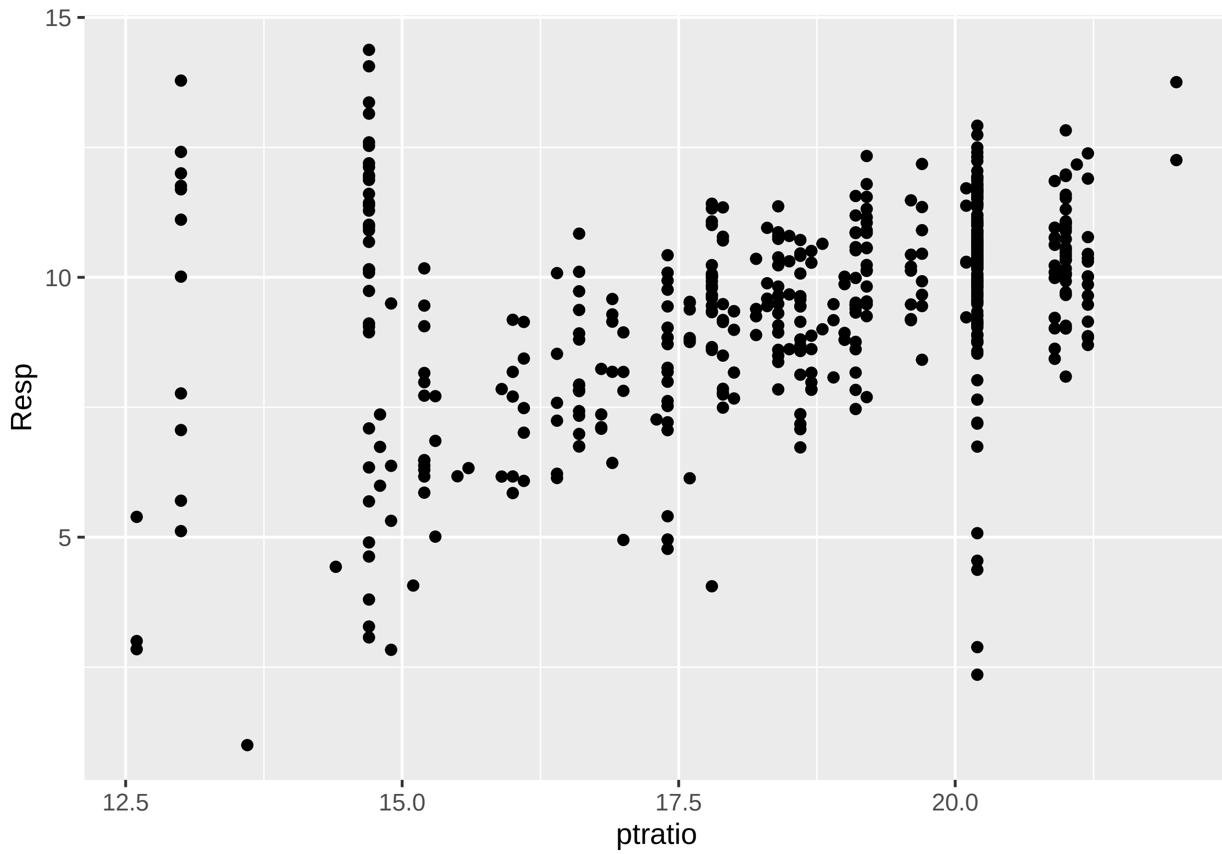
```
## Warning: attributes are not identical across measure variables; they will be
## dropped
```




```
#Investigating potential similarities between zn and ptratio's relationship to Resp. Found to be non si
ggplot(df_no_na, aes(x = zn, y = Resp)) +
  geom_point()
```

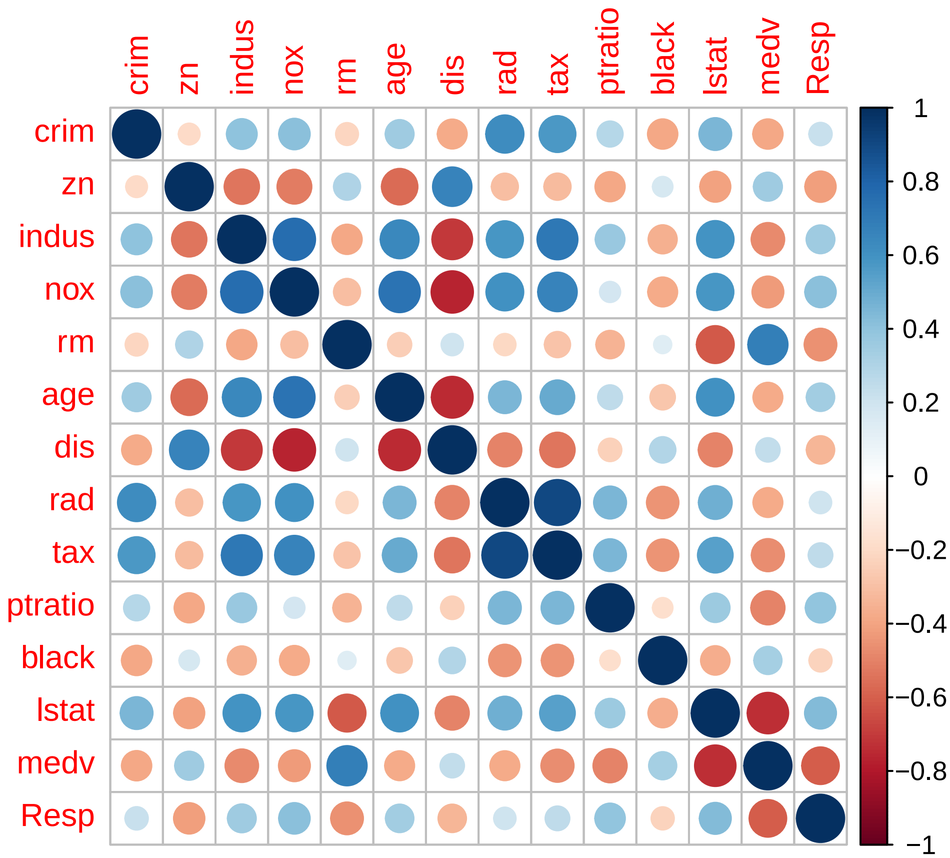



```r
ggplot(df_no_na, aes(x = ptratio, y = Resp)) +
  geom_point()
```



Correlation Analysis

```
corrplot(cor(df_no_na[,-c(4)]))
```




```r
(corrmatrix <- cor(df_no_na[,-c(4)], use = "complete.obs")[14,])
```

```
##       crim         zn      indus        nox         rm        age        dis
##  0.2276160 -0.4150077  0.3502033  0.4192069 -0.4538013  0.3417129 -0.3398125
##        rad        tax    ptratio      black      lstat       medv       Resp
##  0.2088079  0.2553058  0.3921175 -0.2237777  0.4313767 -0.6020060  1.0000000
```

```r
corrmatrix[corrmatrix > 0.5 | corrmatrix < -0.5]
```

```
##      medv      Resp
## -0.602006  1.000000
```

The variable 'medv' has the strongest correlation with a -0.6 implying that as the median value of the house decreases, the response variable increases.

Takeaways

Some takeaways so far:

- There were some NA values in our response variable, likely placed to be intentionally found by us. They seem to be randomly placed.
- There are no immediately obvious strong corollary effects between any variables and the response with the slight exception of medv.
- Additionally, there are no variables with no corollary effect with the response.
- The affect of the predictors on the response will be seen when we experiment with our models.
- dis, nox, indus, tax, rad, and age all present potential inter correlation concerns.

Section 2.2

Linear Models

Creating test and train data sets

```r
set.seed(123)
Boston.Stat4620<- na.omit(Boston.Stat4620)
ix <- sample(1:nrow(Boston.Stat4620),nrow(Boston.Stat4620)/2)
train <- Boston.Stat4620[ix, ]
test<- Boston.Stat4620[-ix, ]
test_mses <- c()
```

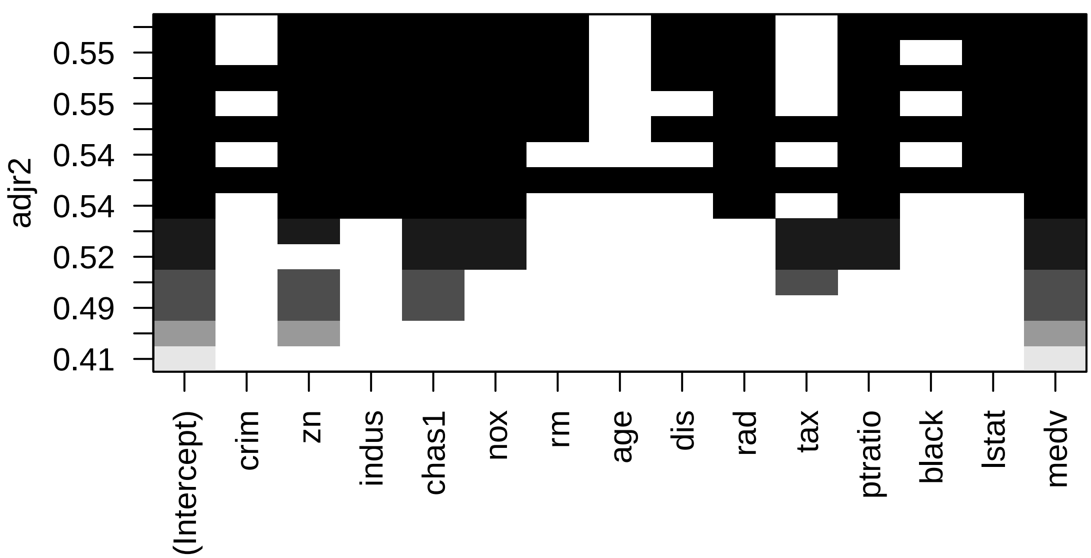
Using best subset selection to fit data using least squares

```r
set.seed(123)
x.test <- model.matrix(Resp ~., test)[,-1]
y.test<- test$Resp
regfit<- regsubsets(Resp~ . , train, nvmax = 14)
reg.summary <- summary(regfit)
reg.summary
```

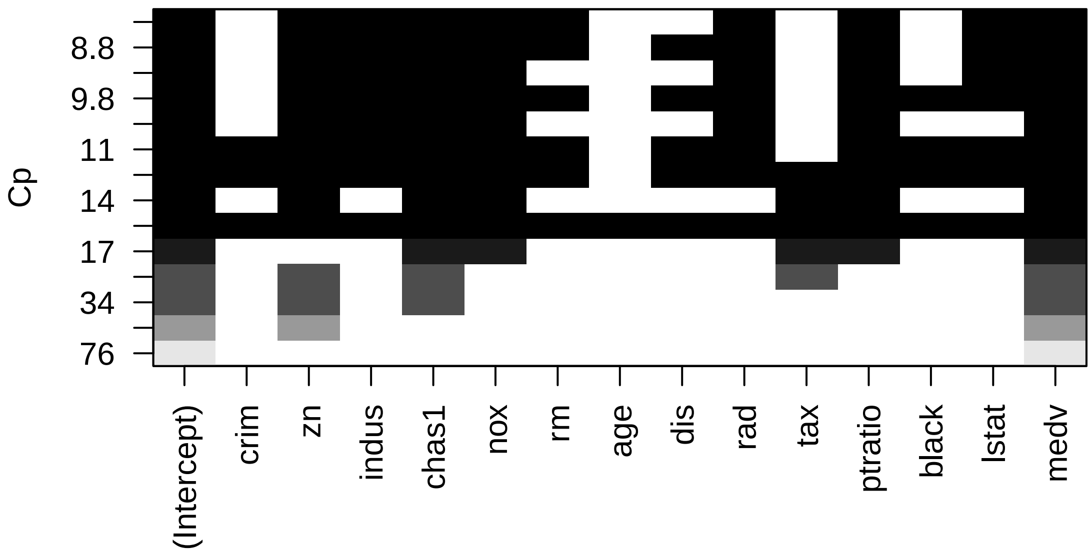
```
## Subset selection object
## Call: regsubsets.formula(Resp ~ ., train, nvmax = 14)
## 14 Variables  (and intercept)
##         Forced in Forced out
## crim        FALSE      FALSE
## zn          FALSE      FALSE
## indus       FALSE      FALSE
## chas1       FALSE      FALSE
## nox         FALSE      FALSE
## rm          FALSE      FALSE
## age         FALSE      FALSE
## dis         FALSE      FALSE
## rad         FALSE      FALSE
## tax         FALSE      FALSE
## ptratio     FALSE      FALSE
## black       FALSE      FALSE
## lstat       FALSE      FALSE
## medv        FALSE      FALSE
## 1 subsets of each size up to 14
## Selection Algorithm: exhaustive
##           crim zn  indus chas1 nox rm  age dis rad tax ptratio black lstat medv
## 1  ( 1 )  " "  " " " "   " "   " " " " " " " " " " " " " "     " "   " "   "*" 
## 2  ( 1 )  " "  "*" " "   " "   " " " " " " " " " " " " " "     " "   " "   "*" 
## 3  ( 1 )  " "  "*" " "   "*"   " " " " " " " " " " " " " "     " "   " "   "*" 
## 4  ( 1 )  " "  "*" " "   "*"   " " " " " " " " " " "*" " "     " "   " "   "*" 
## 5  ( 1 )  " "  " " " "   "*"   "*" " " " " " " " " "*" "*"     " "   " "   "*" 
## 6  ( 1 )  " "  "*" " "   "*"   "*" " " " " " " " " "*" "*"     " "   " "   "*" 
## 7  ( 1 )  " "  "*" "*"   "*"   "*" " " " " " " "*" " " "*"     " "   " "   "*" 
```

```
## 8  ( 1 )  " "  "*" "*"   "*"   "*" " " " " " " "*" " " "*"     " "    "*"   "*"
## 9  ( 1 )  " "  "*" "*"   "*"   "*" "*" " " " " "*" " " "*"     " "    "*"   "*"
## 10  ( 1 ) " "  "*" "*"   "*"   "*" "*" " " "*" "*" " " "*"     " "    "*"   "*"
## 11  ( 1 ) " "  "*" "*"   "*"   "*" "*" " " "*" "*" " " "*"     "*"    "*"   "*"
## 12  ( 1 ) "*"  "*" "*"   "*"   "*" "*" " " "*" "*" " " "*"     "*"    "*"   "*"
## 13  ( 1 ) "*"  "*" "*"   "*"   "*" "*" " " "*" "*" "*" "*"     "*"    "*"   "*"
## 14  ( 1 ) "*"  "*" "*"   "*"   "*" "*" "*" "*" "*" "*" "*"     "*"    "*"   "*"
```

```r
plot(regfit, scale = "adjr2")
```




```r
plot(regfit, scale = "Cp")
```




```r
test.mat <- model.matrix(Resp~ ., data = test)
val.err <- rep(NA, 14)
for(i in 1:14){
  coefi <- coef(regfit, id =i)
  pred <- test.mat[, names(coefi)] %*% coefi
    val.err[i]<- mean((test$Resp- pred )^2)
}
val.err
```

```
##  [1] 2.372786 2.197629 2.010952 2.034192 1.727986 1.725316 1.803466 1.795624
##  [9] 1.792285 1.775017 1.773710 1.782396 1.758189 1.756358
```

```r
which.min(val.err)
```

```
## [1] 6
```

```r
coef(regfit,6)
```

```
##  (Intercept)           zn        chas1          nox          tax      ptratio
##  7.265979027 -0.009863672 -1.790245548  5.613405527 -0.003893106  0.171698560
##         medv
## -0.101323276
```

```r
best.reg<- lm(Resp~zn+chas+nox+tax+ptratio+medv, data = train)
bestreg.sum <- summary(best.reg)
reg.pred <- predict(best.reg, test)
#MSE Least Squares
best.reg.mse <- mean((reg.pred-y.test)^2)
test_mses <- c(test_mses, best.reg.mse )
```

The plot shows that the model containing the 6 variables Zn, Chas, Nox, Tax, Ptratio, and medv results in the lowest Cp and the coefficients for this model are as follows:

```r
coef(best.reg, 7)
```

```
##  (Intercept)           zn        chas1          nox          tax      ptratio
##  7.265979027 -0.009863672 -1.790245548  5.613405527 -0.003893106  0.171698560
##         medv
## -0.101323276
```

Ridge Regression

```r
set.seed(123)
x <- model.matrix(Resp ~ . ,Boston.Stat4620) [, -1]
y <- Boston.Stat4620$Resp
grid <- 10^seq(10,-2, length=100)
ridge.mod <- glmnet(x[ix, ],y[ix], alpha = 0, lambda = grid)
#finding which is best lambda value
cv.ridgeglm <- cv.glmnet(x[ix, ], y[ix], alpha=0)
best.lam<- cv.ridgeglm$lambda.min
ridge.pred <- predict(ridge.mod, s =0.1363408, newx = x.test)
#MSE Ridge
ridge.mse <- mean((ridge.pred - y[-ix])^2)
test_mses <- c(test_mses, ridge.mse )
```

Lasso Regression

```r
x.train <- model.matrix(Resp~ ., train)[,-1]
y.train <- train$Resp
lasso.cv <- cv.glmnet(x.train, y.train, alpha=1)
lambda.cv <- lasso.cv$lambda.min
lasso.mod <- glmnet(x.train, y.train, alpha = 1, lambda = lambda.cv)
lasso.pred <- predict(lasso.mod, newx=x.test)
#MSE LASSO
lasso.mse <- mean((lasso.pred-y.test)^2)
test_mses <- c(test_mses, lasso.mse )
```

```r
Boston <- Boston.Stat4620
Boston <- subset(Boston, select=-12)
Boston <- na.omit(Boston)
set.seed(1738)
```

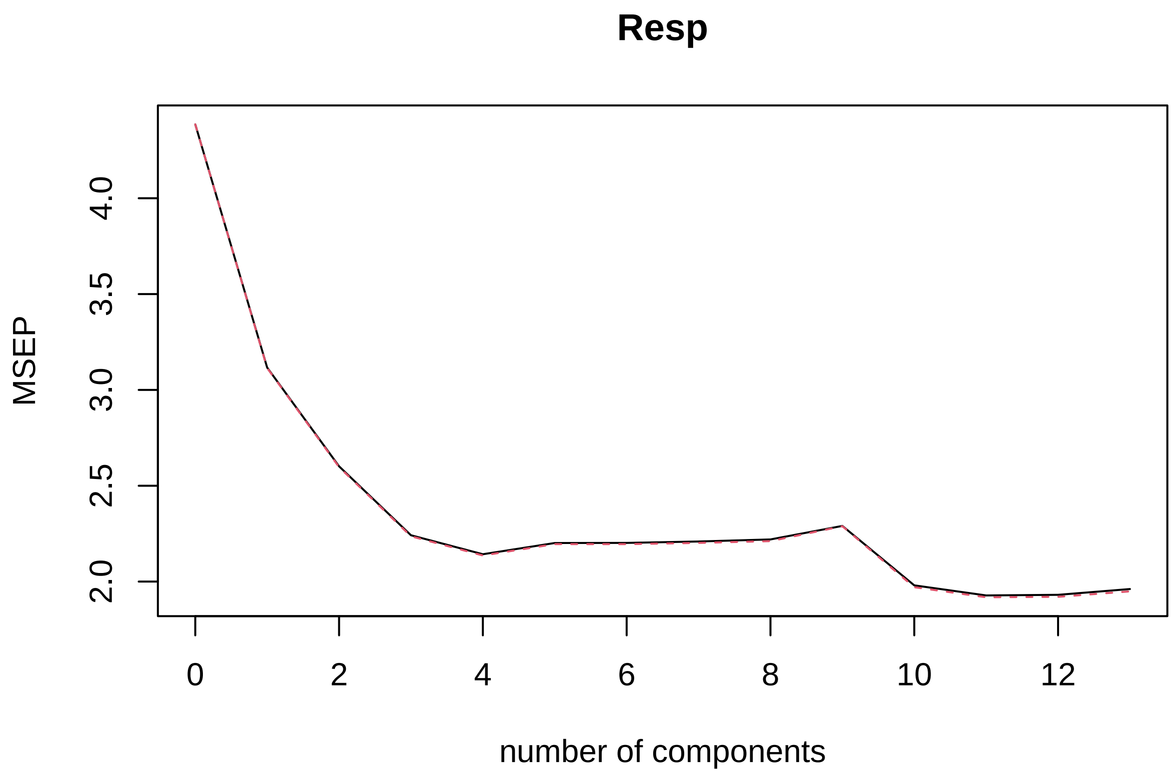
```r
ix <- sample(1:nrow(Boston), nrow(Boston)/2)
Boston_train <- Boston[ix,]
Boston_test <- Boston[-ix,]
```

PCR for Resp

```r
library(pls)
set.seed(1738)
Boston.pcr <- pcr(Resp~., data=Boston_train, scale=T, validation="CV")
summary(Boston.pcr)
```

```
## Data:    X dimension: 248 13
##  Y dimension: 248 1
## Fit method: svdpc
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           2.094    1.765    1.613    1.497    1.464    1.484    1.484
## adjCV        2.094    1.765    1.612    1.495    1.462    1.482    1.482
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       1.486    1.490    1.513     1.407     1.388     1.390     1.400
## adjCV    1.484    1.487    1.513     1.404     1.385     1.386     1.396
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       49.73    62.92    72.60    79.21    84.90    89.06    91.93    94.00
## Resp    29.21    42.17    50.09    51.86    51.87    52.09    52.54    52.66
##       9 comps  10 comps  11 comps  12 comps  13 comps
## X       95.91     97.51     98.73     99.49    100.00
## Resp    52.91     58.04     59.55     59.95     59.95
```

```r
validationplot(Boston.pcr, val.type="MSEP")
```




```
pcr.pred <- predict(Boston.pcr, Boston_test, ncomp=4)
pcr.mse <- mean((pcr.pred-Boston_test$Resp)^2)
test_mses <- c(test_mses, mean((pcr.pred-Boston_test$Resp)^2))
```

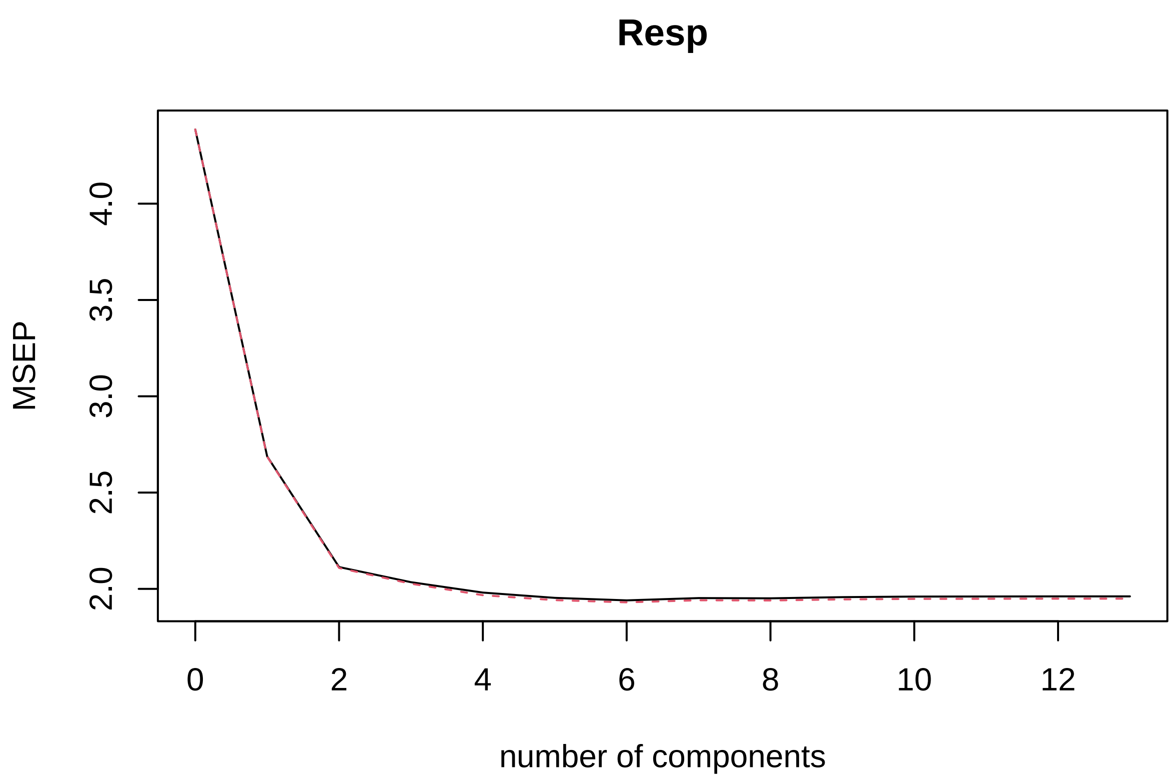
PLS for Resp

```
set.seed(1738)
Boston.pls <- plsr(Resp~., data=Boston_train, scale=T, validation="CV")
summary(Boston.pls)
```

```
## Data:    X dimension: 248 13
##  Y dimension: 248 1
## Fit method: kernelpls
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           2.094    1.640    1.454    1.426    1.407    1.398    1.393
## adjCV        2.094    1.639    1.452    1.424    1.403    1.393    1.389
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       1.397    1.397    1.399     1.400     1.400     1.400     1.400
## adjCV    1.393    1.393    1.395     1.396     1.396     1.396     1.396
```

```
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       48.19    61.61    69.53    74.49    80.44    83.70    86.20    88.69
## Resp    39.51    53.89    57.01    59.44    59.82    59.91    59.94    59.95
##       9 comps  10 comps  11 comps  12 comps  13 comps
## X       91.08     95.58     97.51     99.04    100.00
## Resp    59.95     59.95     59.95     59.95     59.95
```

```
validationplot(Boston.pls, val.type="MSEP")
```




```
pls.pred <- predict(Boston.pls, Boston_test, ncomp=5)
pls.mse<- mean((pls.pred-Boston_test$Resp)^2)
test_mses <- c(test_mses, mean((pls.pred-Boston_test$Resp)^2))
```

Non Linear Models

Bagging for Resp

```
set.seed(1738)
Boston.bag <- randomForest(Resp~., data=Boston_train, mtry=13, importance=T, ntree=100)
bag.pred <- predict(Boston.bag, newdata=Boston_test)
test_mses <- c(test_mses, mean((bag.pred-Boston_test$Resp)^2))
```

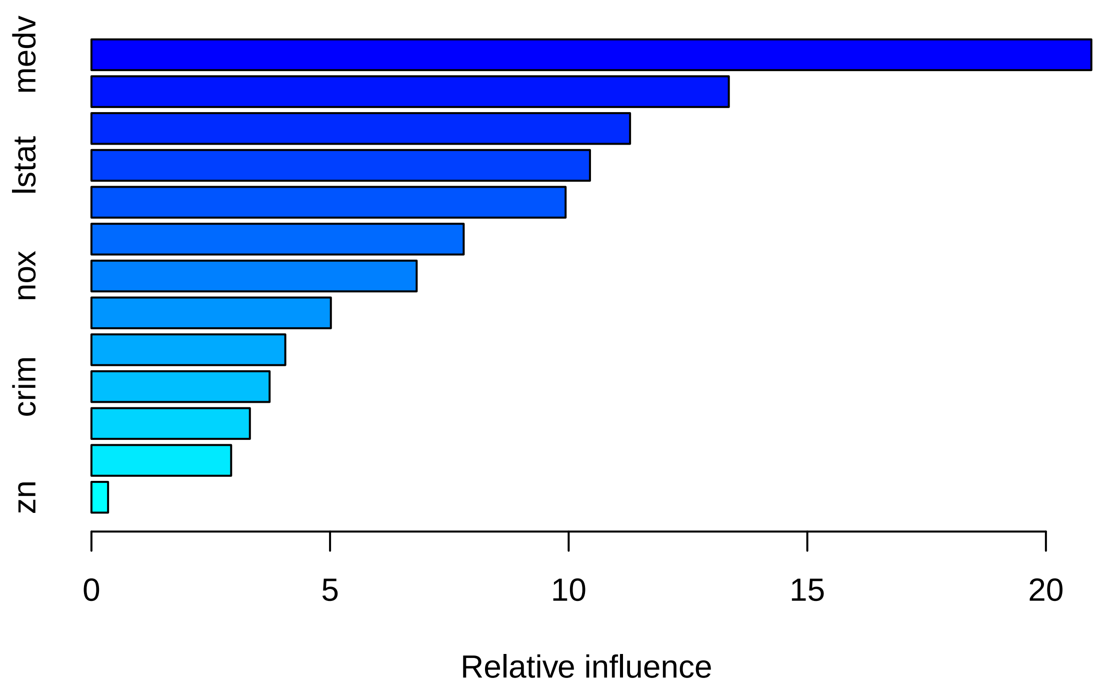
```
bag.mse <- mean(bag.pred-Boston_test$Resp)^2
names(bag.mse) <- "Bagging Model MSE"
bag.importance <- importance(Boston.bag)
```

Random Forest for Resp

```
set.seed(1738)
Boston.rf <- randomForest(Resp~., data=Boston_train, importance=T, ntree=100)
rf.pred <- predict(Boston.rf, newdata=Boston_test)
test_mses <- c(test_mses, mean((rf.pred-Boston_test$Resp)^2))
rf.mse <- mean(rf.pred-Boston_test$Resp)^2
names(rf.mse) <- "Random Forest Model MSE"
rf.importance <- importance(Boston.rf)
```

Boosting for Resp

```
set.seed(1738)
Boston.boost <- gbm(Resp~., data=Boston_train, distribution="gaussian", n.trees=500, interaction.depth=
summary(Boston.boost)
```




```
##              var    rel.inf
```

```
## medv       medv 20.9522827
## dis         dis 13.3557231
## rm           rm 11.2859462
## lstat     lstat 10.4461444
## ptratio ptratio  9.9357292
## age         age  7.7999080
## nox         nox  6.8148607
## indus     indus  5.0172470
## tax         tax  4.0625375
## crim       crim  3.7331887
## chas       chas  3.3204155
## rad         rad  2.9271954
## zn           zn  0.3488216
```

```
boost.pred <- predict(Boston.boost, newdata=Boston_test)
```

```
## Using 500 trees...
```

```
test_mses <- c(test_mses, mean((boost.pred-Boston_test$Resp)^2))

boost.mse <- mean((boost.pred-Boston_test$Resp)^2)
names(boost.mse) <- "Boosting Model MSE"
```

```
names(test_mses) <- c("Least Squares","Ridge","Lasso", "PCR", "PLS", "Bagging", "Random Forest", "Boosti
test_mses
```

```
## Least Squares         Ridge         Lasso           PCR           PLS 
##      1.725316      1.733120      1.748026      2.353810      1.988015 
##       Bagging Random Forest      Boosting 
##      1.543723      1.616818      1.770873
```

Early Observations:

As we can see, from our models so far Bagging gives the lowest test MSE. However, we can also see that the variable medv has high importance in all of our CART methods. I will see if just using this variable in splines could be a better predictor of the mystery response.

Cubic Splines With medv

```
Boston.cubic <- lm(Resp~bs(medv, df=4), data=Boston_train)
cubic.pred <- predict(Boston.cubic, newdata=data.frame(Boston_test))
```

```
## Warning in bs(medv, degree = 3L, knots = 21.2, Boundary.knots = c(5.6, 50: some
## 'x' values beyond boundary knots may cause ill-conditioned bases
```

```
mean((cubic.pred-Boston_test$Resp)^2)
```

```
## [1] 2.455012
```

```r
summary(Boston.cubic)
```

```
##
## Call:
## lm(formula = Resp ~ bs(medv, df = 4), data = Boston_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8211 -0.8747  0.0665  0.9510  4.8840
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          10.3024     0.7349  14.020  < 2e-16 ***
## bs(medv, df = 4)1     1.4605     1.1223   1.301  0.19438
## bs(medv, df = 4)2    -2.6391     0.8351  -3.160  0.00178 **
## bs(medv, df = 4)3    -1.2412     1.2673  -0.979  0.32833
## bs(medv, df = 4)4    -6.0839     0.8542  -7.123 1.19e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.566 on 243 degrees of freedom
## Multiple R-squared:  0.4475, Adjusted R-squared:  0.4384
## F-statistic:  49.2 on 4 and 243 DF,  p-value: < 2.2e-16
```

```r
cs.mse <- mean((cubic.pred-Boston_test$Resp)^2)
names(cs.mse) = "Cubic Splines MSE"
```

Natural Splines With medv

```r
Boston.natural <- lm(Resp~ns(medv, df=4), data=Boston_train)
natural.pred <- predict(Boston.natural, newdata=data.frame(Boston_test))
mean((natural.pred-Boston_test$Resp)^2)
```

```
## [1] 2.435072
```

```r
summary(Boston.natural)
```

```
##
## Call:
## lm(formula = Resp ~ ns(medv, df = 4), data = Boston_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8868 -0.8345  0.0585  0.9812  4.7806
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          10.6387     0.5976  17.804  < 2e-16 ***
## ns(medv, df = 4)1    -1.2443     0.5886  -2.114  0.03555 *
```

```
## ns(medv, df = 4)2  -1.7306     0.5638  -3.069  0.00239 **
## ns(medv, df = 4)3  -3.6187     1.4285  -2.533  0.01193 *
## ns(medv, df = 4)4  -6.2153     0.5194 -11.967  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.581 on 243 degrees of freedom
## Multiple R-squared:  0.4373, Adjusted R-squared:  0.428
## F-statistic: 47.21 on 4 and 243 DF,  p-value: < 2.2e-16
```

```r
nat.mse <- mean((natural.pred-Boston_test$Resp)^2)
names(nat.mse) = "Natural Splines MSE"
```

Smoothing Spline With medv

```r
set.seed(1738)
Boston.smoothcv <- smooth.spline(Boston_train$medv, Boston_train$Resp, cv=T)
```

```
## Warning in smooth.spline(Boston_train$medv, Boston_train$Resp, cv = T):
## cross-validation with non-unique 'x' values seems doubtful
```

```r
smooth.pred <- predict(Boston.smoothcv, Boston_test$medv)
mean((smooth.pred$y-Boston_test$Resp)^2)
```

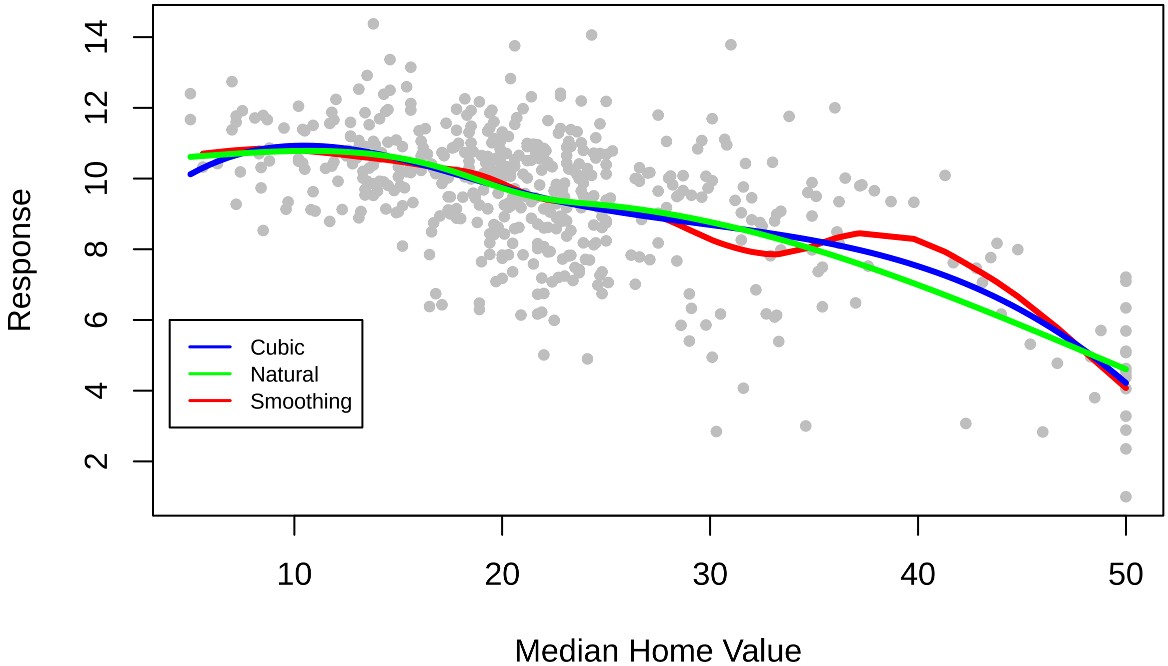
```
## [1] 2.51081
```

```r
smooth.mse <- mean((smooth.pred$y-Boston_test$Resp)^2)
names(smooth.mse) = "Smoothing Splines MSE"
```

```r
medv.grid <- seq(min(Boston$medv),max(Boston$medv),length=100)
pred.cs <- predict(Boston.cubic, newdata=data.frame(medv=medv.grid), se=T)
```

```
## Warning in bs(medv, degree = 3L, knots = 21.2, Boundary.knots = c(5.6, 50: some
## 'x' values beyond boundary knots may cause ill-conditioned bases
```

```r
pred.ns <- predict(Boston.natural, newdata=data.frame(medv=medv.grid), se=T)
plot(Boston$medv,Boston$Resp,pch=20,xlab="Median Home Value",ylab="Response", col="grey")
lines(Boston.smoothcv,col="red",lwd=3)
lines(medv.grid, pred.cs$fit,col="blue",lwd=3)
lines(medv.grid, pred.ns$fit,col="green",lwd=3)
legend(4, 6, c("Cubic","Natural", "Smoothing"), lwd=c(1.5,1.5, 1.5), col=c("blue","green", "red"), cex=(
```



There does not seem to be evidence to suggest that the splines with the predictor medv improve our model over bagging, as all three models have much higher test MSEs.