Data 3 HW 1

Sean Duan

8/31/2020

1.

The advantages of a flexible approach is that you are more able to adapt to potential nonlinearity in your data, additionally, as you increase your flexibilty, your training MSE continues to improve. A more flexible approach would be preferred if you are unable to make parametric assumptions of your dataset, and are able to accept higher variance in exchange for potentially lower bias. A less flexible approach would be preferred if you believe your data to fall neatly in line as a linear model, and are confident in making the parametric assumption with regards to the form of your model, additionally, if you are concerned with potentially overfitting your data, a less flexible approach would be ideal.

2.

A parametric statistical learning approach allows you to simplify the process of determining what your model, f, is. By allowing you to assume a given form for the function, the only remaining thing to estimate are the values for the parameters of that given form. Concerns with the parametric approach arise when you consider it may be unlikely that the form you assume for f does not indeed fit the true, unknown, form for f that exists. A non-parametric statistical learning approach makes no assumptions about the form of f, which allows f to be in a broader variety of forms, hopefully leading it to match the true unknown form of the function f. The advantages and disadvantages of choosing a parametric vs a non parametric approach are as follows: With a parametric approach, you need a great deal less observations to estimate f, as you are reducing the problem of estimation of f to a simple concern of finding the values of a small number of parameters, however, your disadvantage is that you potentially do not have an acceptable quality of estimation of the true functional form of f, as you could have with a non-parametric approach.

3

A

```
#a.
seeds<-read.csv("seeds.csv")
seeds$Type<-as.factor(seeds$Type)

seeds$Type<-revalue(seeds$Type, c("1" = "Kama", "2" = "Rosa", "3" = "Canadian"))

levels(seeds$Type)
```

```
## [1] "Kama"     "Rosa"     "Canadian"
```

B

```
#b.
summary(seeds)
```

```
##       Area         perimeter      compactness     length.of.kernel
##  Min.   :10.59   Min.   :12.41   Min.   :0.8081   Min.   :4.899
##  1st Qu.:12.27   1st Qu.:13.45   1st Qu.:0.8569   1st Qu.:5.262
##  Median :14.36   Median :14.32   Median :0.8734   Median :5.524
##  Mean   :14.85   Mean   :14.56   Mean   :0.8710   Mean   :5.629
##  3rd Qu.:17.30   3rd Qu.:15.71   3rd Qu.:0.8878   3rd Qu.:5.980
##  Max.   :21.18   Max.   :17.25   Max.   :0.9183   Max.   :6.675
##  width.of.kernel asymmetry.coefficient length.of.kernel.groove       Type
##  Min.   :2.630   Min.   :0.7651        Min.   :4.519           Kama    :70
##  1st Qu.:2.944   1st Qu.:2.5615        1st Qu.:5.045           Rosa    :70
##  Median :3.237   Median :3.5990        Median :5.223           Canadian:70
##  Mean   :3.259   Mean   :3.7002        Mean   :5.408
##  3rd Qu.:3.562   3rd Qu.:4.7687        3rd Qu.:5.877
##  Max.   :4.033   Max.   :8.4560        Max.   :6.550
```

C

```
count(seeds$perimeter > 15)
```

```
##       x freq
## 1 FALSE  139
## 2  TRUE   71
```

```
71/210
```

```
## [1] 0.3380952
```

D

```
which.max(seeds$asymmetry.coefficient)
```

```
## [1] 204
```

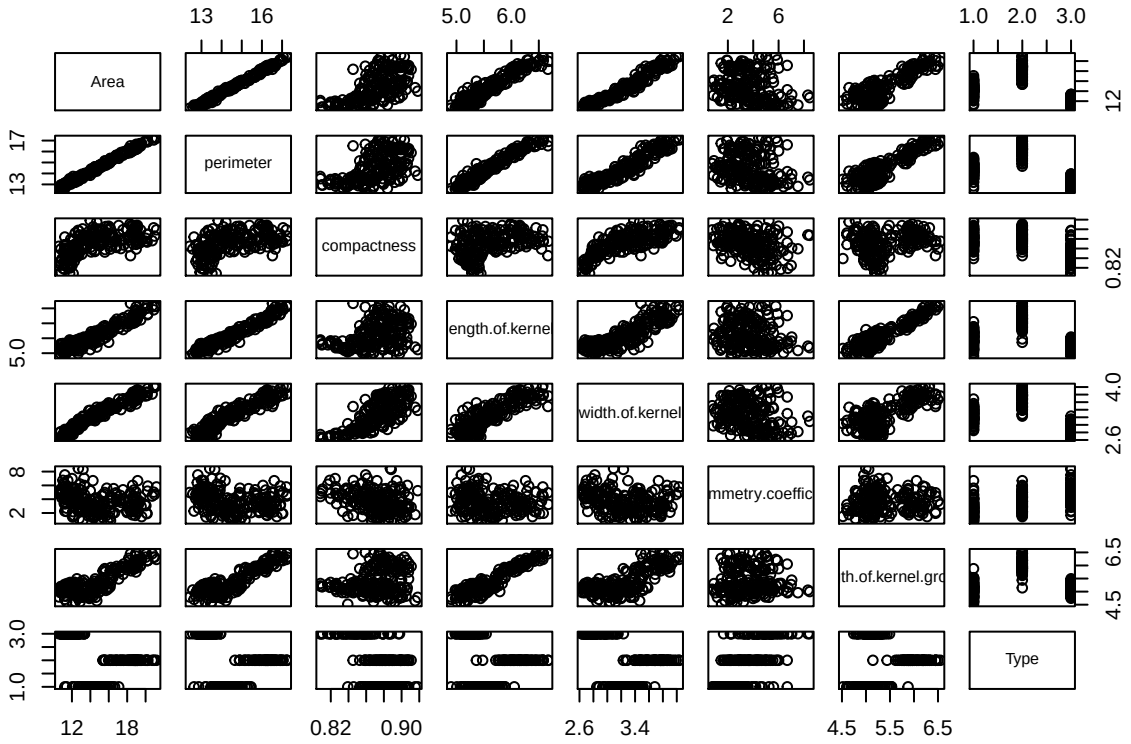
```
seeds[which.max(seeds$asymmetry.coefficient),]
```

```
##     Area perimeter compactness length.of.kernel width.of.kernel
## 204 12.7     13.41      0.8874            5.183           3.091
##     asymmetry.coefficient length.of.kernel.groove     Type
## 204                 8.456                       5 Canadian
```

Entry #204, and the type is Canadian

E

```
pairs(seeds)
```

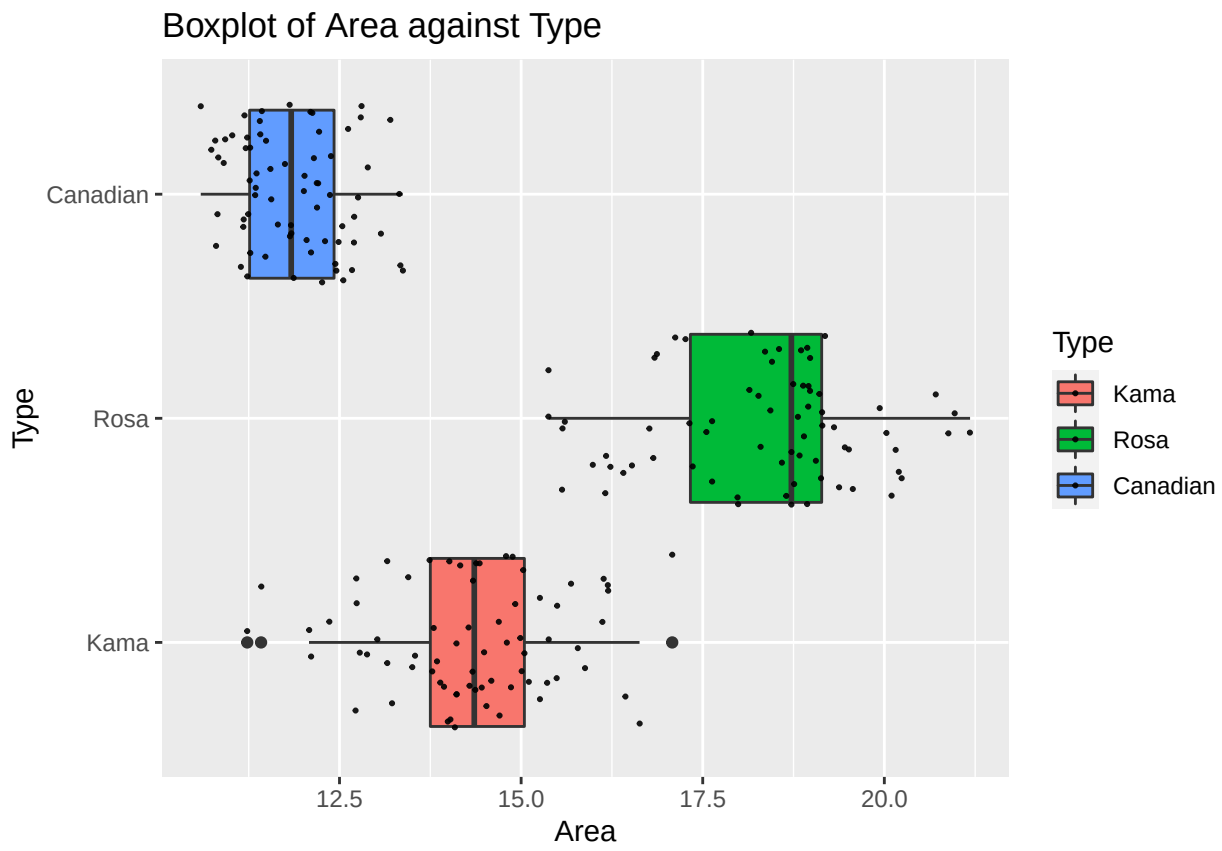


Looking at the scatterplot matrix, it looks like the most highly related factors are area and perimeter, area and length of kernel, area and width of kernel, perimeter and length of kernel, perimeter and width of kernel, and length of kernel groove and length of kernel.

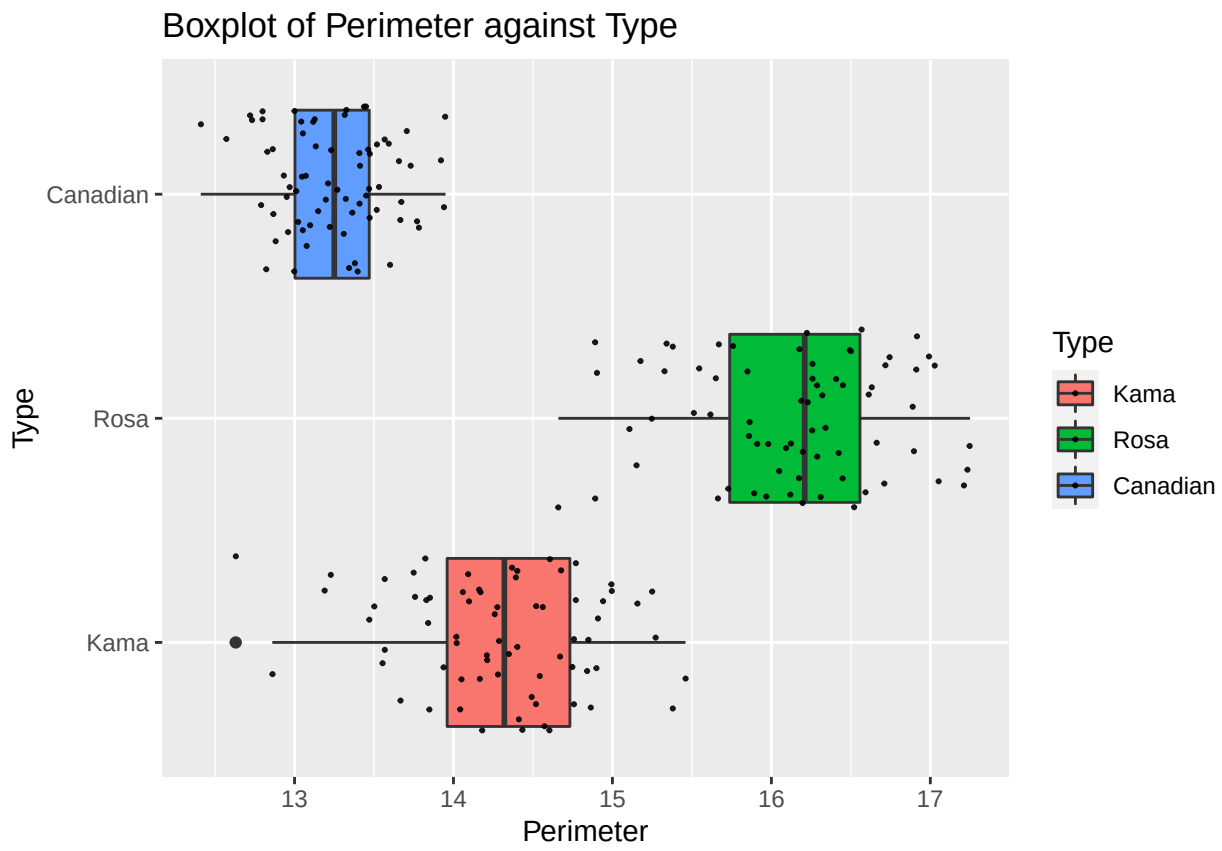
It looks like area, perimeter, and length of kernel groove are the best at distinguishing type.

F

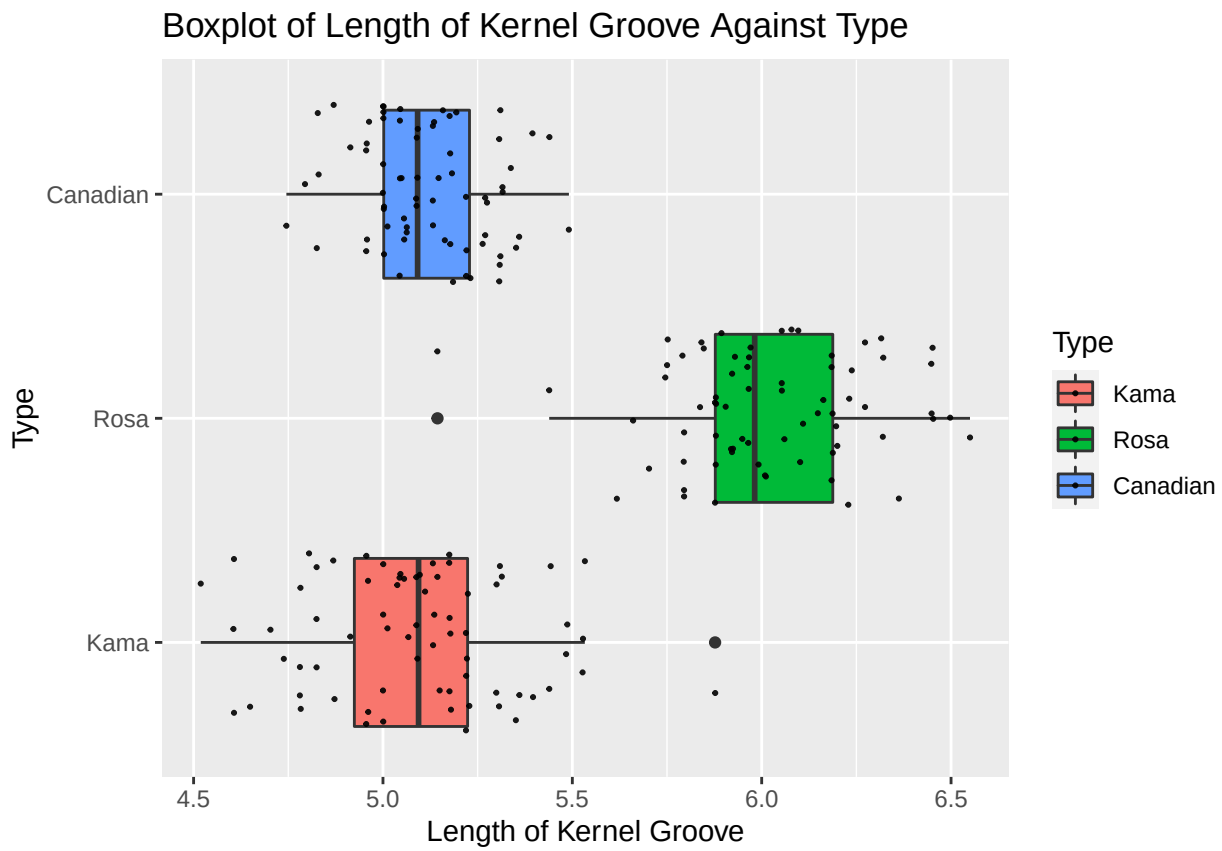
```
p3_f1<-ggplot(data =seeds, aes(x=Area, y=Type, fill=Type ))
p3_f1 + geom_boxplot() + geom_jitter(color = "black", size = 0.4, alpha = 0.9)+
  ggtitle("Boxplot of Area against Type") + xlab("Area") + ylab("Type")
```




```
p3_f2<-ggplot(data =seeds, aes(x=perimeter, y=Type, fill=Type ))
p3_f2 + geom_boxplot() + geom_jitter(color = "black", size = 0.4, alpha = 0.9)+
  ggtitle("Boxplot of Perimeter against Type") + xlab("Perimeter") + ylab("Type")
```



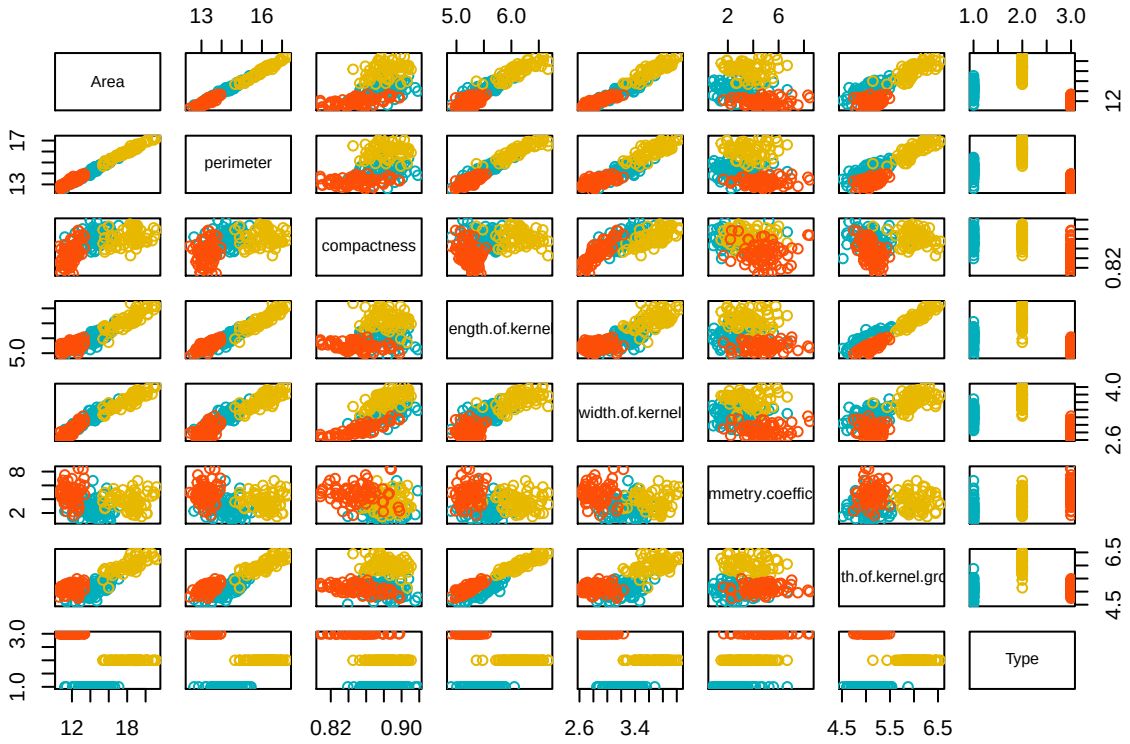

```
p3_f3<-ggplot(data =seeds, aes(x=length.of.kernel.groove, y=Type, fill=Type ))
p3_f3 + geom_boxplot() + geom_jitter(color = "black", size = 0.4, alpha = 0.9)+
  ggtitle("Boxplot of Length of Kernel Groove Against Type") + xlab("Length of Kernel Groove") + ylab("
```



Looking at these plots I would like to look at Area and Perimeter as our predictors variables in a model for Kernal type based on these plots. The reasonining behind this is when you look at the graph, there is clear separation between all three groups on both the variables examined. Following a clear trend of Canadian being smaller than Kama, which is smaller than Rosa.

G

```
my_cols <- c("#00AFBB", "#E7B800", "#FC4E07")
pairs(seeds, col = my_cols[seeds$Type])
```



4

$$\hat{y_a} = x_a \hat{\beta}$$

$$= x_a \frac{\sum_{b=1}^{n} x_b y_b}{\sum_{c=1}^{n} x_c^2}$$

We can replace our $\hat{\beta}$ with our given information. Also note that each of our variables which we are choosing to index has it's own index variable, either a, b, or c.

$$= \frac{\sum_{b=1}^{n} x_a x_b y_b}{\sum_{c=1}^{n} x_c^2}$$

Since x_a isn't indexed to b, we can pull it into the numerator for our term

$$= \sum_{b=1}^{n} \frac{x_a x_b y_b}{\sum_{c=1}^{n} x_c^2}$$

Likewise, since our denominator x_c^2 isn't indexed to b, we can pull our summation sign out to the left

$$= \sum_{b=1}^{n} \frac{x_a x_b}{\sum_{c=1}^{n} x_c^2} y_b$$

Since it is all within our summation term indexing b, we can pull our y_b to the right, which is what we want for our final solution

$$= \sum_{b=1}^{n} a_b y_b$$

Lastly, we redefine all elements other than y_b and our index of summation out as a new item, a_b

Where we define a_b as

$$a_b = \frac{x_a x_b}{\sum_{c=1}^{n} x_c^2}$$

5

A

```
data(Boston)
Boston_2<-Boston^2
Boston_3<-Boston^3
Boston$chas <- factor(Boston$chas)
#code for each single regression, plus plots of each
b_m1<-lm(crim~zn, data = Boston)
#summary(b_m1)
b_m1p<-ggplot(Boston, aes(x = zn, y = crim)) +
  geom_point() +
  geom_abline(intercept = coef(b_m1)[1], slope = coef(b_m1)[2],
              col = "deepskyblue3",
              size = 1) +
  geom_smooth(se = F)
b_m1p
```

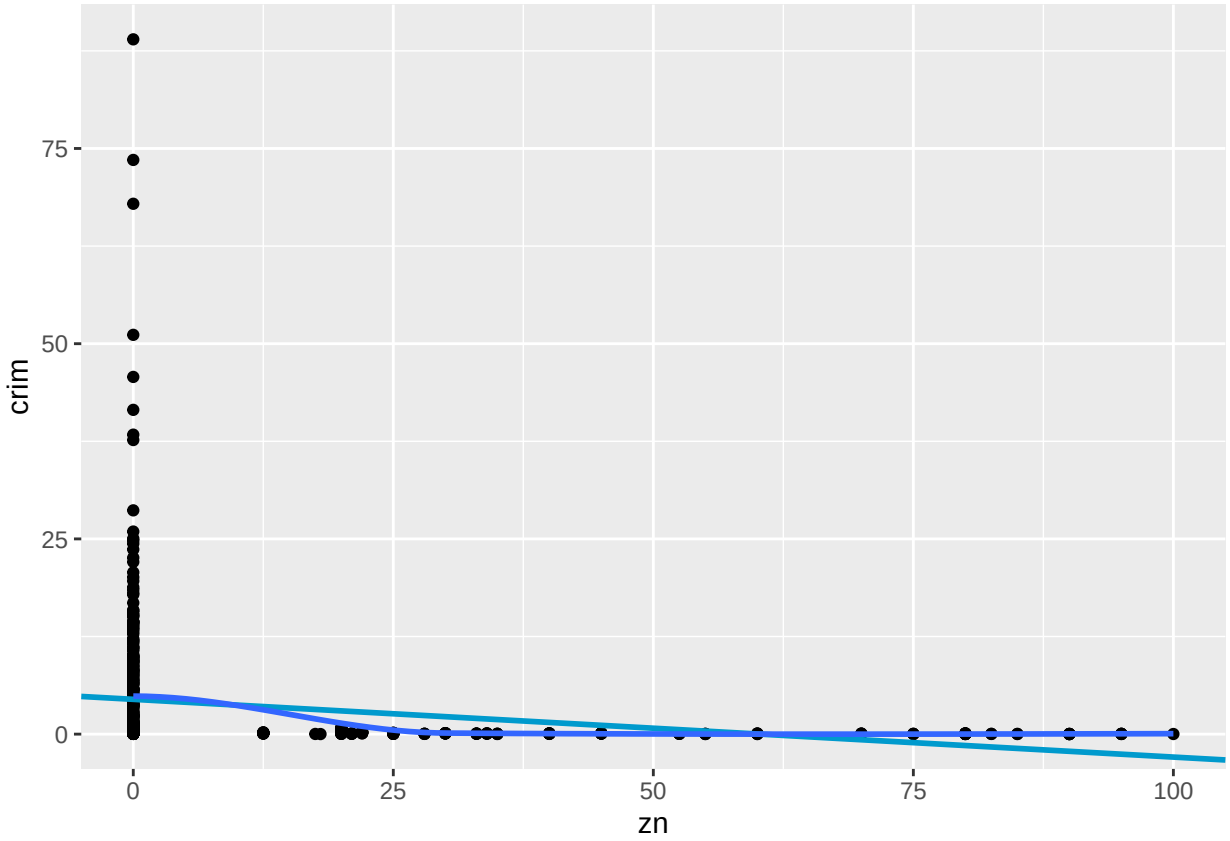
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -0.5
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 13
```

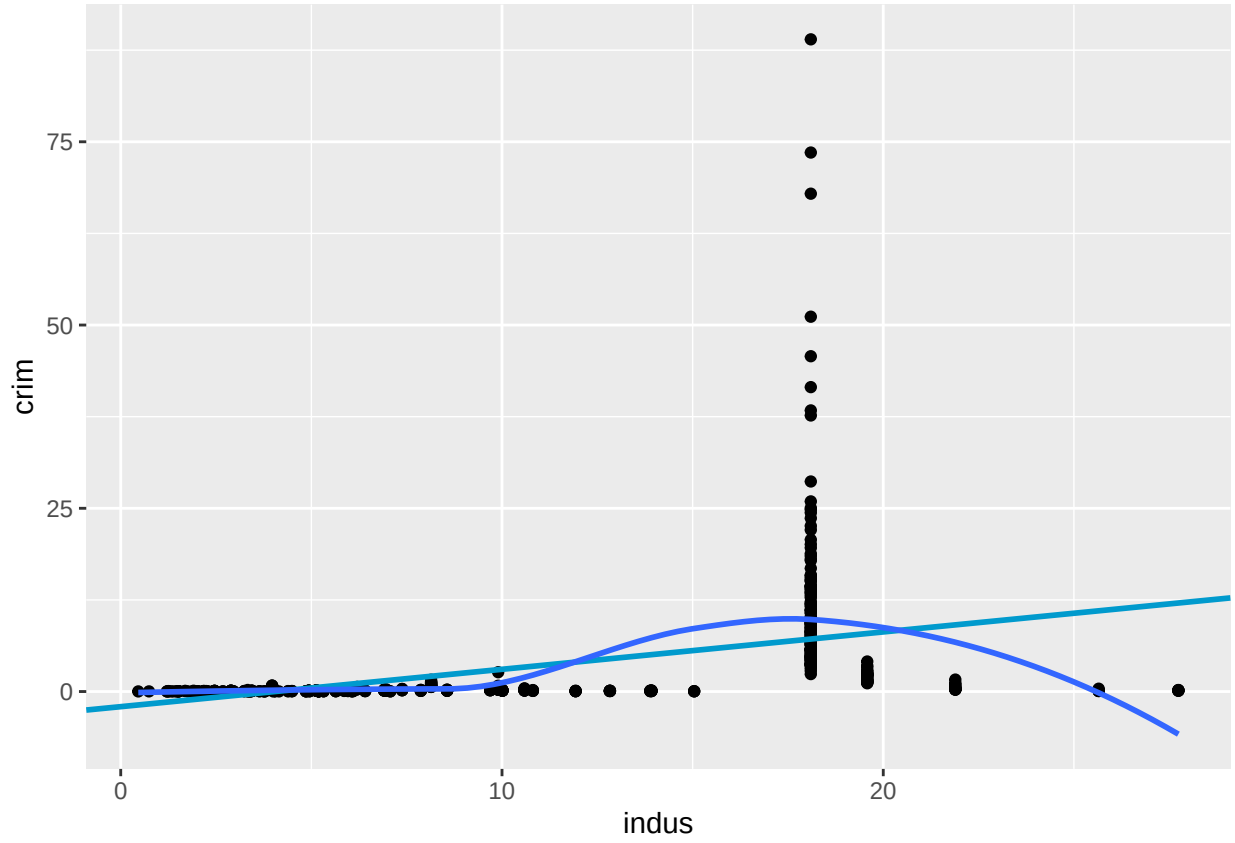
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 2.9038e-031
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : There are other near singularities as well. 156.25
```



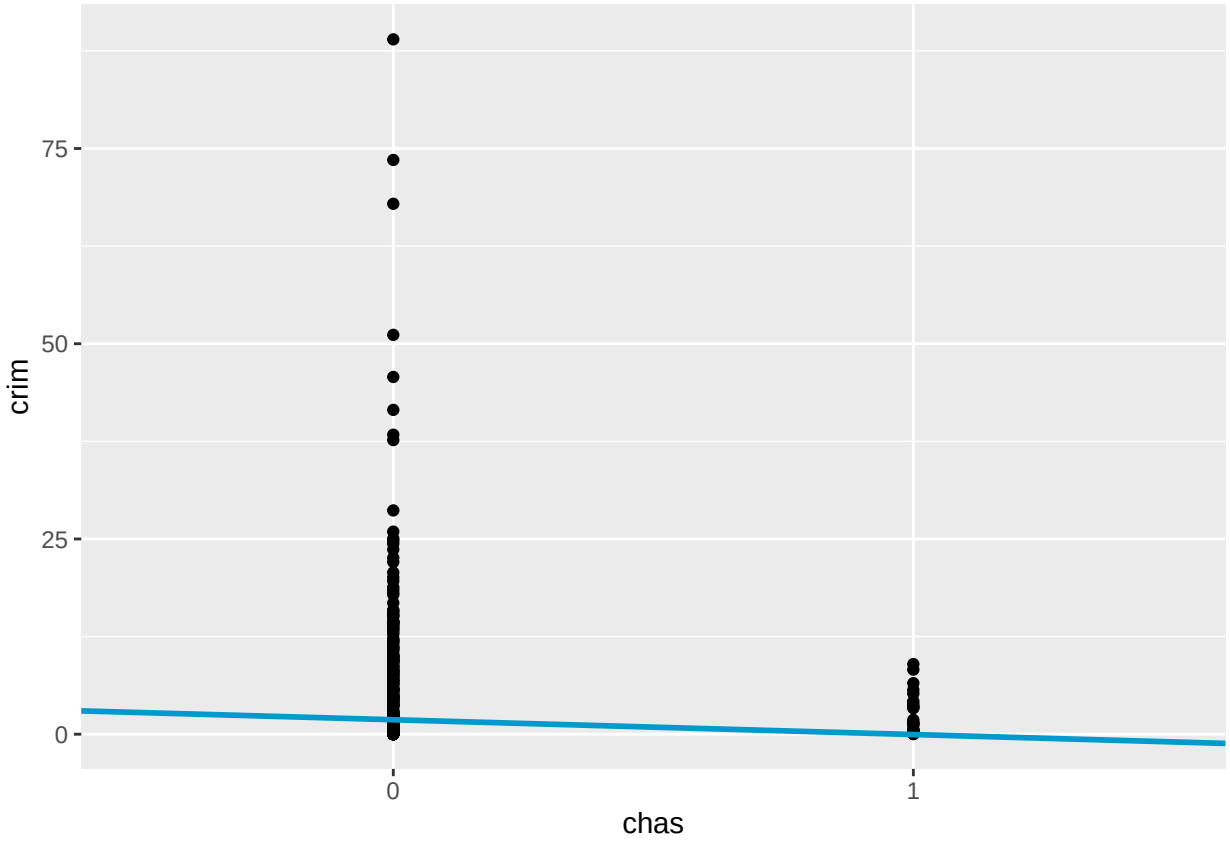

```
b_m2<-lm(crim~indus, data = Boston)
#summary(b_m2)
b_m2p<-ggplot(Boston, aes(x = indus, y = crim)) +
  geom_point() +
  geom_abline(intercept = coef(b_m2)[1], slope = coef(b_m2)[2],
              col = "deepskyblue3",
              size = 1) +
  geom_smooth(se = F)
b_m2p
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



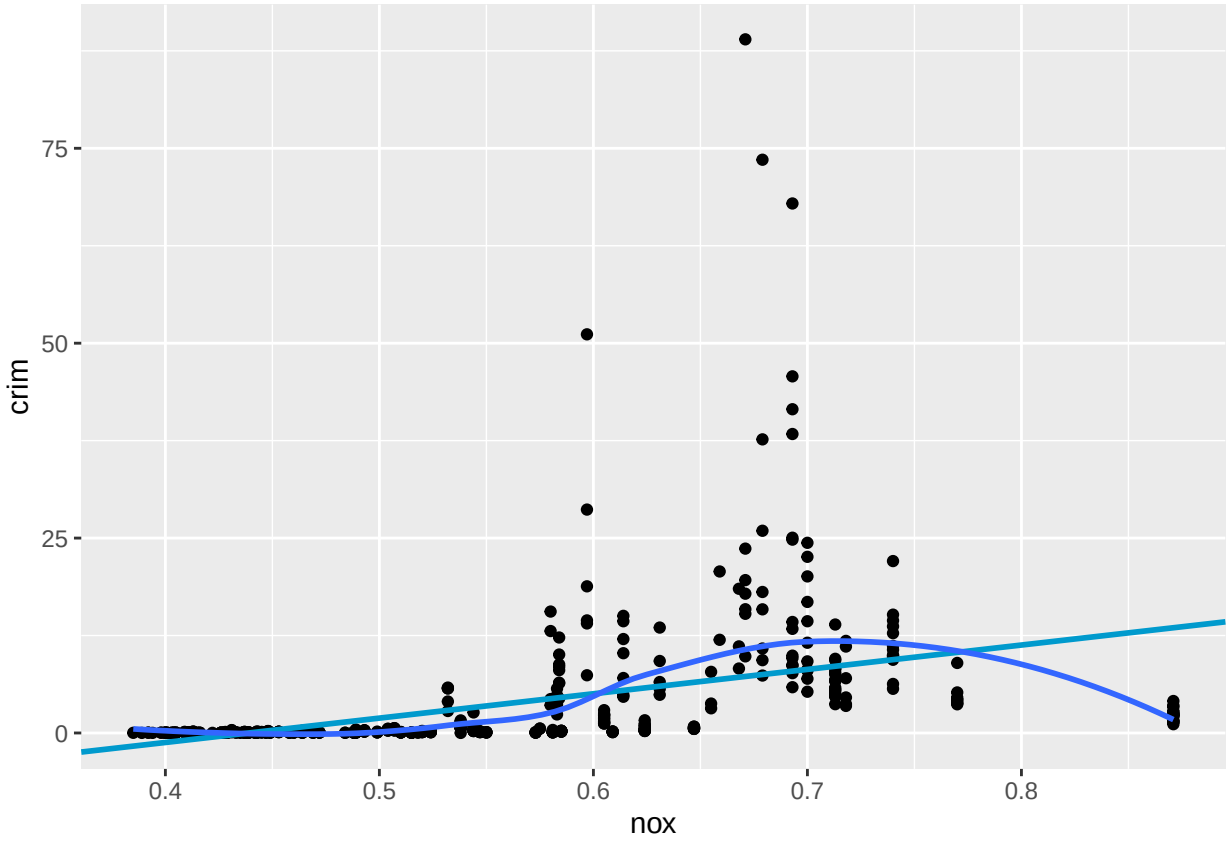

```
b_m3<-lm(crim~chas, data = Boston)
#summary(b_m3)
b_m3p<-ggplot(Boston, aes(x = chas, y = crim)) +
  geom_point() +
  geom_abline(intercept = coef(b_m3)[1], slope = coef(b_m3)[2],
              col = "deepskyblue3",
              size = 1) +
  geom_smooth(se = F)
b_m3p
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



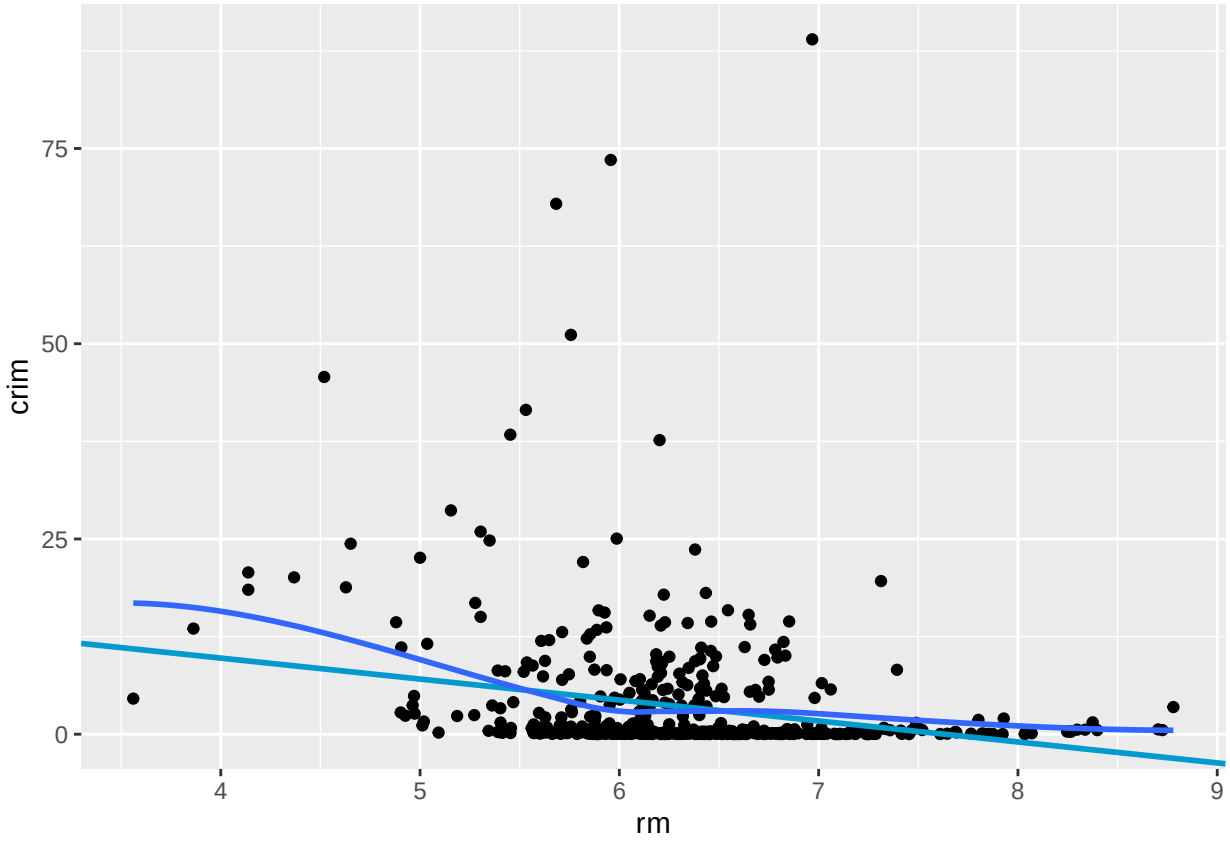

```
b_m4<-lm(crim~nox, data = Boston)
#summary(b_m4)
b_m4p<-ggplot(Boston, aes(x = nox, y = crim)) +
  geom_point() +
  geom_abline(intercept = coef(b_m4)[1], slope = coef(b_m4)[2],
              col = "deepskyblue3",
              size = 1) +
  geom_smooth(se = F)
b_m4p
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



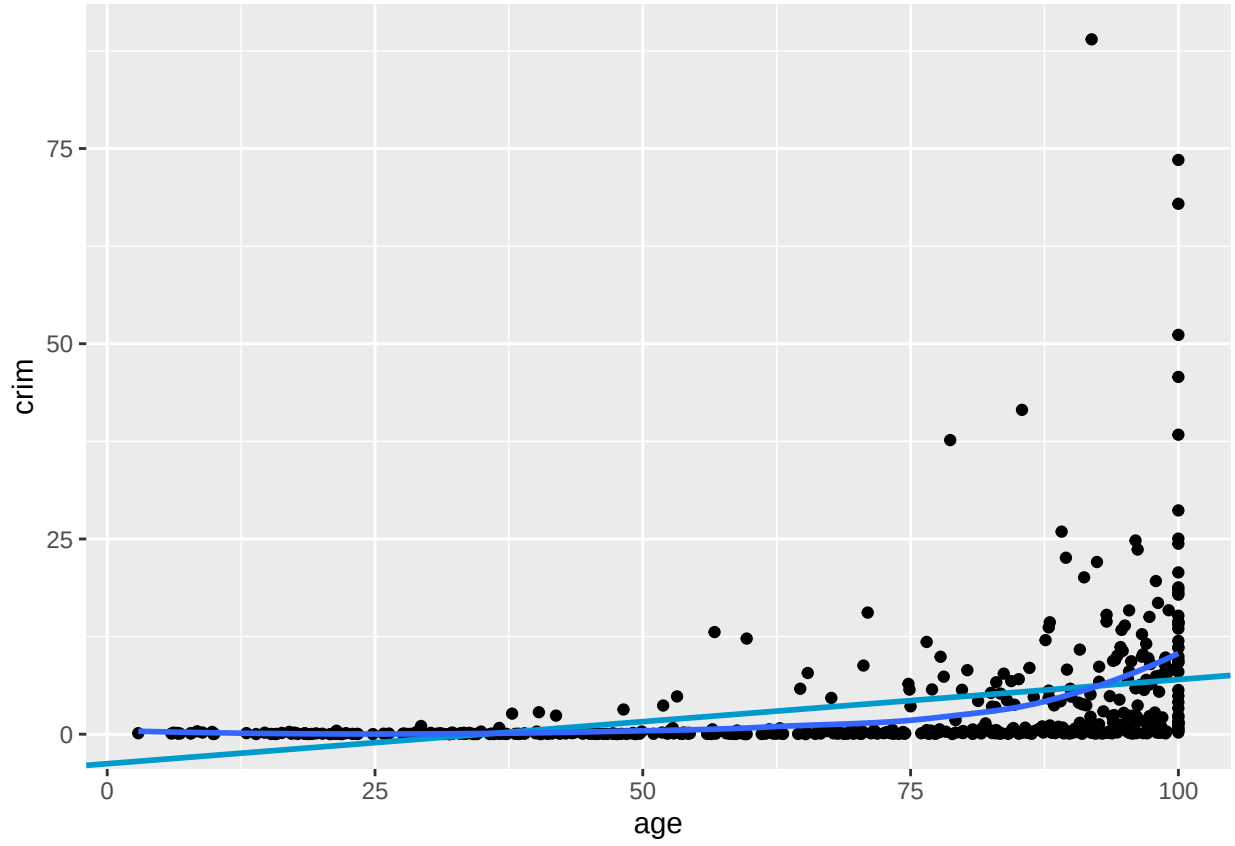

```
b_m5<-lm(crim~rm, data = Boston)
#summary(b_m5)
b_m5p<-ggplot(Boston, aes(x = rm, y = crim)) +
  geom_point() +
  geom_abline(intercept = coef(b_m5)[1], slope = coef(b_m5)[2],
              col = "deepskyblue3",
              size = 1) +
  geom_smooth(se = F)
b_m5p
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



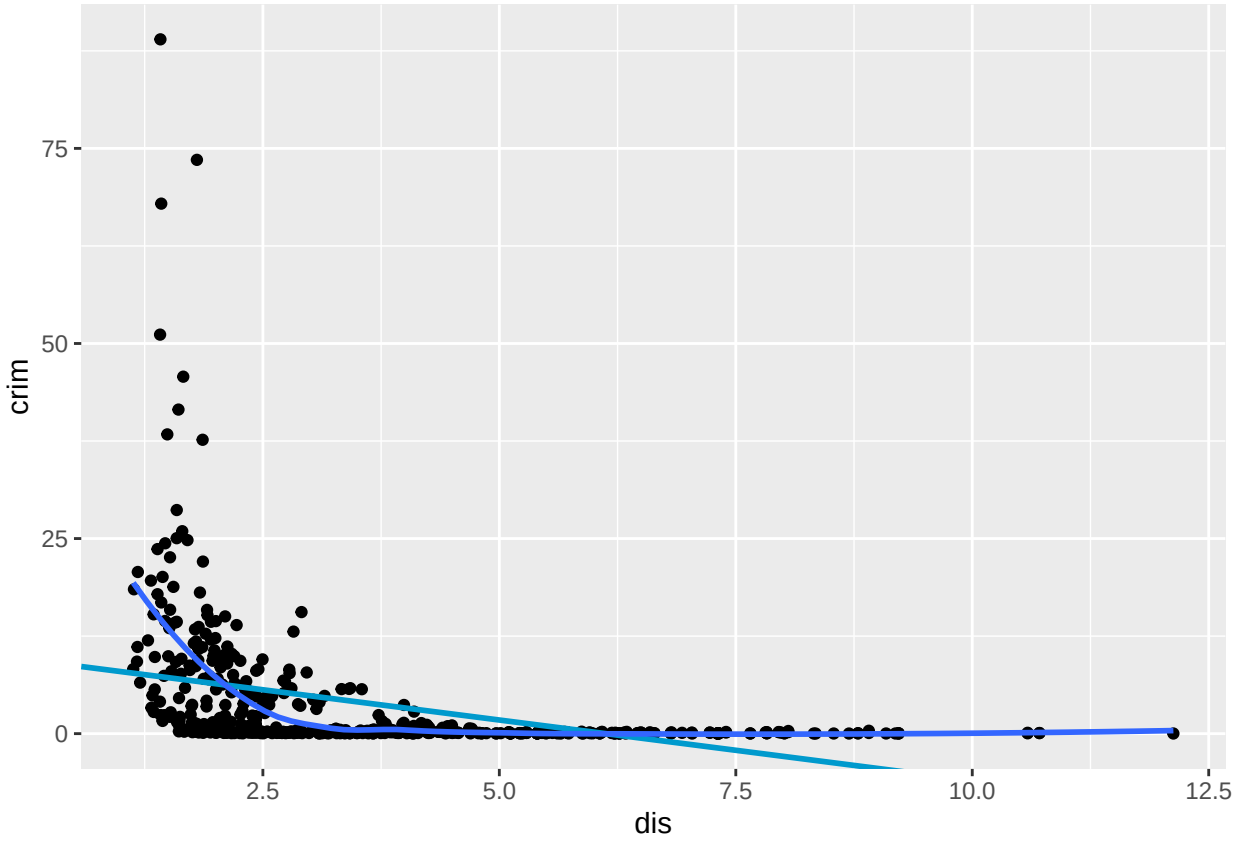

```
b_m6<-lm(crim~age, data = Boston)
#summary(b_m6)
b_m6p<-ggplot(Boston, aes(x = age, y = crim)) +
  geom_point() +
  geom_abline(intercept = coef(b_m6)[1], slope = coef(b_m6)[2],
              col = "deepskyblue3",
              size = 1) +
  geom_smooth(se = F)
b_m6p
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



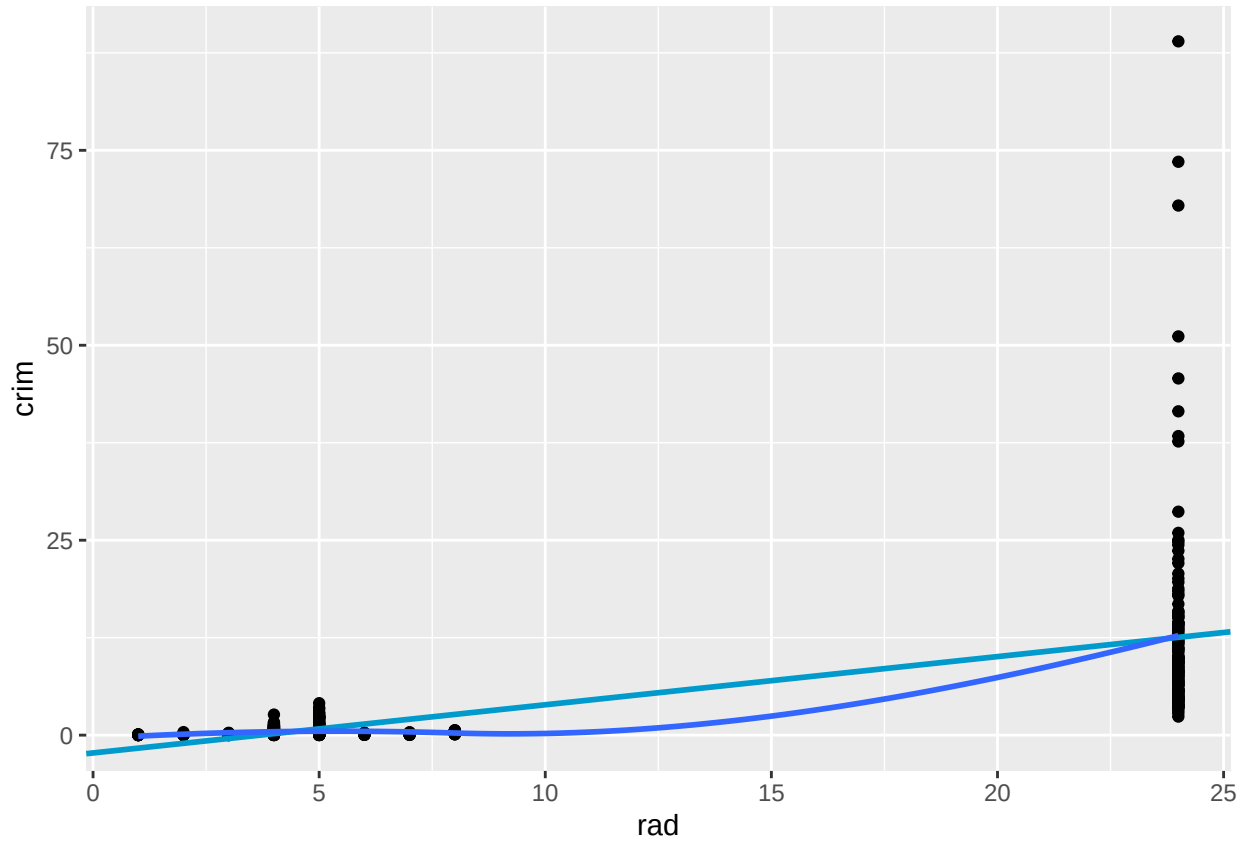

```
b_m7<-lm(crim~dis, data = Boston)
#summary(b_m7)
b_m7p<-ggplot(Boston, aes(x = dis, y = crim)) +
  geom_point() +
  geom_abline(intercept = coef(b_m7)[1], slope = coef(b_m7)[2],
              col = "deepskyblue3",
              size = 1) +
  geom_smooth(se = F)
b_m7p
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



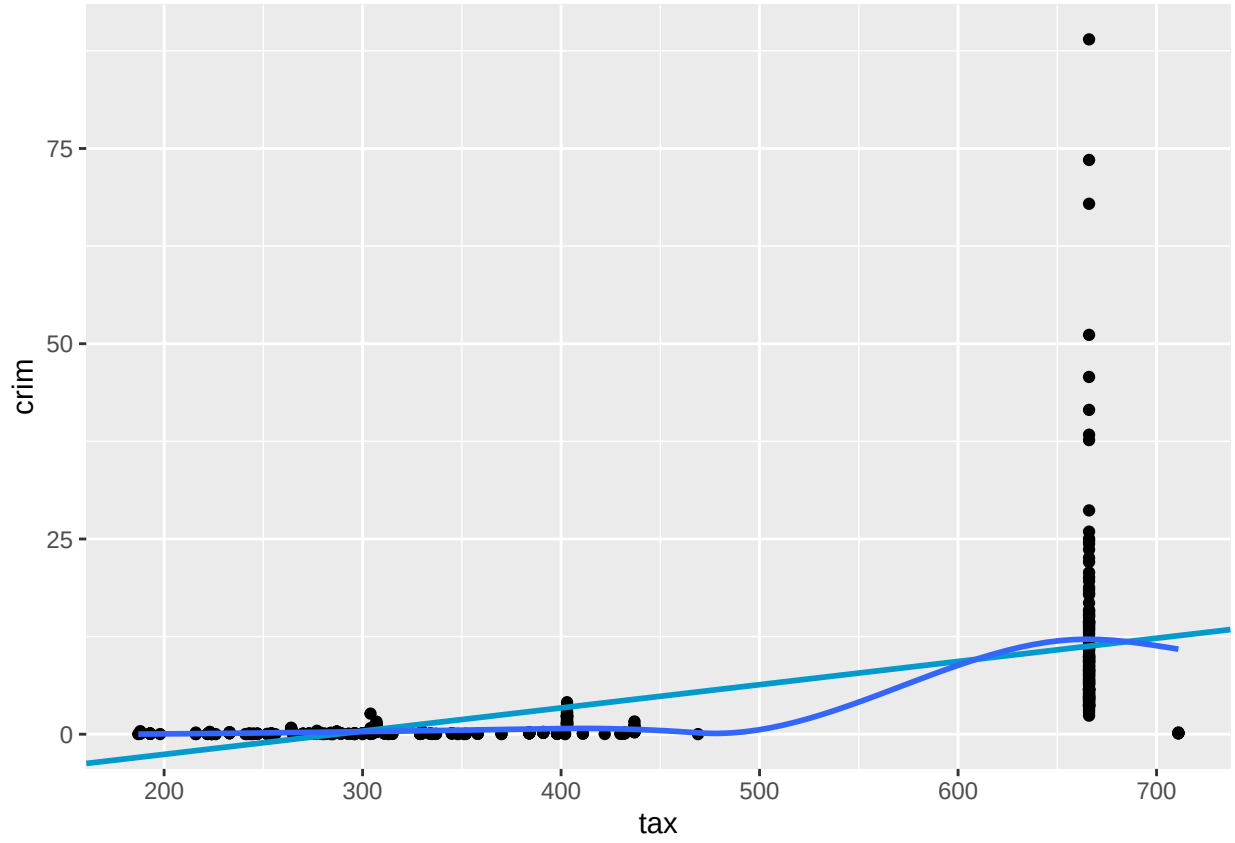

```
b_m8<-lm(crim~rad, data = Boston)
#summary(b_m8)
b_m8p<-ggplot(Boston, aes(x = rad, y = crim)) +
  geom_point() +
  geom_abline(intercept = coef(b_m8)[1], slope = coef(b_m8)[2],
              col = "deepskyblue3",
              size = 1) +
  geom_smooth(se = F)
b_m8p
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



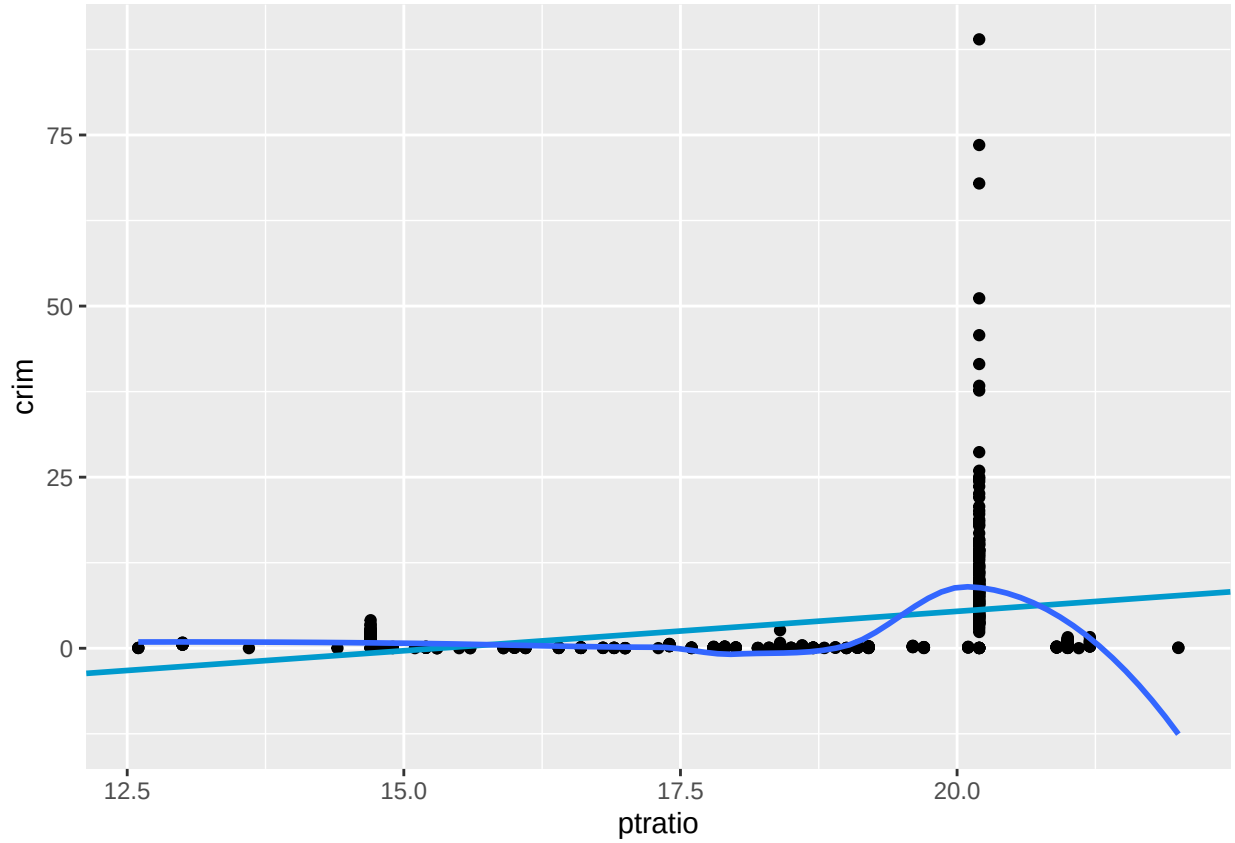

```
b_m9<-lm(crim~tax, data = Boston)
#summary(b_m9)
b_m9p<-ggplot(Boston, aes(x = tax, y = crim)) +
  geom_point() +
  geom_abline(intercept = coef(b_m9)[1], slope = coef(b_m9)[2],
              col = "deepskyblue3",
              size = 1) +
  geom_smooth(se = F)
b_m9p
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



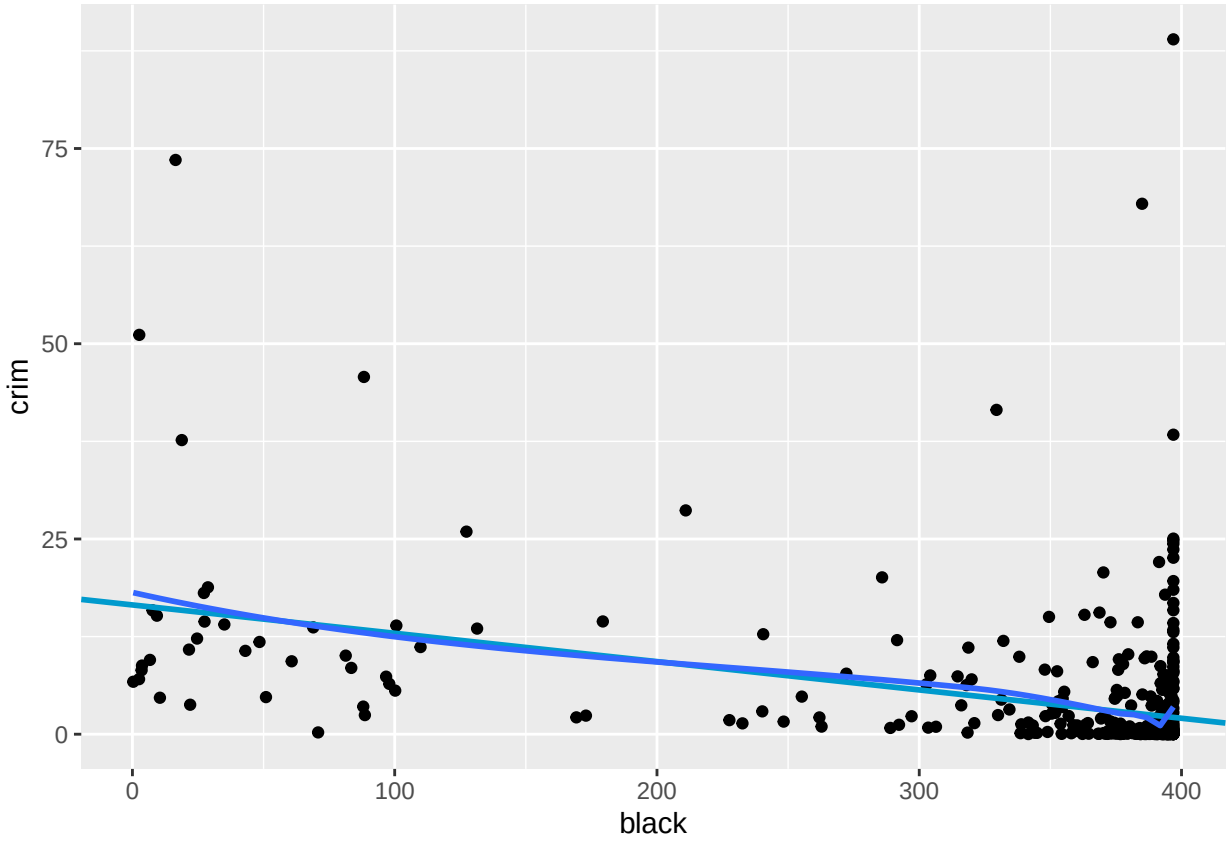

```
b_m10<-lm(crim~ptratio, data = Boston)
#summary(b_m10)
b_m10p<-ggplot(Boston, aes(x = ptratio, y = crim)) +
  geom_point() +
  geom_abline(intercept = coef(b_m10)[1], slope = coef(b_m10)[2],
              col = "deepskyblue3",
              size = 1) +
  geom_smooth(se = F)
b_m10p
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



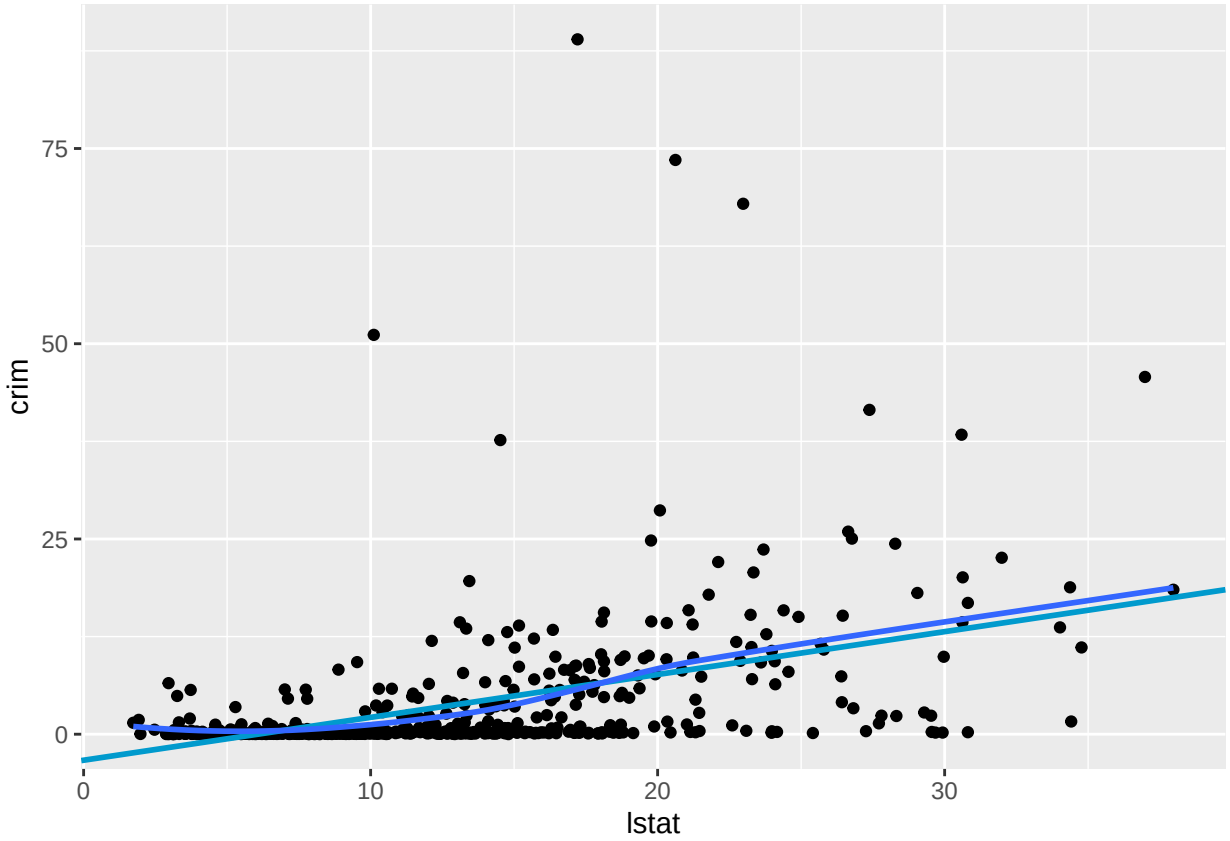

```
b_m11<-lm(crim~black, data = Boston)
#summary(b_m11)
b_m11p<-ggplot(Boston, aes(x = black, y = crim)) +
  geom_point() +
  geom_abline(intercept = coef(b_m11)[1], slope = coef(b_m11)[2],
              col = "deepskyblue3",
              size = 1) +
  geom_smooth(se = F)
b_m11p
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



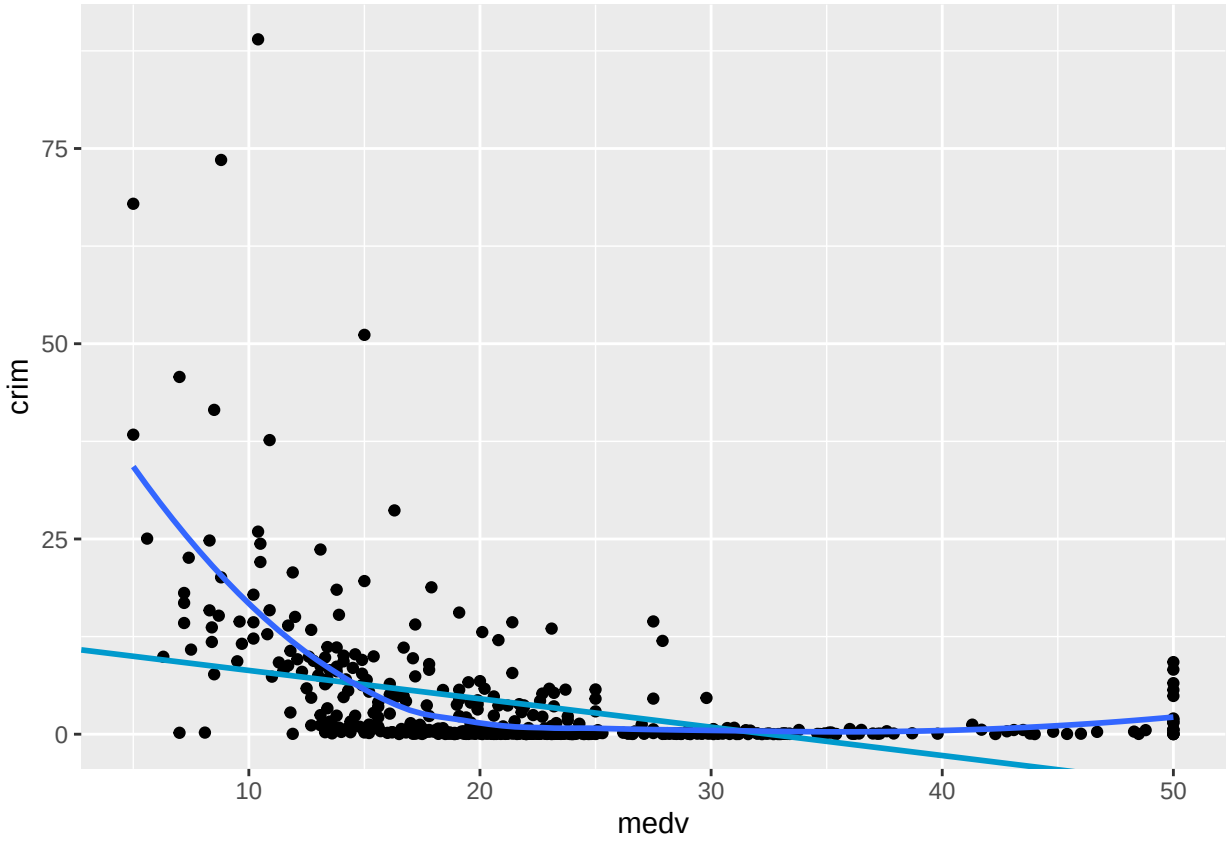

```
b_m12<-lm(crim~lstat, data = Boston)
#summary(b_m12)
b_m12p<-ggplot(Boston, aes(x = lstat, y = crim)) +
  geom_point() +
  geom_abline(intercept = coef(b_m12)[1], slope = coef(b_m12)[2],
              col = "deepskyblue3",
              size = 1) +
  geom_smooth(se = F)
b_m12p
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
b_m13<-lm(crim~medv, data = Boston)
#summary(b_m13)
b_m13p<-ggplot(Boston, aes(x = medv, y = crim)) +
  geom_point() +
  geom_abline(intercept = coef(b_m13)[1], slope = coef(b_m13)[2],
              col = "deepskyblue3",
              size = 1) +
  geom_smooth(se = F)
b_m13p
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The results are fairly clear, insofar as we can see that some of our predictors look like they are a better potential variable to use than others. The ones that stand out simply from looking at the graphs are the rad and tax predictors. The only predictor without a statistically significant association in our single simple linear regression model is the 'chas' variable, with a p-value greater than our alpha of 0.05.

B

```
b_m14<-lm(crim~., data = Boston)
summary(b_m14)
```

```
##
## Call:
## lm(formula = crim ~ ., data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.924 -2.120 -0.353  1.019 75.051
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  17.033228   7.234903   2.354 0.018949 *
## zn            0.044855   0.018734   2.394 0.017025 *
## indus        -0.063855   0.083407  -0.766 0.444294
## chas1        -0.749134   1.180147  -0.635 0.525867
## nox         -10.313535   5.275536  -1.955 0.051152 .
## rm            0.430131   0.612830   0.702 0.483089
## age           0.001452   0.017925   0.081 0.935488
```

```
## dis          -0.987176   0.281817  -3.503 0.000502 ***
## rad           0.588209   0.088049   6.680 6.46e-11 ***
## tax          -0.003780   0.005156  -0.733 0.463793
## ptratio      -0.271081   0.186450  -1.454 0.146611
## black        -0.007538   0.003673  -2.052 0.040702 *
## lstat         0.126211   0.075725   1.667 0.096208 .
## medv         -0.198887   0.060516  -3.287 0.001087 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.439 on 492 degrees of freedom
## Multiple R-squared:  0.454,  Adjusted R-squared:  0.4396
## F-statistic: 31.47 on 13 and 492 DF,  p-value: < 2.2e-16
```

Looking at our statistical output for our multiple regression, we can reject our null hypothesis for the predictors indus, chas, nox, rm, age, tax, ptratio, and lstat.

c

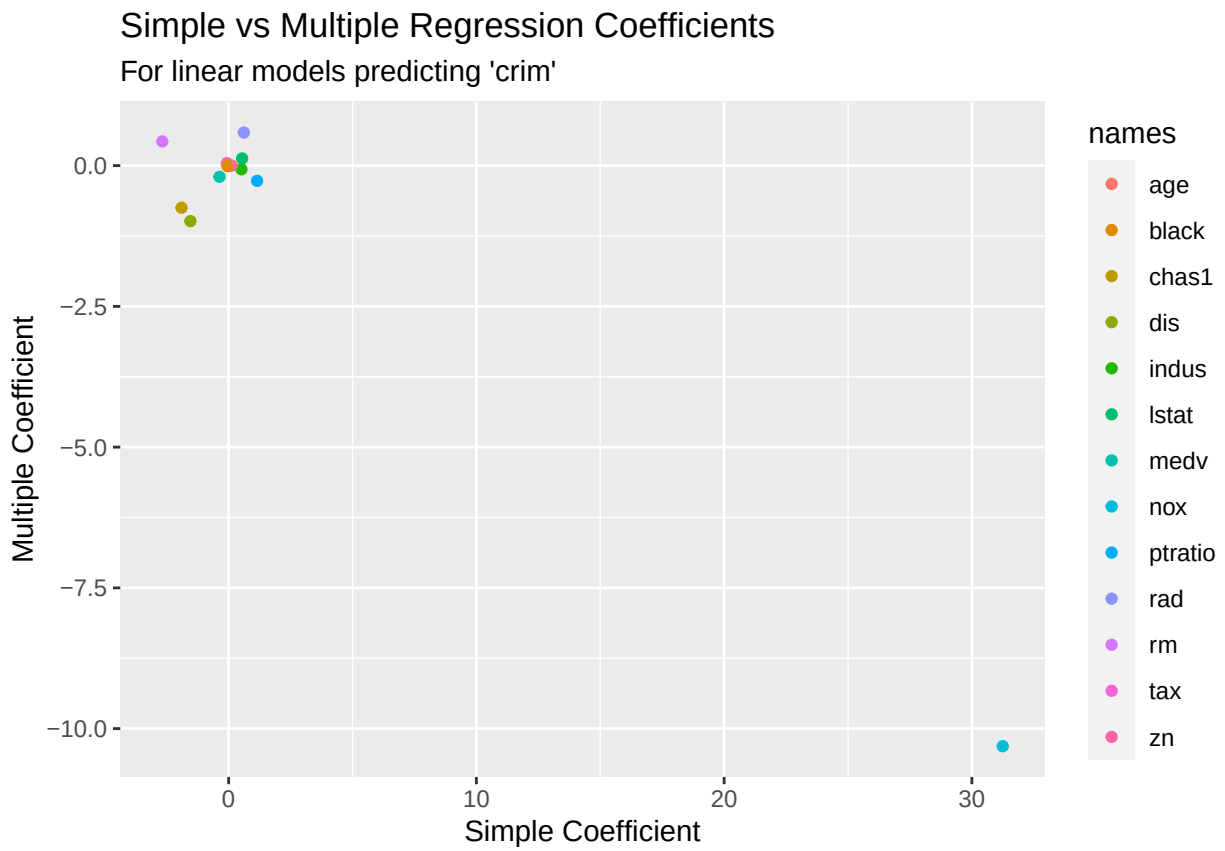
```
single_reg<-c(coef(b_m1)[[2]], coef(b_m2)[[2]],coef(b_m3)[[2]],coef(b_m4)[[2]],coef(b_m5)[[2]],coef(b_m
              coef(b_m8)[[2]],coef(b_m9)[[2]],coef(b_m10)[[2]],coef(b_m11)[[2]],coef(b_m12)[[2]],coef(b_
single_reg<-as.data.frame(single_reg)

multiple_reg<-b_m14[[1]]
multiple_reg<-as.data.frame(multiple_reg)
multiple_reg<-multiple_reg[-1,]
multiple_reg<-as.data.frame(multiple_reg)

names<-names((b_m14$coefficients))
names<-as.data.frame(names)
names<-names[-1,]
names<-as.data.frame(names)

regression_plot<-cbind(single_reg, multiple_reg,names)


ggplot(regression_plot, aes(x = single_reg, y = multiple_reg, col = names)) +
  geom_jitter() +
  labs(title = "Simple vs Multiple Regression Coefficients",
       subtitle = "For linear models predicting 'crim'",
       x = "Simple Coefficient",
       y = "Multiple Coefficient")
```



D

```
testlist<-list(NA)
for (i in 2:ncol(Boston)){
  testlist[[i-1]]<-summary(lm(Boston$crim ~ Boston[,i] + Boston_2[,i] + Boston_3[,i] ))
}
#print(testlist)
#commenting out this code in order to reduce size of output
```

There is evidence of a nonlinear association between several of our predictors and their responses. The models from 1 to 13 follow the ordering of the columns in our dataset, Boston.

Given this, we can clearly see that there is a cubic relationship for the medv, nox, dis, indus, age, and ptratio predictors.

6

Given $f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2)$

We begin with our function $p_k(x)$

$$p_k(x) = \frac{\pi_k \cdot f_k(x)}{\sum_{l=1}^{K} \pi_l \cdot f_l(x)}$$

Taking the natural log, we end up with

$$ln(p_k(x)) = ln(\pi_k) - ln(\sqrt{2\pi_l}\sigma) - \frac{(x-\mu_k)^2}{2\sigma^2} - ln(\sum_{l=1}^{K} \pi_l \cdot f_l(x))$$

Simplifying and reducing the elements, we come to

$$ln(p_k(x)) = ln(\pi_k) - .5 * ln(2\pi_l * \sigma^2) - \frac{x^2 - 2\mu_k x + \mu^2}{2\sigma^2} - ln(\sum_{l=1}^{K} \pi_l \cdot f_l(x))$$

Simplifying further, we approach our 'final' form which is

$$ln(p_k(x)) = x * \frac{\mu_k}{\sigma_k^2} - \frac{\mu_k^2}{2\sigma_k^2} - \frac{x^2}{2\sigma_k^2} - .5 * ln(2\pi_l * \sigma^2) - ln(\sum_{l=1}^{K} \pi_l \cdot f_l(x))$$

Given that we can 'cancel' out the righthand most term in our comparisions between classes, as it is the summated probability across all classes (which does not change), the remaining terms comprise the differences between classes μ_k and σ_k.

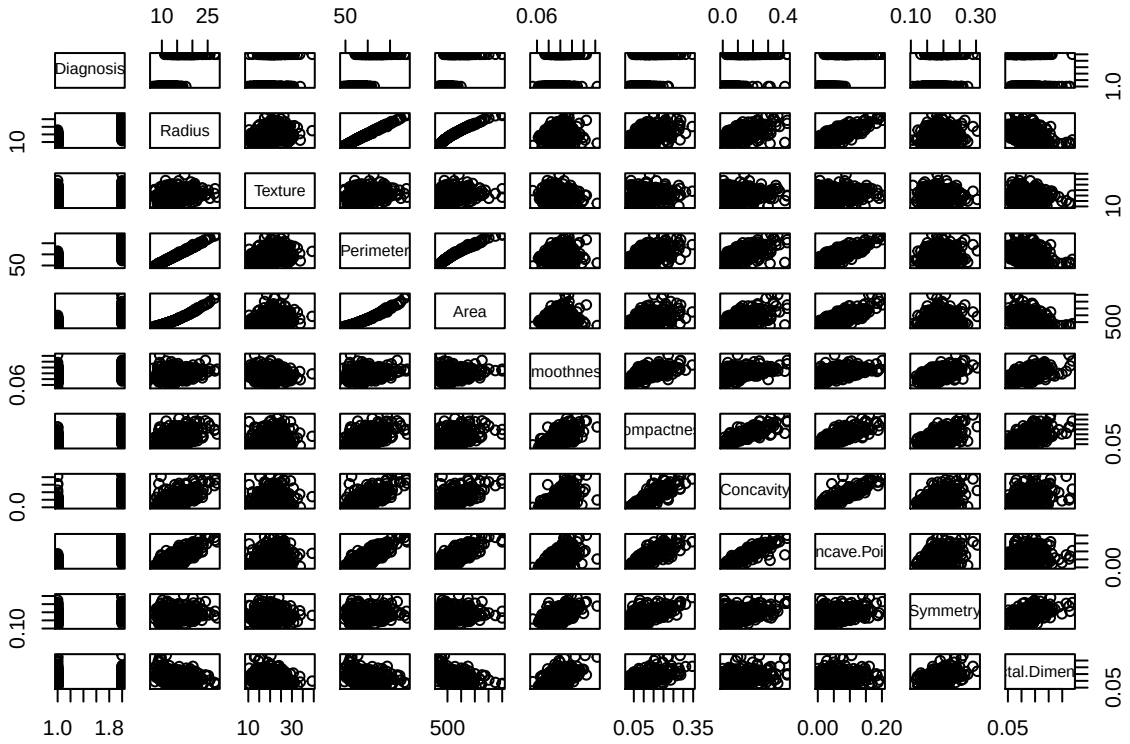
We can thus see that our Bayes classifier is nonlinear as we have a x^2 term remaining in our formula.

The reason why this is nonlinear in the QDA case versus the LDA case is because in the LDA case, any term with σ is allowed to be 'cancelled out' when comparing each class to each other, as it is assumed that they all share the same σ term when doing LDA. Looking at our formula for the QDA, when you are not allowed to 'cancel' out the terms with σ in them because the classes are assumed to have different variance/covariance matrices, we are left with a remaining x^2 term, thus rendering our bayes classifier as nonlinear.

7

A

```
tumor<-read.csv("tumor.csv")
tumor$Diagnosis<-as.factor(tumor$Diagnosis)
pairs(tumor)
```



Radius, perimeter, and concave points seem most related to outcome.

Some features are highly related, radius and perimter, radius and area, area highly so.

Some features which are related strongly, but less so than the previously mentioned ones are perimeter, concavity/concave points, and compactness/concavity.

B

```
set.seed(1)
test=1:512
train_tumor<-tumor[test,]
test_tumor<-tumor[-test,]
nrow(test_tumor)
```

```
## [1] 57
```

C

```
tumor_logistic<-glm(Diagnosis~Radius+Symmetry+Concave.Points, data = train_tumor, family = binomial)

tumor_logistic_probs<-predict(tumor_logistic, test_tumor,type = "response")
tumor_logistic_pred<-rep("Benign", 57)
tumor_logistic_pred[tumor_logistic_probs >.5]= "Malignant"
table(tumor_logistic_pred, test_tumor$Diagnosis)
```

```
##
## tumor_logistic_pred Benign Malignant
```

```
##           Benign        42         1
##           Malignant      1        13
```

```
mean(tumor_logistic_pred==test_tumor$Diagnosis)
```

```
## [1] 0.9649123
```

```
1-mean(tumor_logistic_pred==test_tumor$Diagnosis)
```

```
## [1] 0.03508772
```

Our fraction of correct predictions is roughly 96/100 Our misclassification rate is roughly 3%

D

```
tumor_logistic_pred2<-rep("Benign", 57)
tumor_logistic_pred2[tumor_logistic_probs >.25]= "Malignant"
table(tumor_logistic_pred2, test_tumor$Diagnosis)
```

```
##
## tumor_logistic_pred2 Benign Malignant
##            Benign        40         0
##            Malignant      3        14
```

```
mean(tumor_logistic_pred2==test_tumor$Diagnosis)
```

```
## [1] 0.9473684
```

```
1-mean(tumor_logistic_pred2==test_tumor$Diagnosis)
```

```
## [1] 0.05263158
```

Our fraction of correct predictions is roughly 94/100 Our misclassification rate is roughly 5%

Compared to our model with a higher classification threshold it looks like we have an increased rate of false positives, but have decreased our rate of false negatives. If this is a situation where false negatives are an extreme deterrant compared to false positives, the improvement in our false negative rate may be worth the increase in our misclassification rate overall.

E

```
tumor_lda<-lda(Diagnosis~Radius+Symmetry+Concave.Points, data = train_tumor)
tumor_lda_pred<-predict(tumor_lda, test_tumor)

tumor_lda_class<-tumor_lda_pred$class

table(tumor_lda_class, test_tumor$Diagnosis)
```

```
##
## tumor_lda_class Benign Malignant
##       Benign        43         3
##       Malignant      0        11
```

```
1-mean(tumor_lda_class==test_tumor$Diagnosis)
```

```
## [1] 0.05263158
```

The test error is roughly 5%

F

```
tumor_qda<-qda(Diagnosis~Radius+Symmetry+Concave.Points, data = train_tumor)

tumor_qda_class<-predict(tumor_qda, test_tumor)$class

table(tumor_qda_class, test_tumor$Diagnosis)
```

```
##
## tumor_qda_class Benign Malignant
##       Benign        42         1
##       Malignant      1        13
```

```
1-mean(tumor_qda_class==test_tumor$Diagnosis)
```

```
## [1] 0.03508772
```

The test error is roughly 3% ### G

```
train_tumor_x<-cbind(train_tumor$Radius,train_tumor$Symmetry,train_tumor$Concave.Points)
test_tumor_x<-cbind(test_tumor$Radius,test_tumor$Symmetry,test_tumor$Concave.Points)
train_tumor_direction<-train_tumor$Diagnosis

tumor_knn_1<-knn(train_tumor_x,test_tumor_x,train_tumor_direction ,k=1)
table(tumor_knn_1 , test_tumor$Diagnosis)
```

```
##
## tumor_knn_1 Benign Malignant
##   Benign        36         2
##   Malignant      7        12
```

```
1-mean(tumor_knn_1==test_tumor$Diagnosis)
```

```
## [1] 0.1578947
```

```
tumor_knn_2<-knn(train_tumor_x,test_tumor_x,train_tumor_direction ,k=2)
table(tumor_knn_2 , test_tumor$Diagnosis)
```

```
##
## tumor_knn_2 Benign Malignant
##   Benign        37         2
##   Malignant      6        12
```

```
1-mean(tumor_knn_2==test_tumor$Diagnosis)
```

```
## [1] 0.1403509
```

```
tumor_knn_3<-knn(train_tumor_x,test_tumor_x,train_tumor_direction ,k=3)
table(tumor_knn_3 , test_tumor$Diagnosis)
```

```
##
## tumor_knn_3 Benign Malignant
##   Benign        40         0
##   Malignant      3        14
```

```
1-mean(tumor_knn_3==test_tumor$Diagnosis)
```

```
## [1] 0.05263158
```

```
tumor_knn_4<-knn(train_tumor_x,test_tumor_x,train_tumor_direction ,k=4)
table(tumor_knn_4 , test_tumor$Diagnosis)
```

```
##
## tumor_knn_4 Benign Malignant
##   Benign        40         1
##   Malignant      3        13
```

```
1-mean(tumor_knn_4==test_tumor$Diagnosis)
```

```
## [1] 0.07017544
```

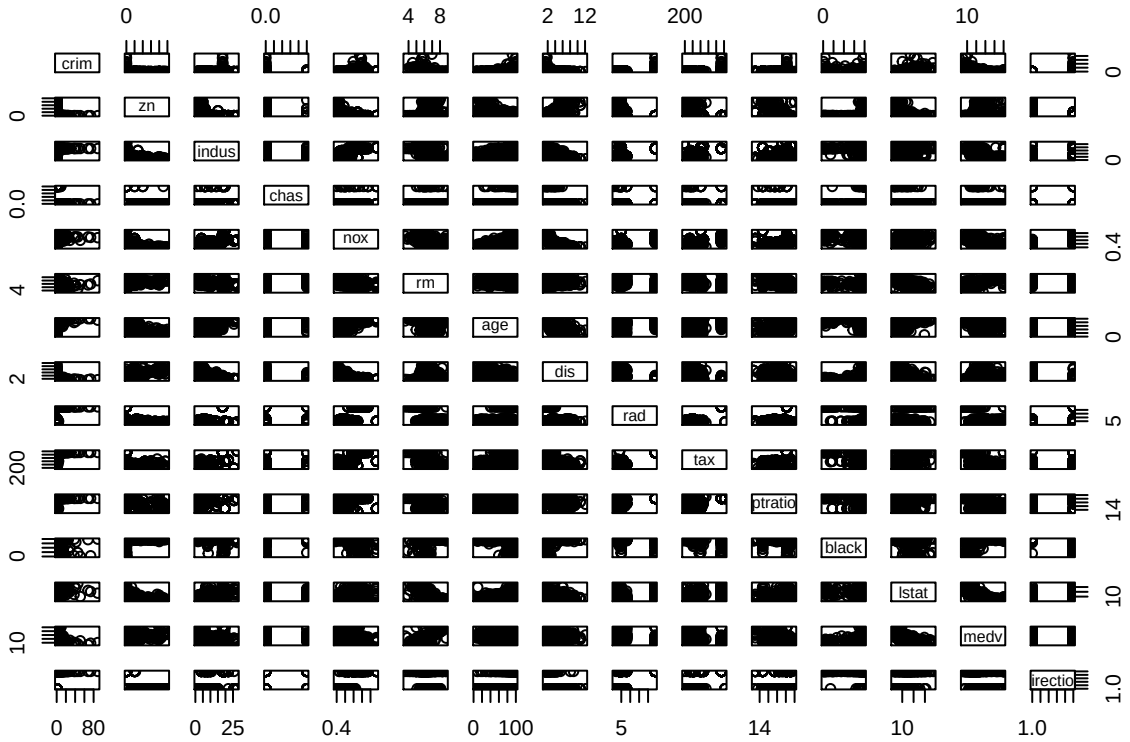
Our error rate for K = 1 is roughly 15% Our error rate for K = 2 is roughly 15% Our error rate for K = 3 is roughly 5% Our error rate for K = 4 is roughly 2%

The values chosen for our KNN approximation were 1, 2, 3, and 4. We see that our error rate is smallest for our model with 4 nearest neighbors, which is the value of k that works best for the data

8

Graphing

```
data(Boston)
Boston$direction<-rep("Up", 506)
Boston$direction[Boston$crim <median(Boston$crim)]="Down"
Boston$direction<-as.factor(Boston$direction)
pairs(Boston[])
```



```
test2<-(1:455)
Boston_train<-Boston[test2,]
Boston_test<-Boston[-test2,]
```

We begin by looking at a scatterplot matrix of all of our predictors.Looking at the graph, we can see that rad, ptratio, nox, and dis look most strongly related to our outcome of whether or not crime is above or below the median.

For the purpose of this problem, I will look at two sets of the predictors. The first set will be every single predictor in our data set, our second set will be the predictors that our graphical analysis look to be most related to crime, namely rad, ptratio, box, and dis.

Additionally, we will be splitting our dataset into a 90% size training set, and a 10% testing set. ### Logistic Regression full set

```
Boston_logistic_1<-glm(direction~.-crim, data = Boston_train, family = binomial)
summary(Boston_logistic_1)
```

```
##
## Call:
## glm(formula = direction ~ . - crim, family = binomial, data = Boston_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3643  -0.1256  -0.0020   0.0005   3.4545
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -42.799138   7.890630  -5.424 5.83e-08 ***
## zn           -0.066836   0.036978  -1.807 0.070691 .
## indus        -0.088478   0.051755  -1.710 0.087348 .
## chas          1.023591   0.753022   1.359 0.174048
## nox          59.170914   9.555384   6.192 5.92e-10 ***
## rm           -0.676178   0.816091  -0.829 0.407355
## age           0.008651   0.012974   0.667 0.504912
## dis           0.654216   0.232581   2.813 0.004910 **
## rad           0.621349   0.183643   3.383 0.000716 ***
## tax          -0.001433   0.003760  -0.381 0.703171
## ptratio       0.485266   0.141218   3.436 0.000590 ***
## black        -0.009549   0.006112  -1.562 0.118191
## lstat         0.068709   0.054149   1.269 0.204479
## medv          0.202732   0.080269   2.526 0.011548 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 629.79  on 454  degrees of freedom
## Residual deviance: 182.50  on 441  degrees of freedom
## AIC: 210.5
##
## Number of Fisher Scoring iterations: 9
```

```
Boston_logistic_probs_1<-predict(Boston_logistic_1, Boston_test,type = "response")
Boston_logistic_pred_1<-rep("Down", 51)
Boston_logistic_pred_1[Boston_logistic_probs_1 >.5]= "Up"
```

```
table(Boston_logistic_pred_1, Boston_test$direction)
```

```
##
## Boston_logistic_pred_1 Down Up
##                   Down    5  0
##                   Up     10 36
```

```
1-mean(Boston_logistic_pred_1==Boston_test$direction)
```

```
## [1] 0.1960784
```

For our logistic regression on the full set of predictors, our misclassification rate looks to be roughly 20%

Logistic Regression reduced set

```
Boston_logistic_2<-glm(direction~ rad + nox + ptratio + dis, data = Boston_train, family = binomial)
summary(Boston_logistic_2)
```

```
##
## Call:
## glm(formula = direction ~ rad + nox + ptratio + dis, family = binomial,
##     data = Boston_train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.78870  -0.18714  -0.02149   0.00137   2.91193
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -33.15965    5.19677  -6.381 1.76e-10 ***
## rad           0.67434    0.13459   5.011 5.43e-07 ***
## nox          45.09503    6.75202   6.679 2.41e-11 ***
## ptratio       0.27726    0.09428   2.941  0.00327 **
## dis           0.18749    0.15085   1.243  0.21391
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 629.79  on 454  degrees of freedom
## Residual deviance: 209.53  on 450  degrees of freedom
## AIC: 219.53
##
## Number of Fisher Scoring iterations: 9
```

```
Boston_logistic_probs_2<-predict(Boston_logistic_2, Boston_test,type = "response")
Boston_logistic_pred_2<-rep("Down", 51)
Boston_logistic_pred_2[Boston_logistic_probs_2 >.5]= "Up"
table(Boston_logistic_pred_2, Boston_test$direction)
```

```
##
## Boston_logistic_pred_2 Down Up
##                   Down    5  0
##                   Up     10 36
```

```
1-mean(Boston_logistic_pred_2==Boston_test$direction)
```

```
## [1] 0.1960784
```

For our logistic regression on the reduced set of predictors, our misclassification rate looks to be roughly 20%

LDA full set

```
Boston_lda_1<-lda(direction~.-crim, data = Boston_train, family = binomial)
Boston_lda_1
```

```
## Call:
## lda(direction ~ . - crim, data = Boston_train, family = binomial)
##
## Prior probabilities of groups:
##      Down        Up
## 0.5230769 0.4769231
##
## Group means:
##             zn     indus       chas       nox       rm      age      dis
## Down 22.882353  6.406639 0.05462185 0.4635324 6.421639 49.58824 5.270933
## Up    1.400922 14.918249 0.10138249 0.6417005 6.180756 87.40691 2.439953
##            rad      tax  ptratio    black     lstat     medv
## Down  4.189076 296.1261 17.76891 388.8326  9.114664 25.40756
## Up   13.686636 488.7742 18.81935 322.9463 16.004654 20.27650
##
## Coefficients of linear discriminants:
##                  LD1
## zn      -0.003072272
## indus    0.004529749
## chas    -0.024269455
## nox      8.868370286
## rm       0.142655670
## age      0.011668842
## dis     -0.004674052
## rad      0.001701238
## tax      0.001550333
## ptratio  0.104978128
## black   -0.001610518
## lstat    0.015121684
## medv     0.036348661
```

```
Boston_lda_pred_1<-predict(Boston_lda_1, Boston_test)
Boston_lda_class_1<-Boston_lda_pred_1$class

table(Boston_lda_class_1, Boston_test$direction)
```

```
##
## Boston_lda_class_1 Down Up
##               Down    2  1
##               Up     13 35
```

```
1-mean(Boston_lda_class_1==Boston_test$direction)
```

```
## [1] 0.2745098
```

For our LDA on our full set of predictors, our misclassification rate looks to be 27%

LDA reduced set

```
Boston_lda_2<-lda(direction~ rad + nox + ptratio + dis, data = Boston_train, family = binomial)
Boston_lda_2
```

```
## Call:
## lda(direction ~ rad + nox + ptratio + dis, data = Boston_train,
##     family = binomial)
##
## Prior probabilities of groups:
##      Down        Up
## 0.5230769 0.4769231
##
## Group means:
##            rad       nox  ptratio      dis
## Down  4.189076 0.4635324 17.76891 5.270933
## Up   13.686636 0.6417005 18.81935 2.439953
##
## Coefficients of linear discriminants:
##                 LD1
## rad      0.03266363
## nox      9.06385633
## ptratio  0.05953421
## dis     -0.14746618
```

```
Boston_lda_pred_2<-predict(Boston_lda_2, Boston_test)
Boston_lda_class_2<-Boston_lda_pred_2$class

table(Boston_lda_class_2, Boston_test$direction)
```

```
##
## Boston_lda_class_2 Down Up
##               Down    0  0
##               Up     15 36
```

```
1-mean(Boston_lda_class_2==Boston_test$direction)
```

```
## [1] 0.2941176
```

For our LDA on our full set of predictors, our misclassification rate looks to be 29%

KNN full set

```
Boston_train_x_1<-Boston_train[,c(-1,-15)]
Boston_test_x_1<-Boston_test[,c(-1,-15)]
Boston_train_direction<-Boston_train$direction

Boston_knn_1_A<-knn(Boston_train_x_1,Boston_test_x_1,Boston_train_direction,k=1)
table(Boston_knn_1_A , Boston_test$direction)
```

```
##
## Boston_knn_1_A Down Up
##           Down   10  3
##           Up      5 33
```

```
1-mean(Boston_knn_1_A==Boston_test$direction)
```

```
## [1] 0.1568627
```

```
Boston_knn_1_B<-knn(Boston_train_x_1,Boston_test_x_1,Boston_train_direction,k=2)
table(Boston_knn_1_B , Boston_test$direction)
```

```
##
## Boston_knn_1_B Down Up
##           Down   10  3
##           Up      5 33
```

```
1-mean(Boston_knn_1_B==Boston_test$direction)
```

```
## [1] 0.1568627
```

```
Boston_knn_1_C<-knn(Boston_train_x_1,Boston_test_x_1,Boston_train_direction,k=4)
table(Boston_knn_1_C , Boston_test$direction)
```

```
##
## Boston_knn_1_C Down Up
##           Down   10  3
##           Up      5 33
```

```
1-mean(Boston_knn_1_C==Boston_test$direction)
```

```
## [1] 0.1568627
```

KNN reduced set

```
Boston_train_x_2<-cbind(Boston_train$rad, Boston_train$nox, Boston_train$ptratio, Boston_train$dis)
Boston_test_x_2<-cbind(Boston_test$rad, Boston_test$nox, Boston_test$ptratio, Boston_test$dis)

Boston_knn_2_A<-knn(Boston_train_x_2,Boston_test_x_2,Boston_train_direction,k=1)
table(Boston_knn_2_A , Boston_test$direction)
```

```
##
## Boston_knn_2_A Down Up
##           Down   10  3
##           Up      5 33
```

```
1-mean(Boston_knn_2_A==Boston_test$direction)
```

```
## [1] 0.1568627
```

```
Boston_knn_2_B<-knn(Boston_train_x_2,Boston_test_x_2,Boston_train_direction,k=2)
table(Boston_knn_2_B , Boston_test$direction)
```

```
##
## Boston_knn_2_B Down Up
##           Down   10  3
##           Up      5 33
```

```
1-mean(Boston_knn_2_B==Boston_test$direction)
```

```
## [1] 0.1568627
```

```
Boston_knn_2_C<-knn(Boston_train_x_2,Boston_test_x_2,Boston_train_direction,k=10)
table(Boston_knn_2_C , Boston_test$direction)
```

```
##
## Boston_knn_2_C Down Up
##           Down   10  3
##           Up      5 33
```

```
1-mean(Boston_knn_2_C==Boston_test$direction)
```

```
## [1] 0.1568627
```

Shockingly, we see no difference in either of our KNN classifications, with the full or reduced set of parameters. Our misclassification rate looks to be 15%.

Looking at all our methods, I feel like using a KNN classification method would be the most effective, and for parsimony's sake, I would choose the reduced set of predictors, with K = 1 for ease of computation.