Data Mining Assignment3

Yaniv Bronshtein

3/31/2021

Import the necessary libraries

```
library(ISLR)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 4.0.4

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(class)
library(broom)
library(boot)
```

```
## Warning: package 'boot' was built under R version 4.0.4
```

```
library(splines)
library(glm2)
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:glm2':
##
##     crabs

## The following object is masked from 'package:dplyr':
##
##     select
```

Problem 3

Pg 171-172 11.In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set.

a). Create a binary variable, mpg01, that contains a 1 if mpg cotnains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() function. Note you may find it helpful to use the data.frame() function to create a single data set containing both mpg01 and the other Auto variables **Get auto data as a tibble**

```
auto_t <- Auto %>% as_tibble()
```

Create the mpg01 calculated numeric column

```
median <- median(auto_t$mpg)
auto_t <- auto_t %>%
  mutate(mpg01 = ifelse(mpg > median, 1, 0))
auto_t %>% head(10)
```

```
## # A tibble: 10 x 10
##      mpg cylinders displacement horsepower weight acceleration  year origin
##    <dbl>     <dbl>        <dbl>      <dbl>  <dbl>        <dbl> <dbl>  <dbl>
##  1    18         8          307        130   3504         12      70      1
##  2    15         8          350        165   3693         11.5    70      1
##  3    18         8          318        150   3436         11      70      1
##  4    16         8          304        150   3433         12      70      1
##  5    17         8          302        140   3449         10.5    70      1
##  6    15         8          429        198   4341         10      70      1
##  7    14         8          454        220   4354          9      70      1
##  8    14         8          440        215   4312          8.5    70      1
##  9    14         8          455        225   4425         10      70      1
## 10    15         8          390        190   3850          8.5    70      1
## # ... with 2 more variables: name <fct>, mpg01 <dbl>
```

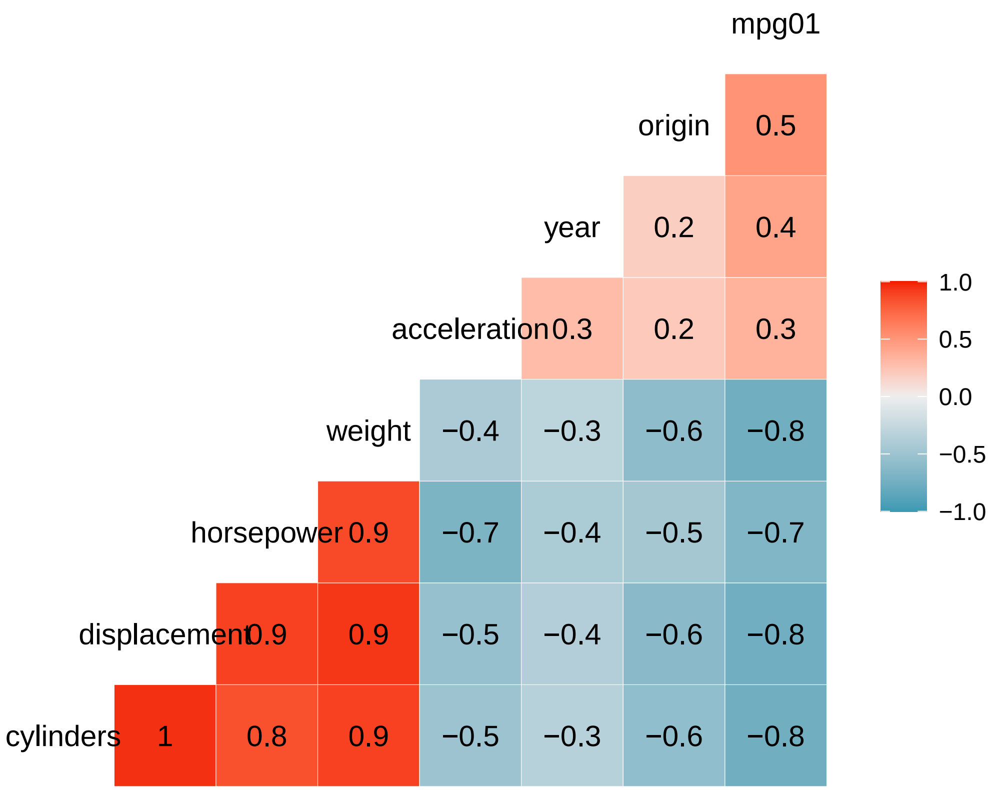
Remove the two non-numeric columns

```
auto_t <- dplyr::select(auto_t, -mpg)
auto_t <- dplyr::select(auto_t, -name)
```

b). Explore the data graphically in order to investigate the association between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings

Create a correlation plot to determine the features most correlated with mpg01

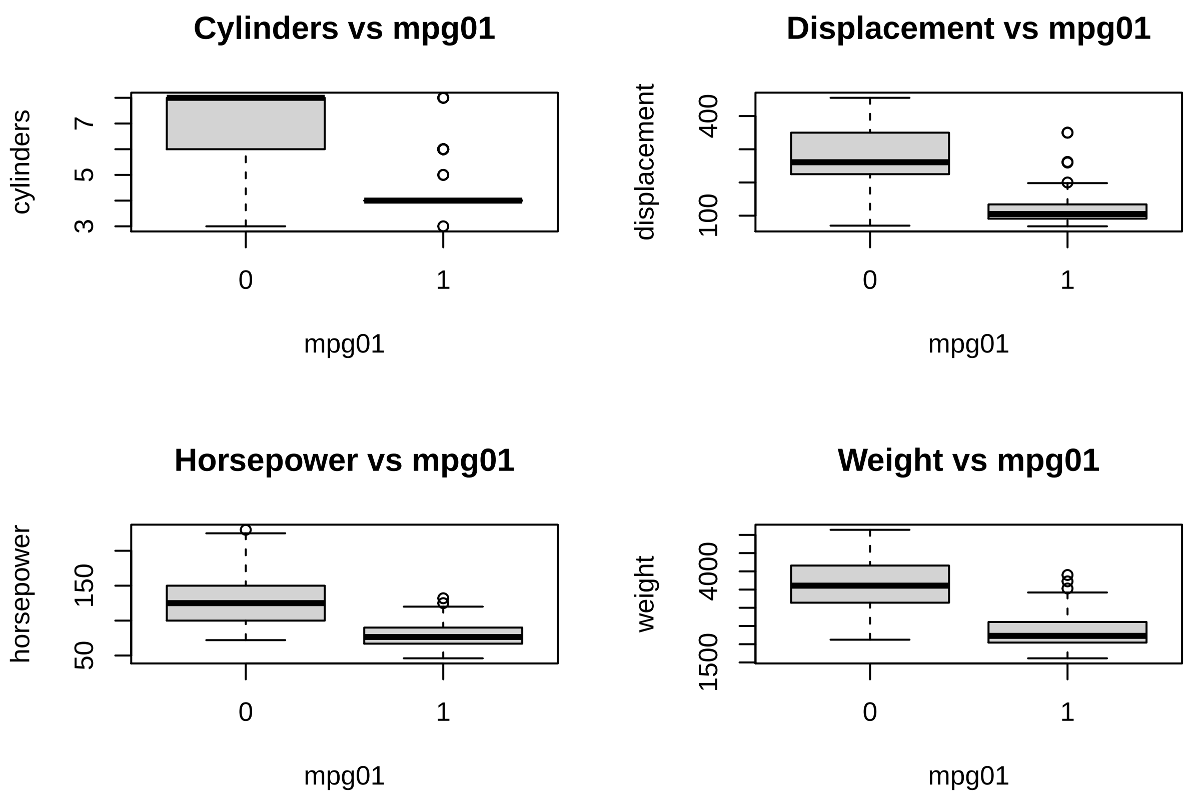
```r
ggcorr(auto_t, label = TRUE, palette = "RdBu")
```




```r
#top four: Weight, Horsepower, displacement, cylinders
```

Generate boxplots for those features

```r
par(mfrow=c(2,2))
boxplot(cylinders ~ mpg01, data = auto_t, main = "Cylinders vs mpg01")
boxplot(displacement ~ mpg01, data = auto_t, main = "Displacement vs mpg01")
boxplot(horsepower ~ mpg01, data = auto_t, main = "Horsepower vs mpg01")
boxplot(weight ~ mpg01, data = auto_t, main = "Weight vs mpg01")
```



c). Split the data into a training set and a test set.

```
set.seed(75)
train <- sample(seq(nrow(auto_t)), size = 0.75 * nrow(auto_t))
test <- -train
data_train <- auto_t[train,]
data_test <- auto_t[test,]
```

d). Perform LDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
lda_model <-
  lda(mpg01 ~ cylinders + displacement + horsepower + weight, data = data_train)

lda_pred <- predict(lda_model, data_test)
lda_class <- lda_pred$class

test_error_lda = mean(lda_class != data_test$mpg01)
test_error_lda
```

```
## [1] 0.1122449
```

e). Perform QDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b).What is the test error of the model obtained?

```
qda_model <-
  qda(mpg01 ~ cylinders + displacement + horsepower + weight, data = data_train)

qda_pred <- predict(qda_model, data_test)
qda_class <- qda_pred$class

test_error_qda = mean(qda_class != data_test$mpg01)
test_error_qda
```

```
## [1] 0.122449
```

f). Perform logistic regression on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

```
glm_model <-
  glm(mpg01 ~ cylinders + displacement + horsepower + weight,
      data = data_train, family = binomial)
glm_pred <- round(predict(glm_model, data_test, type = "response"))

test_error_glm <- mean(glm_pred != data_test$mpg01)
test_error_glm
```

```
## [1] 0.09183673
```

g). Perform KNN on the training data, with several values of K, in order to predict mpg01. Use only the variables that seemed most associated with mpg01 in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

```
vars <- c("cylinders", "displacement", "horsepower", "weight")
scaled_data = scale(auto_t) #Scale the data

#Generate new train and test indexes
set.seed(1234)
new_train <- sample(1:nrow(auto_t), 392 * 0.75, rep = FALSE)
new_test <- -new_train

#Selected only the best data for 4 variables
training_data_scaled = scaled_data[new_train, vars]
#Selected only the best data for 4 variables
testing_data_scaled = scaled_data[new_test, vars]
train_mpg01 <- auto_t$mpg01[new_train] #Save the mpg01 values used for training
test_mpg01 <- auto_t$mpg01[new_test] #Save the mpg01 values used for training

error_list <- NULL
knn_pred <- NULL
#Create a for loop
for (i in 1:nrow(testing_data_scaled)) {
  set.seed(5678)
  knn_pred <- knn(training_data_scaled, testing_data_scaled, train_mpg01, k = i)
  error_list[i] <- mean(test_mpg01 != knn_pred)
}
```

```r
min_error <- min(error_list)
cat("The min value for knn error is: ", min_error, " for k = ", which(error_list==min_error))
```

```
## The min value for knn error is:  0.09183673  for k =  6
```

#Problem 4 Problem 9 Pg 299 ISLR This question uses the variables dis(the weighted mean of distances to five Boston employment centers) and nox (nitrogen oxides concentration in parts per 10 million) from the Boston data. We will treat dis as the predictor and nox as the response.

a). Use the poly() function to fit a cubic polynomial regression to predict nox using dis. Report the regression output, and plot the resulting data and polynomial fits.

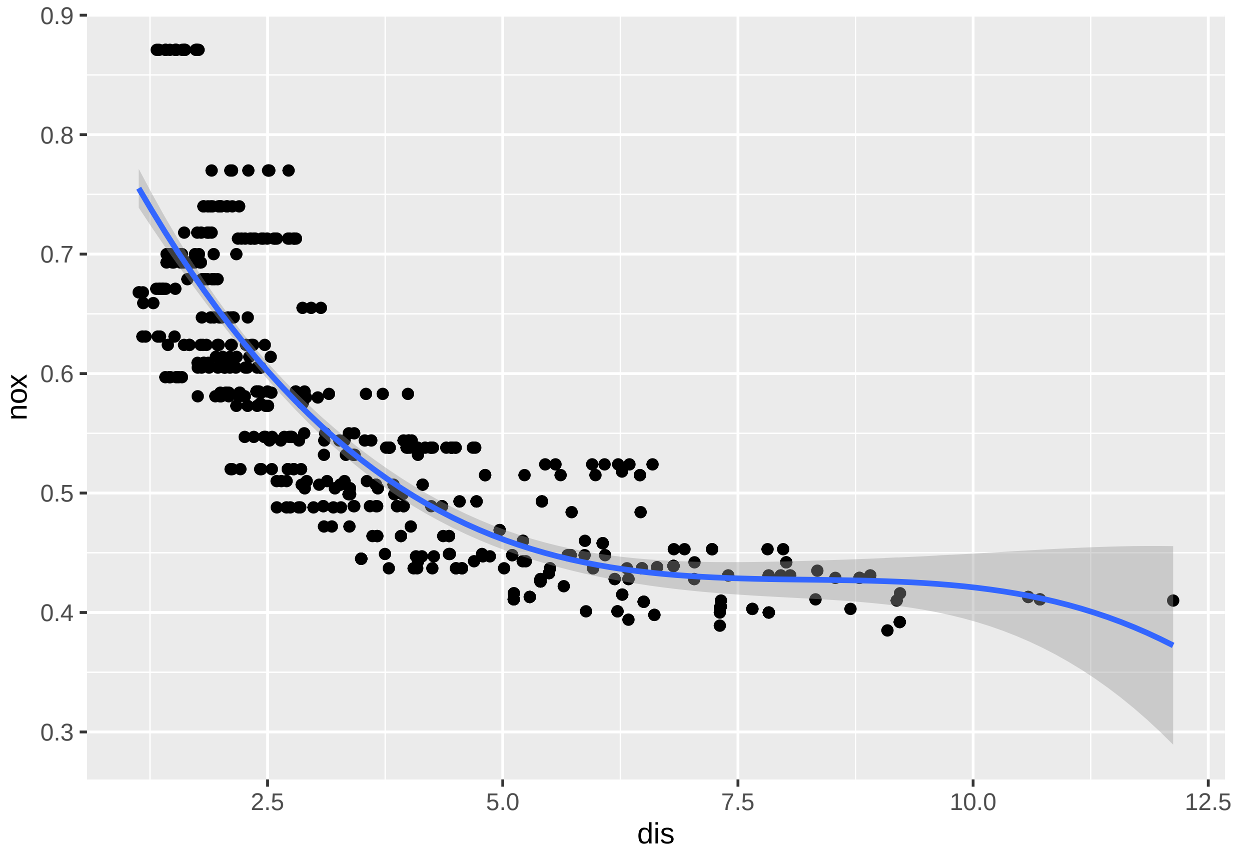
Create tidy() of lm boston

```r
boston_t <- Boston %>% as_tibble()
lm_boston <- lm(nox ~ poly(dis, 3), data = boston_t)
tidy_lm_boston <- tidy(lm_boston)
tidy_lm_boston
```

```
## # A tibble: 4 x 5
##   term          estimate std.error statistic   p.value
##   <chr>            <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)      0.555   0.00276    201.   0.
## 2 poly(dis, 3)1   -2.00    0.0621     -32.3  1.60e-124
## 3 poly(dis, 3)2    0.856   0.0621      13.8  6.13e- 37
## 4 poly(dis, 3)3   -0.318   0.0621      -5.12 4.27e-  7
```

Generate the ggplot

```r
ggplot(data = boston_t, mapping = aes(x = dis, y = nox)) +
  geom_point() +
  stat_smooth(method = "lm", formula = y ~ poly(x, 3))
```



b). Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

```
rss_list <- rep(NA, 10)
i = 0
for (i in seq_len(10)) {
  lm_boston <- lm(nox ~ poly(dis, i), data = boston_t)
  rss_list[i] = sum(lm_boston$residuals^2)
}
print("Residuals:")
```

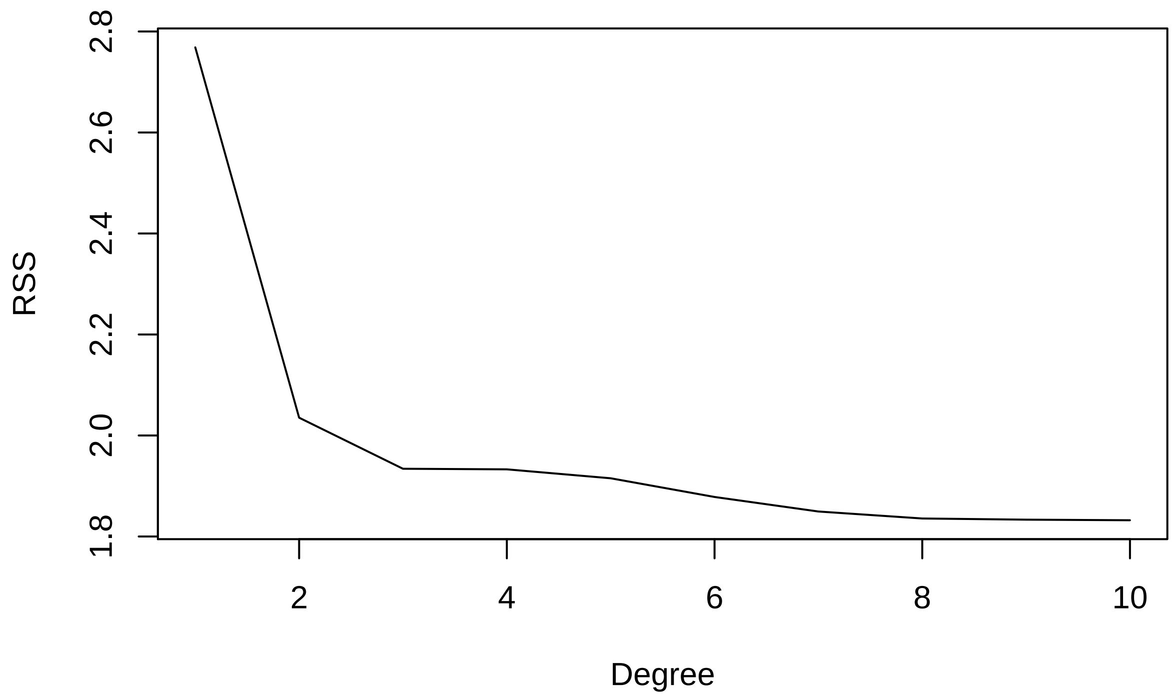
```
## [1] "Residuals:"
```

```
rss_list
```

```
##  [1] 2.768563 2.035262 1.934107 1.932981 1.915290 1.878257 1.849484 1.835630
##  [9] 1.833331 1.832171
```

```
par(mfrow=c(1,1))

plot(seq_len(10), rss_list, xlab = "Degree", ylab = "RSS", type = "l")
```



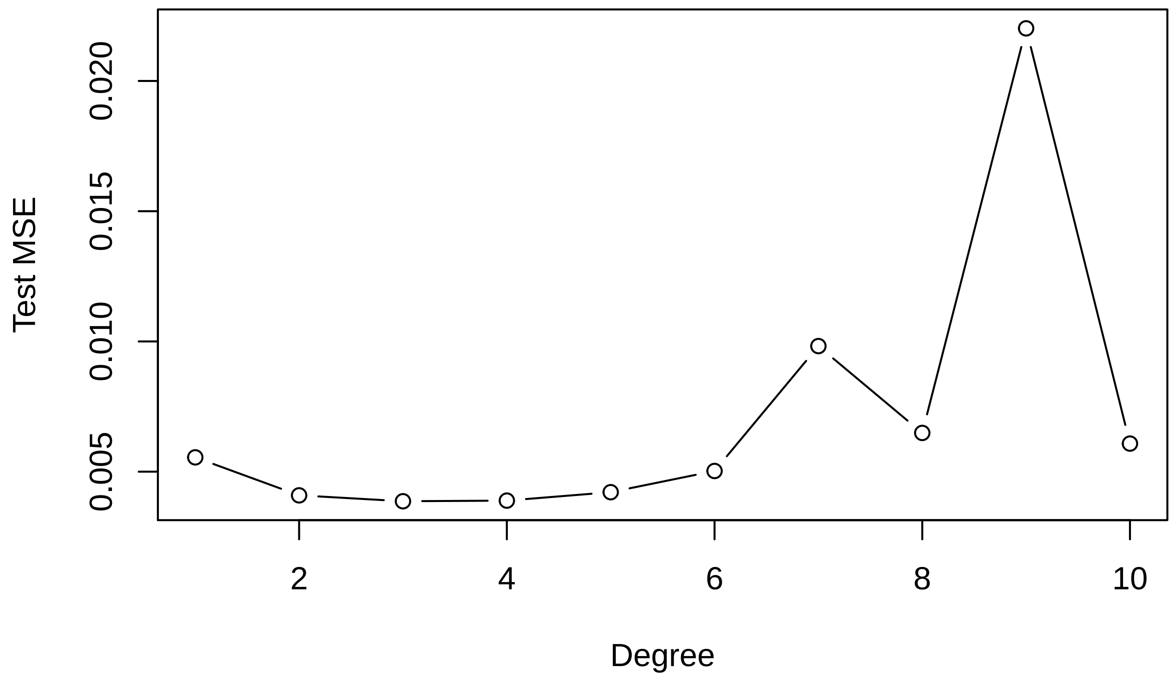
c). Perform cross-validation or another approach to select the optimal degree for the polynomial, and explain your results.

```
deltas <- rep(NA, 10)
for (i in seq_len(10)) {
  fit <- glm(nox ~ poly(dis, i), data = boston_t)
  deltas[i] <- cv.glm(Boston, fit, K = 10)$delta[1]
}
cat("deltas:", deltas, "\n")
```

```
## deltas: 0.005549782 0.004091567 0.003865234 0.003889036 0.004212462 0.005026809 0.00982141 0.0064876
```

In this case, the delta values decrease from 1 to 4, and then rapidly increase until Degree 8, with a slight drop at 9 followed by an ever steeper rise at 10, leading to the conclusion that degree=4 is the optimal since it minimizes cv-error

```
plot(1:10, deltas, type="b", xlab="Degree", ylab="Test MSE")
```



d). Use the bs() function to fit a regression spline to predict nox using dis. Report the output for the fit using four degrees of freedom. How did you choose the knots? Plot the resulting fit.

```
lm_spline <- lm(nox ~ bs(dis, df = 4), data = boston_t)
summary(lm_spline)
```

```
##
## Call:
## lm(formula = nox ~ bs(dis, df = 4), data = boston_t)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.124622 -0.039259 -0.008514  0.020850  0.193891
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.73447    0.01460  50.306  < 2e-16 ***
## bs(dis, df = 4)1 -0.05810    0.02186  -2.658  0.00812 **
## bs(dis, df = 4)2 -0.46356    0.02366 -19.596  < 2e-16 ***
## bs(dis, df = 4)3 -0.19979    0.04311  -4.634 4.58e-06 ***
## bs(dis, df = 4)4 -0.38881    0.04551  -8.544  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06195 on 501 degrees of freedom
## Multiple R-squared:  0.7164, Adjusted R-squared:  0.7142
## F-statistic: 316.5 on 4 and 501 DF,  p-value: < 2.2e-16
```

Display the attributes of the spline

```
attr(bs(boston_t$dis, df = 4), "knots")
```

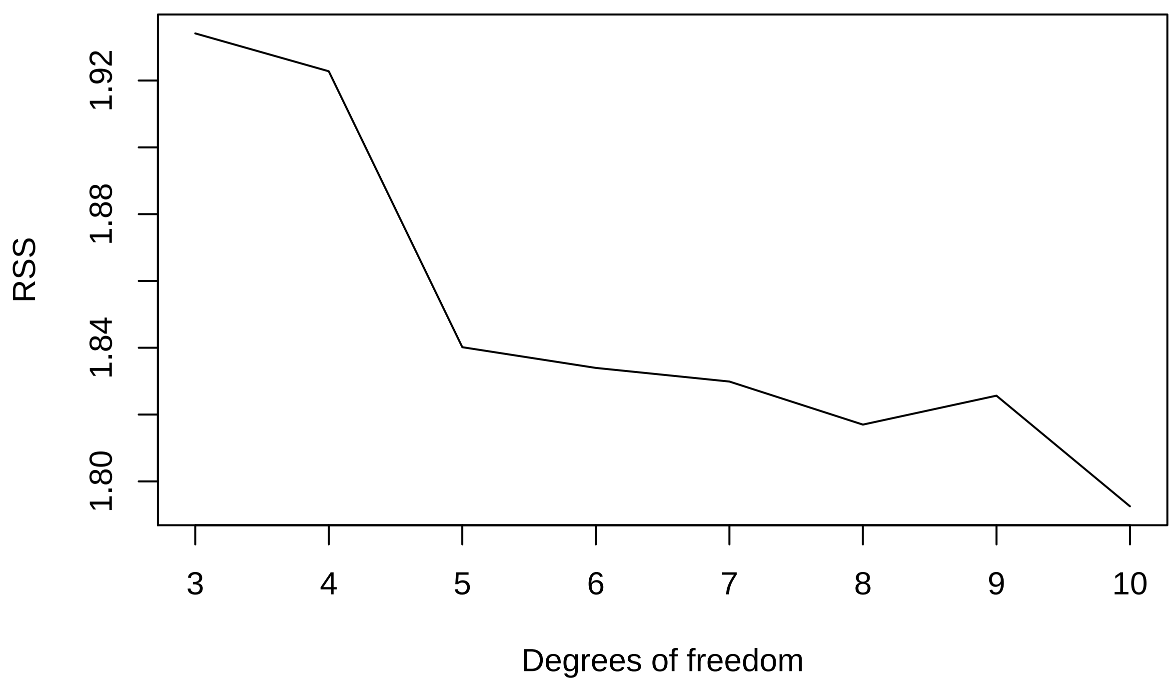
```
##     50%
## 3.20745
```

Based on the attributes, we only choose 1 knot. As a result, we can choose uniform quantiles of the feature

**Calculate the residuals

```
rss_list <- rep(NA, 10)
for (i in 3:10) {
  fit <- lm(nox ~ bs(dis, df = i), data = boston_t)
  rss_list[i] <- sum(fit$residuals^2)
}

plot(3:10, rss_list[3:10], xlab = "Degrees of freedom", ylab = "RSS", type = "l")
```



f). Perform cross-validation or another approach in order to select the best degrees of freedom for a regression spline on this data. Describe your results **Based on the plot, the best degree of freedom is 8**

```r
deltas_2 <- rep(NA, 10)
for (i in 3:10) {
  glm_spline <- glm(nox ~ (bs(dis, df = i)), data = boston_t)
  deltas_2[i] <- cv.glm(Boston, glm_spline, K = 10)$delta[1]
}
```

```
## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.137, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.137, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.1296, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = numeric(0), Boundary.knots = c(1.1296, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.1423), Boundary.knots =
## c(1.1296, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.1423), Boundary.knots =
## c(1.1296, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases

## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.2157), Boundary.knots =
## c(1.137, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.2157), Boundary.knots =
## c(1.137, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.3817, `66.66667%` =
## 4.239: some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.3817, `66.66667%` =
## 4.239: some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.354, `66.66667%`
## = 4.25576666666667: some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.354, `66.66667%`
## = 4.25576666666667: some 'x' values beyond boundary knots may cause ill-
## conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.08285, `50%` = 3.1025, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.08285, `50%` = 3.1025, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.10525, `50%` = 3.1323, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.10525, `50%` = 3.1323, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.97036, `40%` = 2.62334, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.97036, `40%` = 2.62334, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.9669, `40%` = 2.7147, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.9669, `40%` = 2.7147, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.82231666666667, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.82231666666667, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.85586666666667, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.85586666666667, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.7912, `28.57143%` =
## 2.1705, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.7912, `28.57143%` =
## 2.1705, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.78741428571429, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.78741428571429, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.75675, `25%` = 2.10035, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.75675, `25%` = 2.10035, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.7550125, `25%` =
## 2.09705, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.7550125, `25%` =
## 2.09705, : some 'x' values beyond boundary knots may cause ill-conditioned bases
```

```r
plot(3:10, deltas_2[3:10], type="b", xlab="Degrees of Freedom", ylab="Test MSE")
```