

DS-6030 Homework Module 6

Tom Lever

07/03/2023

DS 6030 | Spring 2023 | University of Virginia

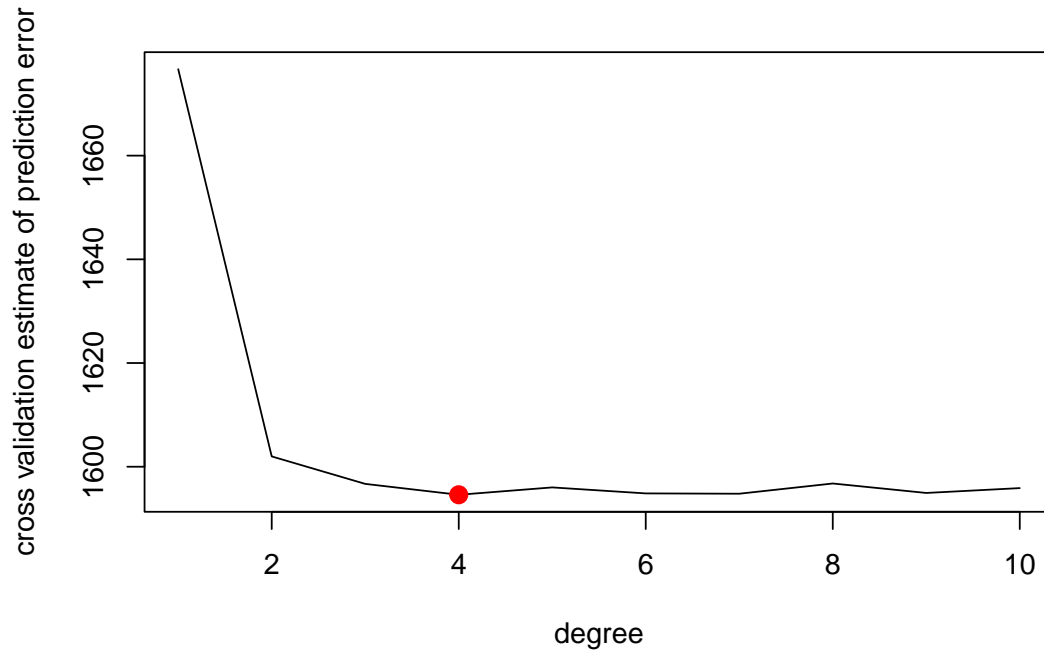
6. In this exercise, you will further analyze the `Wage` data set considered throughout this chapter.

- (a) Perform polynomial regression to predict wage using age. Use cross-validation to select the optimal degree d for the polynomial. What degree was chosen, and how does this compare to the results of hypothesis testing using ANOVA? Make a plot of the resulting polynomial fit to the data.

The optimal degree for a polynomial regression to predict wage vs. age is $d = 4$. According to the documentation for `anova`, “When given a sequence of objects, `anova` tests the models against one another in the order specified... It produces a table which tests whether the model terms [for a given model] are significant [in the context of the previous model].” Examining the column of p values in below table, a term of degree 1 is significant in the context of an intercept-only model, a term of degree 2 is significant in the context of a polynomial of degree 2, a term of degree 3 is significant in the context of a polynomial of degree 2, a term of degree 4 is approximately significant in the context of a polynomial of degree 4, but a term of degree 5 is insignificant in the context of a polynomial of degree 4. This interpretation accords with using cross-validation to select the optimal degree for the polynomial.

```
library(ISLR2)
set.seed(4)
range_of_degrees <- 1:10
number_of_degrees <- length(range_of_degrees)
cross_validation_estimates_of_prediction_errors <- rep(NA, number_of_degrees)
for (degree in range_of_degrees) {
  the_glm <- glm(wage ~ poly(age, degree), data = Wage)
  cross_validation_estimates_of_prediction_errors[degree] <-
    boot::cv.glm(Wage, the_glm, K = 10)$delta[1]
}
plot(
  x = range_of_degrees,
  y = cross_validation_estimates_of_prediction_errors,
  xlab = "degree",
  ylab = "cross validation estimate of prediction error",
  type = "l"
)
optimal_degree <-
  which.min(cross_validation_estimates_of_prediction_errors)
points(
  x = optimal_degree,
  y = cross_validation_estimates_of_prediction_errors[optimal_degree],
  col = "red",
  cex = 2,
```

```
pch = 20
)
```



```
lm_0 <- lm(wage ~ 1, data = Wage)
lm_1 <- lm(wage ~ age, data = Wage)
lm_2 <- lm(wage ~ poly(age, 2), data = Wage)
lm_3 <- lm(wage ~ poly(age, 3), data = Wage)
lm_4 <- lm(wage ~ poly(age, 4), data = Wage)
lm_5 <- lm(wage ~ poly(age, 5), data = Wage)
anova(lm_0, lm_1, lm_2, lm_3, lm_4, lm_5)
```

```
# Analysis of Variance Table
```

```
#
```

```
# Model 1: wage ~ 1
```

```
# Model 2: wage ~ age
```

```
# Model 3: wage ~ poly(age, 2)
```

```
# Model 4: wage ~ poly(age, 3)
```

```
# Model 5: wage ~ poly(age, 4)
```

```
# Model 6: wage ~ poly(age, 5)
```

#	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
# 1	2999	5222086				
# 2	2998	5022216	1	199870	125.4443	< 2.2e-16 ***
# 3	2997	4793430	1	228786	143.5931	< 2.2e-16 ***
# 4	2996	4777674	1	15756	9.8888	0.001679 **
# 5	2995	4771604	1	6070	3.8098	0.051046 .
# 6	2994	4770322	1	1283	0.8050	0.369682

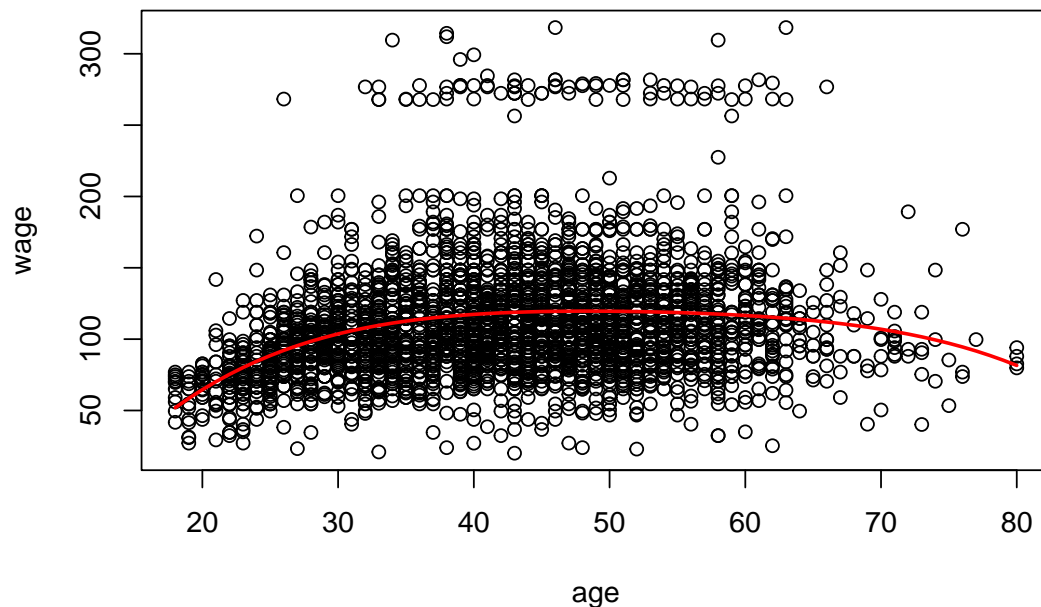
```
# ---
```

```
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```

plot(wage ~ age, data = Wage)
minimum_age <- min(Wage$age)
maximum_age <- max(Wage$age)
sequence_of_ages <- seq(from = minimum_age, to = maximum_age)
list_with_age <- list(age = sequence_of_ages)
vector_of_predicted_wages <- predict(object = lm_4, newdata = list_with_age)
lines(sequence_of_ages, vector_of_predicted_wages, col = "red", lwd = 2)

```



- (b) Fit a step function to predict wage using age, and perform cross-validation to choose the optimal number of cuts. Make a plot of the fit obtained.

A cross-validated estimate of prediction error is minimum for 8 intervals.

```

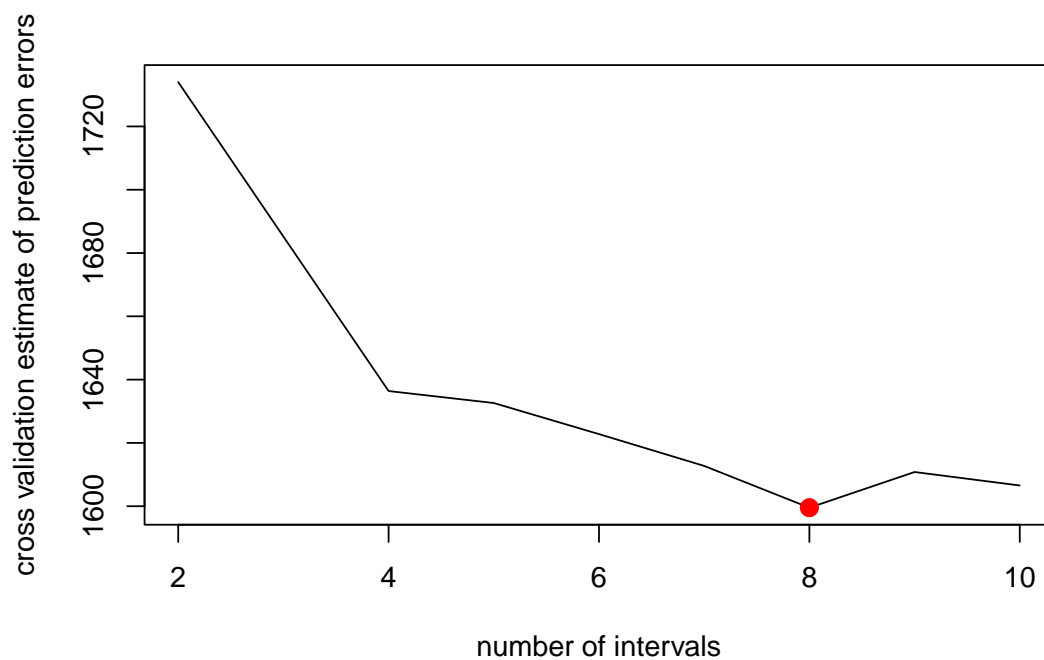
range_of_numbers_of_intervals <- 2:10
number_of_numbers_of_intervals <- length(range_of_numbers_of_intervals)
cross_validation_estimates_of_prediction_errors <-
  rep(NA, number_of_numbers_of_intervals)
for (number_of_intervals in range_of_numbers_of_intervals) {
  Wage$interval <- cut(Wage$age, number_of_intervals)
  the_glm <- glm(wage ~ interval, data = Wage)
  cross_validation_estimates_of_prediction_errors[number_of_intervals - 1] <-
    boot::cv.glm(Wage, the_glm, K = 10)$delta[1]
}
plot(
  x = range_of_numbers_of_intervals,
  y = cross_validation_estimates_of_prediction_errors,
  xlab = "number of intervals",
  ylab = "cross validation estimate of prediction errors",
  type = "l"
)

```

```

optimal_number_of_intervals <-
  which.min(cross_validation_estimates_of_prediction_errors)
points(
  x = optimal_number_of_intervals + 1,
  y = cross_validation_estimates_of_prediction_errors[
    optimal_number_of_intervals
  ],
  col = "red",
  cex = 2,
  pch = 20
)

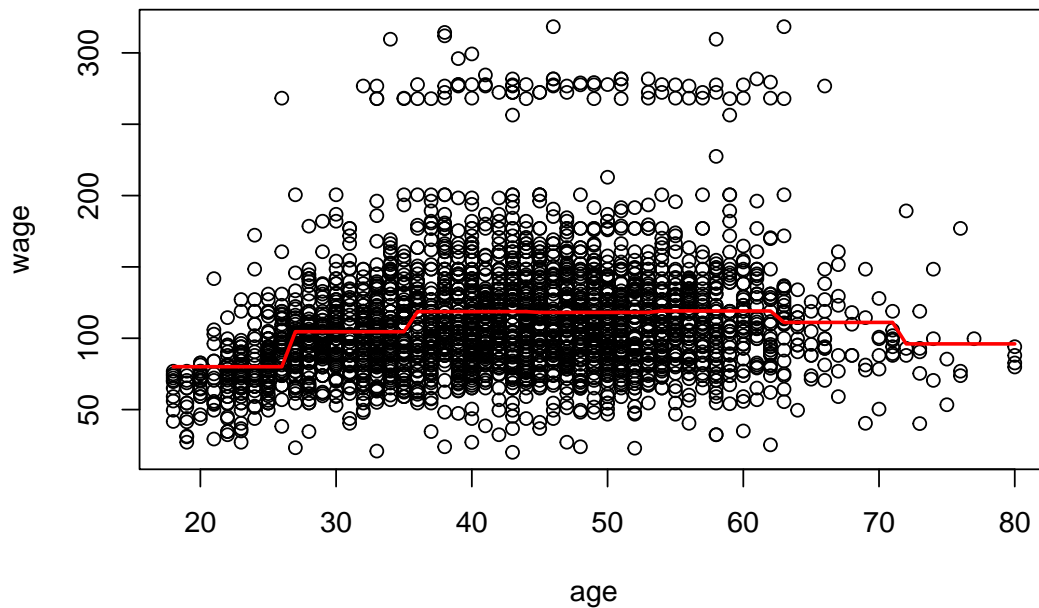
```



```

the_glm <- glm(wage ~ cut(age, optimal_number_of_intervals), data = Wage)
plot(wage ~ age, data = Wage)
vector_of_predicted_wages <- predict(object = the_glm, newdata = list_with_age)
lines(sequence_of_ages, vector_of_predicted_wages, col = "red", lwd = 2)

```



7. 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.

Per the regression output, all coefficients are significant in the context of the multiple linear model.

```
library(ggplot2)
library(MASS)

#
# Attaching package: 'MASS'

# The following object is masked from 'package:ISLR2':
#
#   Boston

set.seed(1)
lm_3 <- lm(nox ~ poly(dis, 3), data = Boston)
summary(lm_3)

#
# Call:
# lm(formula = nox ~ poly(dis, 3), data = Boston)
#
# Residuals:
#      Min       1Q   Median       3Q      Max
# -0.121130 -0.040619 -0.009738  0.023385  0.194904
#
```

```

# Coefficients:
#               Estimate Std. Error t value Pr(>|t|)
# (Intercept)    0.554695   0.002759 201.021 < 2e-16 ***
# poly(dis, 3)1 -2.003096   0.062071 -32.271 < 2e-16 ***
# poly(dis, 3)2  0.856330   0.062071  13.796 < 2e-16 ***
# poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
# Residual standard error: 0.06207 on 502 degrees of freedom
# Multiple R-squared:  0.7148, Adjusted R-squared:  0.7131
# F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16

minimum_weighted_mean_of_distances <- min(Boston$dis)
maximum_weighted_mean_of_distances <- max(Boston$dis)
sequence_of_weighted_means_of_distances <- seq(
  from = minimum_weighted_mean_of_distances,
  to = maximum_weighted_mean_of_distances,
  by = 0.1
)
list_with_dis <- list(dis = sequence_of_weighted_means_of_distances)
data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations <-
  data.frame(
    weighted_mean_of_distances =
      sequence_of_weighted_means_of_distances
  )
for (degree in range_of_degrees) {
  the_lm <- lm(nox ~ poly(dis, degree), data = Boston)
  vector_of_predicted_nitrogen_oxide_concentrations <- predict(
    object = the_lm,
    list_with_dis
  )
  column_label <- paste("NOx_concentration_predicted_by_polynomial_of_degree_", degree, sep = " ")
  data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations[,
    column_label
  ] <- vector_of_predicted_nitrogen_oxide_concentrations
}

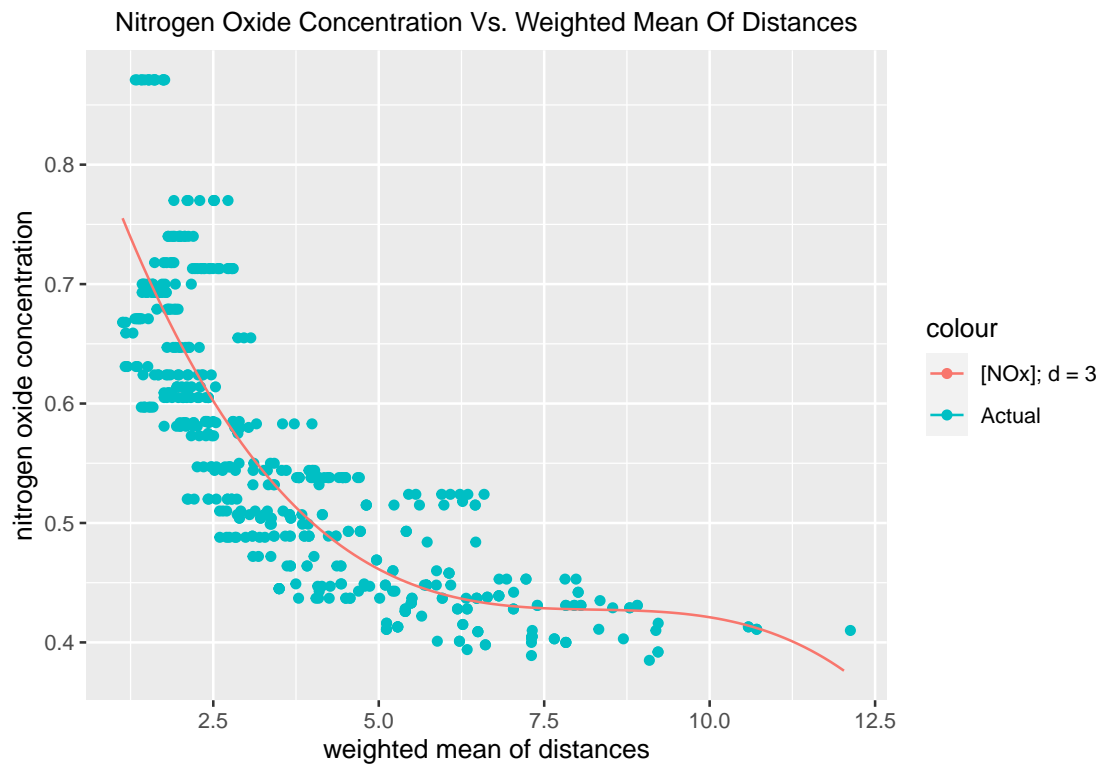
ggplot() +
  geom_point(
    data = Boston,
    mapping = aes(
      x = dis,
      y = nox,
      color = "Actual"
    )
  ) +
  geom_line(
    data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
    mapping = aes(
      x = weighted_mean_of_distances,
      y = NOx_concentration_predicted_by_polynomial_of_degree_3,
      color = "[NOx]"; d = 3"
    )
  )

```

```

) +
labs(
  x = "weighted mean of distances",
  y = "nitrogen oxide concentration",
  title = "Nitrogen Oxide Concentration Vs. Weighted Mean Of Distances"
) +
theme(
  plot.title = element_text(hjust = 0.5, size = 11),
)

```



- (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.

```

ggplot() +
  geom_point(
    data = Boston,
    mapping = aes(
      x = dis,
      y = nox,
      color = "[NOx]"
    )
  ) +
  geom_line(
    data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
    mapping = aes(
      x = weighted_mean_of_distances,
      y = NOx_concentration_predicted_by_polynomial_of_degree_1,
      color = "[NOx]; d = 1"
    )
  )

```

```

) +
geom_line(
  data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
  mapping = aes(
    x = weighted_mean_of_distances,
    y = NOx_concentration_predicted_by_polynomial_of_degree_2,
    color = "[NOx]"; d = 2"
  )
) +
geom_line(
  data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
  mapping = aes(
    x = weighted_mean_of_distances,
    y = NOx_concentration_predicted_by_polynomial_of_degree_3,
    color = "[NOx]"; d = 3"
  )
) +
geom_line(
  data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
  mapping = aes(
    x = weighted_mean_of_distances,
    y = NOx_concentration_predicted_by_polynomial_of_degree_4,
    color = "[NOx]"; d = 4"
  )
) +
geom_line(
  data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
  mapping = aes(
    x = weighted_mean_of_distances,
    y = NOx_concentration_predicted_by_polynomial_of_degree_5,
    color = "[NOx]"; d = 5"
  )
) +
geom_line(
  data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
  mapping = aes(
    x = weighted_mean_of_distances,
    y = NOx_concentration_predicted_by_polynomial_of_degree_6,
    color = "[NOx]"; d = 6"
  )
) +
geom_line(
  data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
  mapping = aes(
    x = weighted_mean_of_distances,
    y = NOx_concentration_predicted_by_polynomial_of_degree_7,
    color = "[NOx]"; d = 7"
  )
) +
geom_line(
  data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
  mapping = aes(
    x = weighted_mean_of_distances,

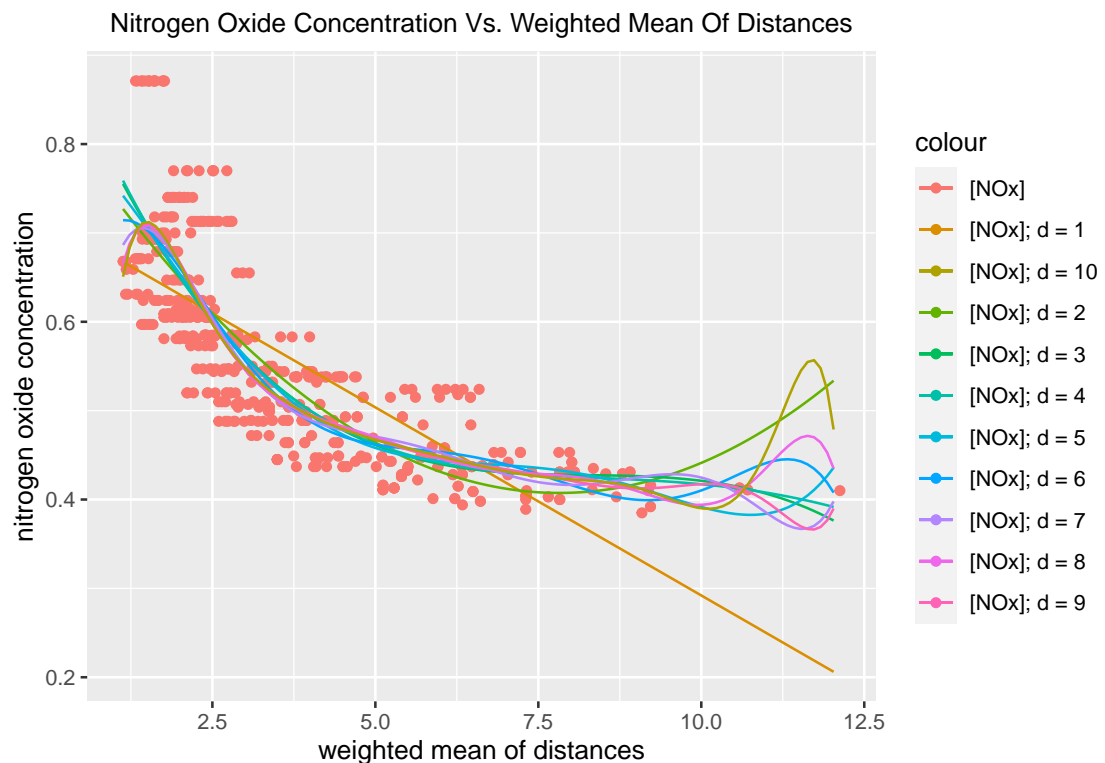
```



```

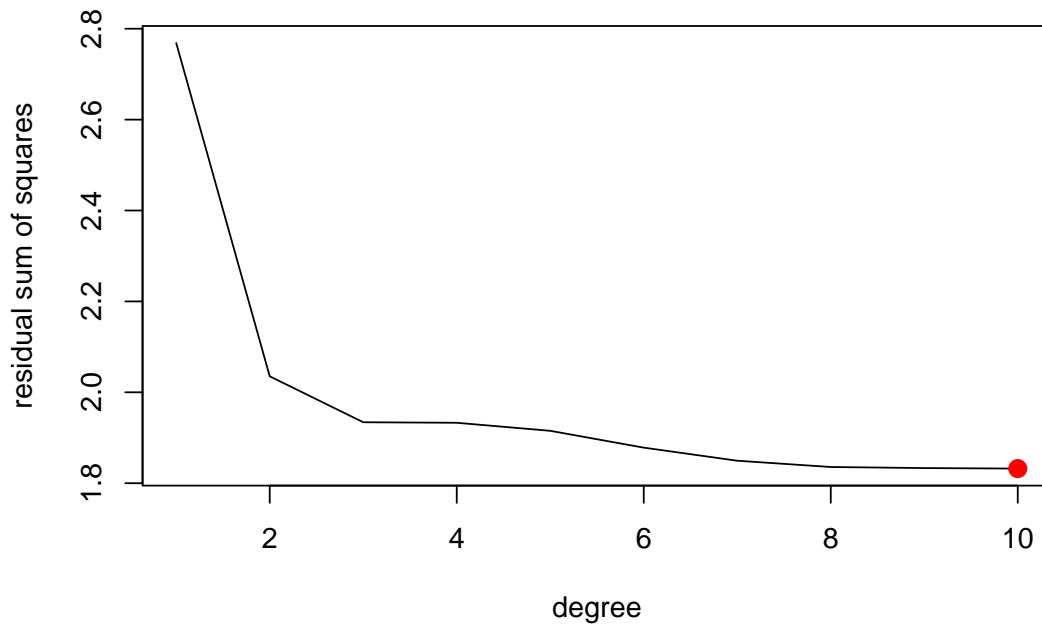
        y = NOx_concentration_predicted_by_polynomial_of_degree_8,
        color = "[NOx]; d = 8"
    )
) +
geom_line(
    data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
    mapping = aes(
        x = weighted_mean_of_distances,
        y = NOx_concentration_predicted_by_polynomial_of_degree_9,
        color = "[NOx]; d = 9"
    )
) +
geom_line(
    data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
    mapping = aes(
        x = weighted_mean_of_distances,
        y = NOx_concentration_predicted_by_polynomial_of_degree_10,
        color = "[NOx]; d = 10"
    )
) +
labs(
    x = "weighted mean of distances",
    y = "nitrogen oxide concentration",
    title = "Nitrogen Oxide Concentration Vs. Weighted Mean Of Distances"
) +
theme(
    plot.title = element_text(hjust = 0.5, size = 11),
)

```



Per a plot of residual sum of squares vs. degree of polynomial, residual sum of squares decreases with degree of polynomial to a minimum for degree 10.

```
range_of_degrees <- 1:10
number_of_degrees <- length(range_of_degrees)
vector_of_residual_sums_of_squares <- rep(NA, number_of_degrees)
for (degree in range_of_degrees) {
  the_lm <- lm(nox ~ poly(dis, degree), data = Boston)
  vector_of_residual_sums_of_squares[degree] <- sum(the_lm$residuals^2)
}
plot(
  x = range_of_degrees,
  y = vector_of_residual_sums_of_squares,
  xlab = "degree",
  ylab = "residual sum of squares",
  type = "l"
)
optimal_degree <-
  which.min(vector_of_residual_sums_of_squares)
points(
  x = optimal_degree,
  y = vector_of_residual_sums_of_squares[optimal_degree],
  col = "red",
  cex = 2,
  pch = 20
)
```



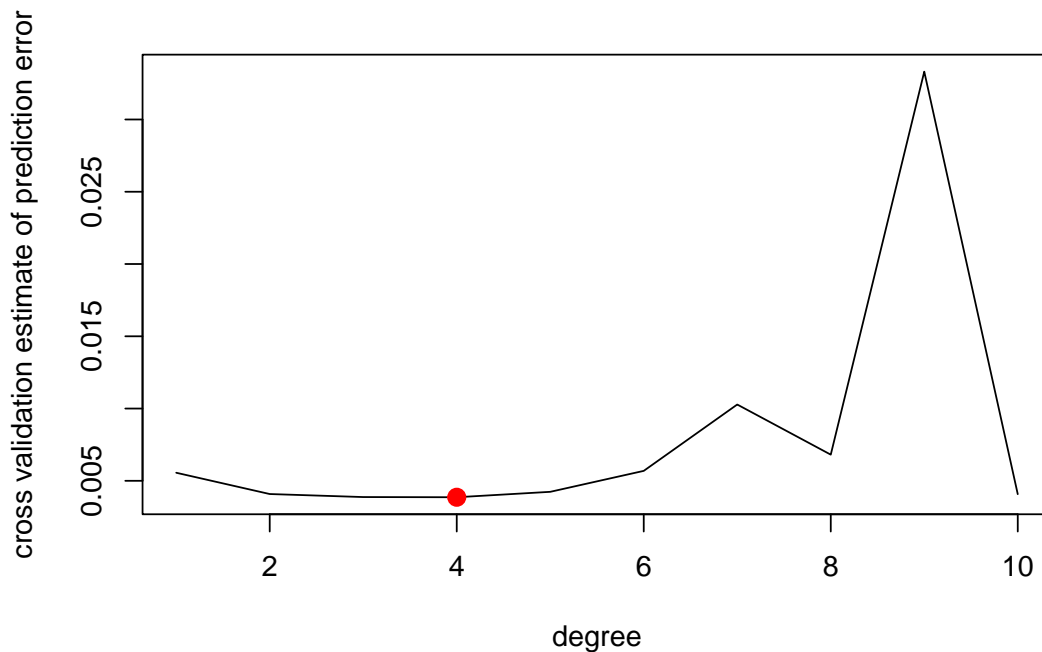
- (c) Perform cross-validation or another approach to select the optimal degree for the polynomial, and explain your results.

The minimum cross-validation estimate of prediction error occurs for degree 4.

```

cross_validation_estimates_of_prediction_errors <- rep(NA, number_of_degrees)
for (degree in range_of_degrees) {
  the_glm <- glm(nox ~ poly(dis, degree), data = Boston)
  cross_validation_estimates_of_prediction_errors[degree] <-
    boot::cv.glm(Boston, the_glm, K = 10)$delta[1]
}
plot(
  range_of_degrees,
  cross_validation_estimates_of_prediction_errors,
  xlab = "degree",
  ylab = "cross validation estimate of prediction error",
  type = "l"
)
optimal_degree <- which.min(cross_validation_estimates_of_prediction_errors)
points(
  x = optimal_degree,
  y = cross_validation_estimates_of_prediction_errors[optimal_degree],
  col = "red",
  cex = 2,
  pch = 20
)

```



- (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. The R interpreter determines knots automatically.

```

library(ggplot2)
library(MASS)
set.seed(1)

```

```

lm_4 <- lm(nox ~ splines::bs(dis, df = 4), data = Boston)
summary(lm_4)

#
# Call:
# lm(formula = nox ~ splines::bs(dis, df = 4), data = Boston)
#
# 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 ***
# splines::bs(dis, df = 4)1 -0.05810    0.02186   -2.658  0.00812 **
# splines::bs(dis, df = 4)2 -0.46356    0.02366  -19.596 < 2e-16 ***
# splines::bs(dis, df = 4)3 -0.19979    0.04311   -4.634  4.58e-06 ***
# splines::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

data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations <-
  data.frame(
    weighted_mean_of_distances =
      sequence_of_weighted_means_of_distances
  )
numbers_of_degrees_of_freedom <- 3:10
for (number_of_degrees_of_freedom in numbers_of_degrees_of_freedom) {
  the_lm <- lm(nox ~ splines::bs(dis, df = number_of_degrees_of_freedom), data = Boston)
  vector_of_predicted_nitrogen_oxide_concentrations <- predict(
    object = the_lm,
    list_with_dis
  )
  column_label <- paste(
    "NOx_concentration_predicted_by_B_spline_with_DF_",
    number_of_degrees_of_freedom,
    sep = ""
  )
  data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations[
    ,
    column_label
  ] <- vector_of_predicted_nitrogen_oxide_concentrations
}

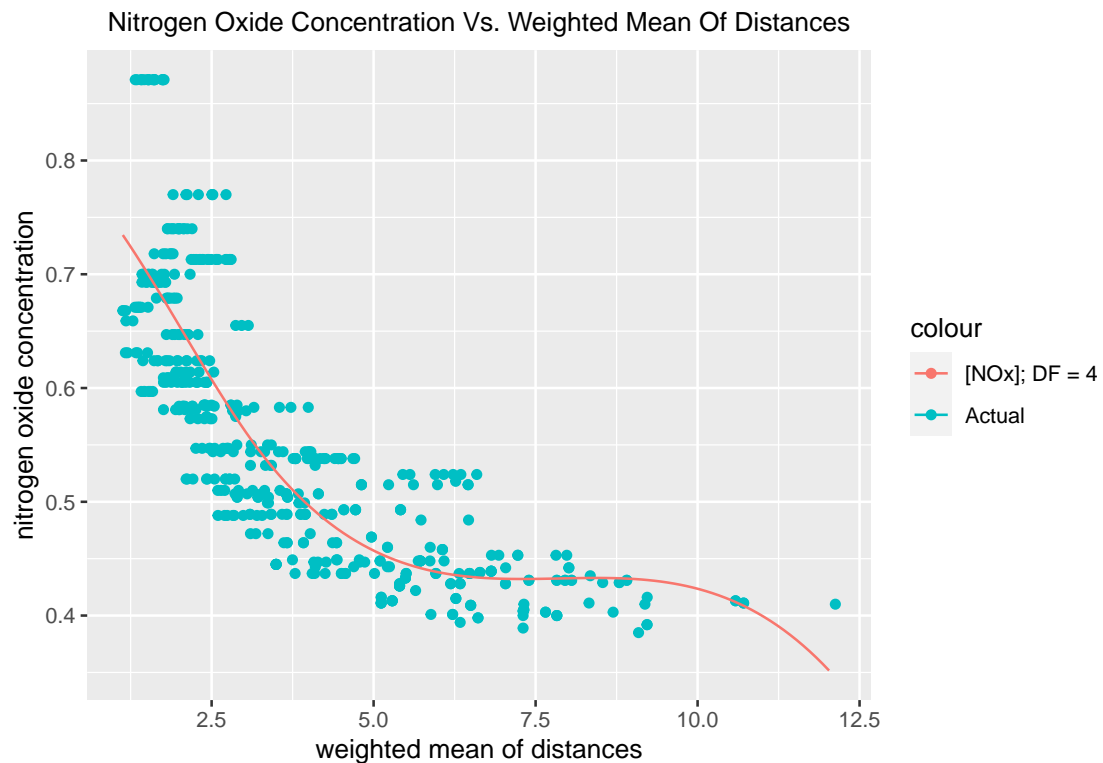
ggplot() +
  geom_point(
    data = Boston,
    mapping = aes(
      x = dis,
      y = nox,
      color = "Actual"
    )
  )

```

```

    )
  ) +
  geom_line(
    data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
    mapping = aes(
      x = weighted_mean_of_distances,
      y = NOx_concentration_predicted_by_B_spline_with_DF_4,
      color = "[NOx]; DF = 4"
    )
  )
  ) +
  labs(
    x = "weighted mean of distances",
    y = "nitrogen oxide concentration",
    title = "Nitrogen Oxide Concentration Vs. Weighted Mean Of Distances"
  ) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 11),
  )
)

```



- (e) Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.

```

ggplot() +
  geom_point(
    data = Boston,
    mapping = aes(
      x = dis,
      y = nox,
      color = "Actual"
    )
  )

```

```

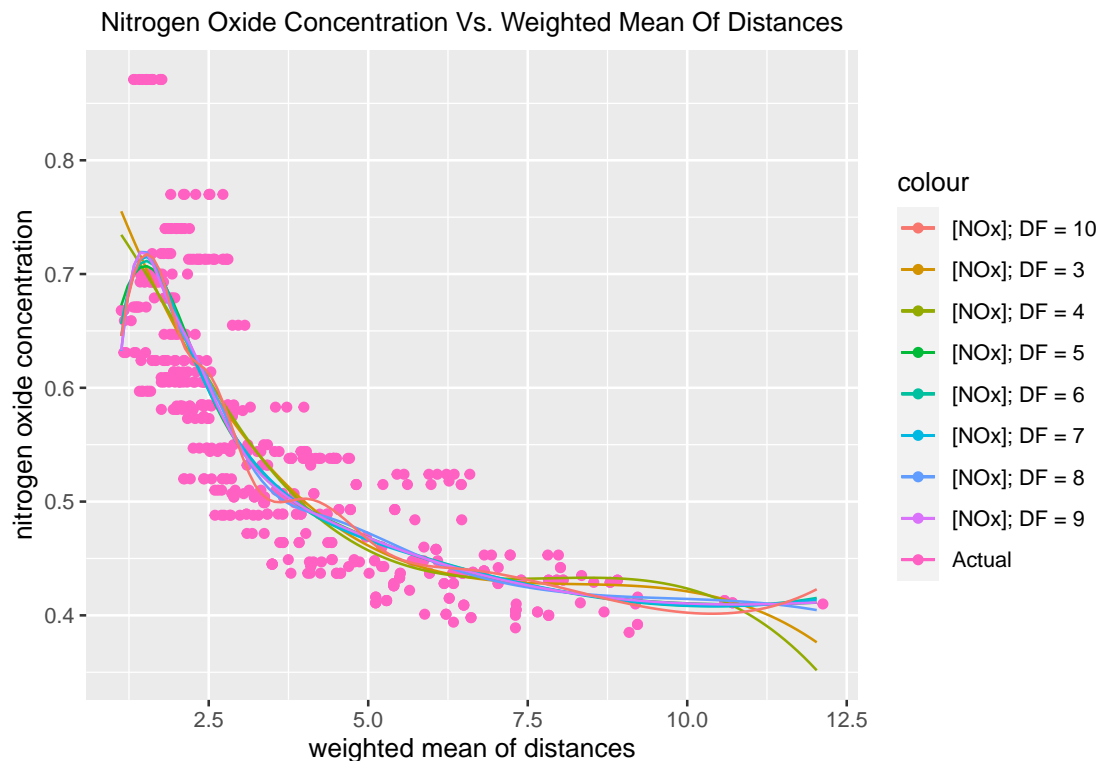
    )
  ) +
  geom_line(
    data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
    mapping = aes(
      x = weighted_mean_of_distances,
      y = NOx_concentration_predicted_by_B_spline_with_DF_3,
      color = "[NOx]; DF = 3"
    )
  ) +
  geom_line(
    data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
    mapping = aes(
      x = weighted_mean_of_distances,
      y = NOx_concentration_predicted_by_B_spline_with_DF_4,
      color = "[NOx]; DF = 4"
    )
  ) +
  geom_line(
    data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
    mapping = aes(
      x = weighted_mean_of_distances,
      y = NOx_concentration_predicted_by_B_spline_with_DF_5,
      color = "[NOx]; DF = 5"
    )
  ) +
  geom_line(
    data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
    mapping = aes(
      x = weighted_mean_of_distances,
      y = NOx_concentration_predicted_by_B_spline_with_DF_6,
      color = "[NOx]; DF = 6"
    )
  ) +
  geom_line(
    data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
    mapping = aes(
      x = weighted_mean_of_distances,
      y = NOx_concentration_predicted_by_B_spline_with_DF_7,
      color = "[NOx]; DF = 7"
    )
  ) +
  geom_line(
    data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
    mapping = aes(
      x = weighted_mean_of_distances,
      y = NOx_concentration_predicted_by_B_spline_with_DF_8,
      color = "[NOx]; DF = 8"
    )
  ) +
  geom_line(
    data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
    mapping = aes(

```

```

    x = weighted_mean_of_distances,
    y = NOx_concentration_predicted_by_B_spline_with_DF_9,
    color = "[NOx]; DF = 9"
  )
) +
geom_line(
  data = data_frame_of_weighted_means_of_distances_and_predicted_NOx_concentrations,
  mapping = aes(
    x = weighted_mean_of_distances,
    y = NOx_concentration_predicted_by_B_spline_with_DF_10,
    color = "[NOx]; DF = 10"
  )
) +
labs(
  x = "weighted mean of distances",
  y = "nitrogen oxide concentration",
  title = "Nitrogen Oxide Concentration Vs. Weighted Mean Of Distances"
) +
theme(
  plot.title = element_text(hjust = 0.5, size = 11),
)

```



Per a plot of residual sum of squares vs. number of degrees of freedom of B-spline, residual sum of squares decreases with number of degrees of freedom of B-spline to a minimum for number of degrees of freedom 10.

```

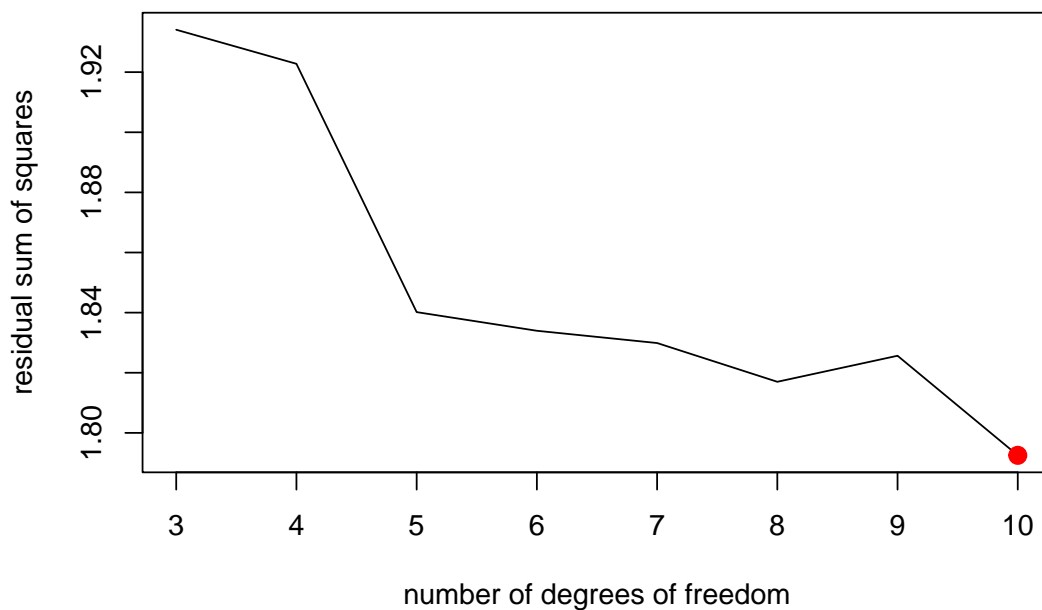
numbers_of_degrees_of_freedom <- 3:10
number_of_numbers_of_degrees_of_freedom <- length(numbers_of_degrees_of_freedom)
vector_of_residual_sums_of_squares <-

```

```

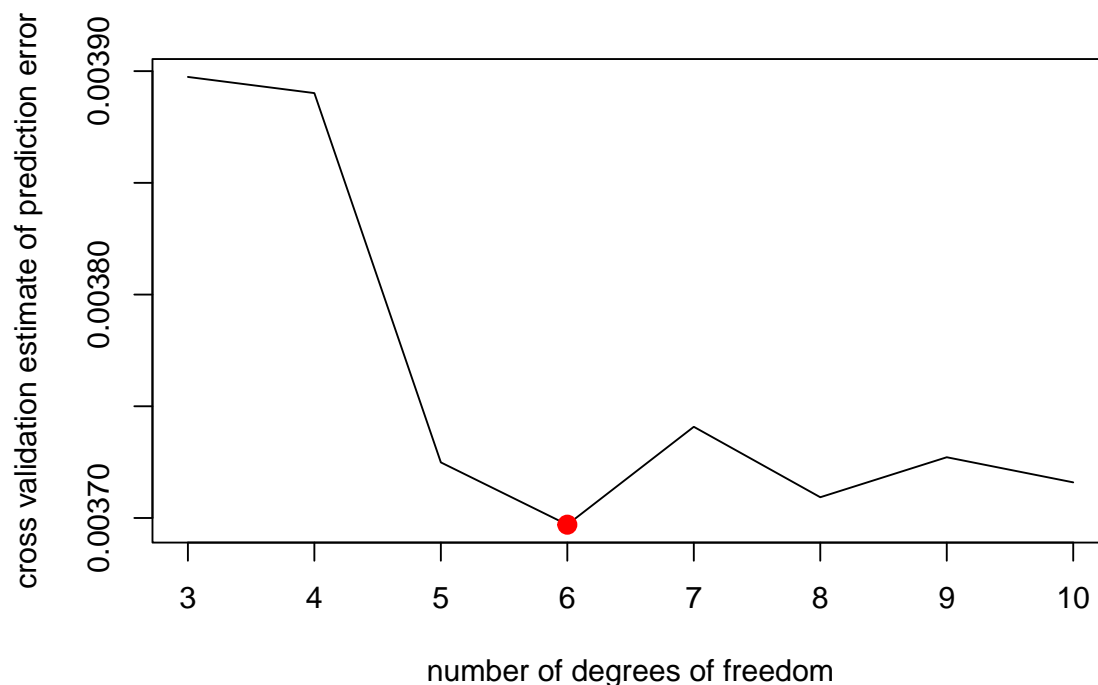
rep(NA, number_of_numbers_of_degrees_of_freedom)
for (number_of_degrees_of_freedom in numbers_of_degrees_of_freedom) {
  the_lm <- lm(
    nox ~ splines::bs(dis, df = number_of_degrees_of_freedom),
    data = Boston
  )
  vector_of_residual_sums_of_squares[number_of_degrees_of_freedom - 2] <-
    sum(the_lm$residuals^2)
}
plot(
  x = numbers_of_degrees_of_freedom,
  y = vector_of_residual_sums_of_squares,
  xlab = "number of degrees of freedom",
  ylab = "residual sum of squares",
  type = "l"
)
optimal_number_of_degrees_of_freedom <-
  which.min(vector_of_residual_sums_of_squares)
points(
  x = optimal_number_of_degrees_of_freedom + 2,
  y = vector_of_residual_sums_of_squares[
    optimal_number_of_degrees_of_freedom
  ],
  col = "red",
  cex = 2,
  pch = 20
)

```



- (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.

```
cross_validation_estimates_of_prediction_errors <-  
  rep(NA, number_of_numbers_of_degrees_of_freedom)  
for (number_of_degrees_of_freedom in numbers_of_degrees_of_freedom) {  
  the_glm <- glm(  
    nox ~ splines::bs(dis, df = number_of_degrees_of_freedom),  
    data = Boston  
  )  
  cross_validation_estimates_of_prediction_errors[number_of_degrees_of_freedom - 2] <-  
    boot::cv.glm(Boston, the_glm, K = 10)$delta[1]  
}  
plot(  
  numbers_of_degrees_of_freedom,  
  cross_validation_estimates_of_prediction_errors,  
  xlab = "number of degrees of freedom",  
  ylab = "cross validation estimate of prediction error",  
  type = "l"  
)  
optimal_degree <- which.min(cross_validation_estimates_of_prediction_errors)  
points(  
  x = optimal_degree + 2,  
  y = cross_validation_estimates_of_prediction_errors[optimal_degree],  
  col = "red",  
  cex = 2,  
  pch = 20  
)
```



10. This question relates to the College data set.

- (a) Split the data into a training set and a test set. Using out-of-state tuition as the response and the other variables as the predictors, perform forward stepwise selection on the training set in order to identify a satisfactory model that uses just a subset of the predictors.
- (b) Fit a GAM on the training data, using out-of-state tuition as the response and the features selected in the previous step as the predictors. Plot the results, and explain your findings.
- (c) Evaluate the model obtained on the test set, and explain the results obtained.
- (d) For which variables, if any, is there evidence of a non-linear relationship with the response?