chapter 07 hw

Fabiani Rafael

Conceptual Questions

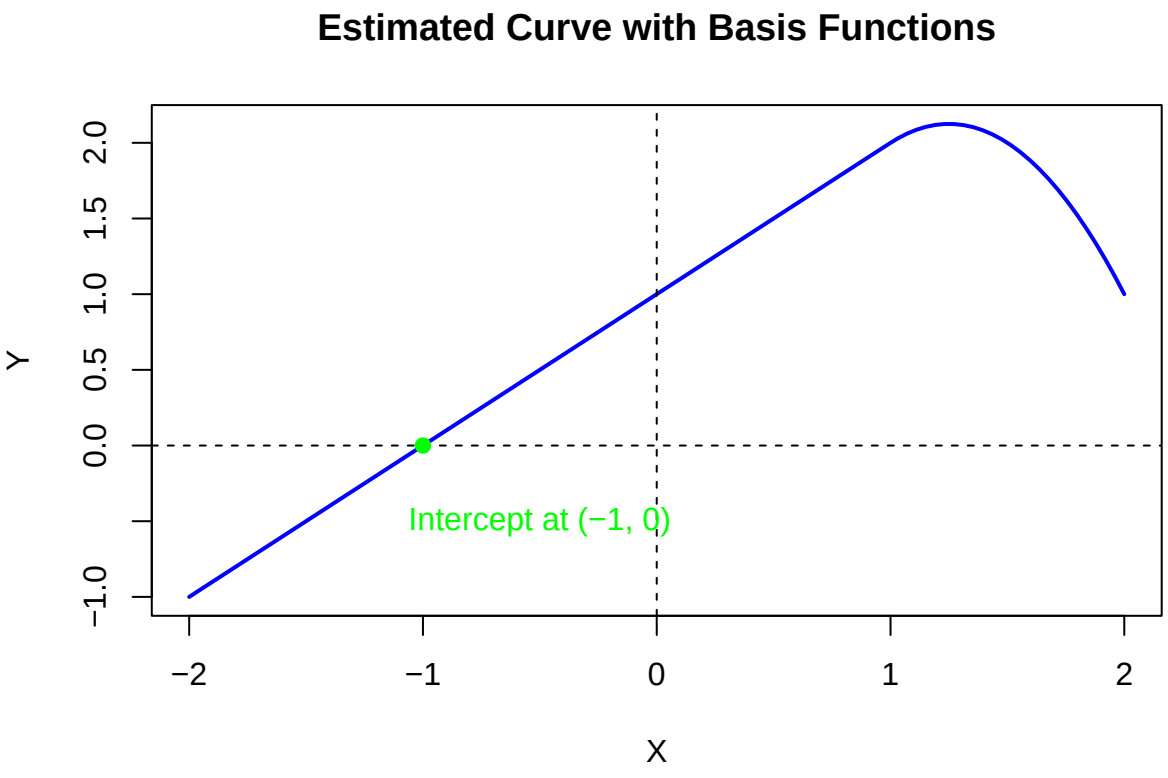
Exercise 3: Suppose we fit a curve with basis functions $b_1(X) = X$, $b_2(X) = (X-1)^2 I(X \geq 1)$. (Note that $I(X \geq 1)$ equals 1 for $X \geq 1$ and 0 otherwise) We fit the linear regression model

$$Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \epsilon,$$

and obtain coefficient estimates $\hat{\beta}_0 = 1, \hat{\beta}_1 = 1, \hat{\beta}_2 = -2$. Sketch the estimated curve between $X = -2$ and $X = 2$. Note the intercepts, slopes and other relevant information.

Using R to create the sketch we see that the slope is 1 for $X < 1$ and when $X \geq 1$ the eqaution $ Y = 1 + X - 2(X - 1)^2 = 1 + X -2(X^2 -2X + 1) = -2X^2 + 5X - 1$ has a slope $-4X + 5$. The intercept is at $(0, 1)$ and the function is continuous at $X = 1$.

```
X <- seq(-2, 2, length.out = 100)
b1 <- X
b2 <- (X - 1)^2 * (X >= 1)
Y <- 1 + b1 + (-2) * b2
plot(X, Y, type = "l", col = "blue", lwd = 2, xlab = "X", ylab = "Y",
     main = "Estimated Curve with Basis Functions")
abline(h = 0, col = "black", lty = 2)
abline(v = 0, col = "black", lty = 2)
points(-1, 0, col = "green", pch = 19)
text(-.5, -.5, "Intercept at (-1, 0)", col = "green")
```



Applied Questions

Exercise 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 crossvalidation 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.

```
data(Wage)
set.seed(488)

# cv for polynomial degrees 1-5
degrees <- 1:5
cv_errors <- numeric(length(degrees))

for (d in degrees) {
  #  model formula with degree d
  model_formula <- as.formula(paste("wage ~ poly(age,", d, ")"))

  #  10-fold cross-validation
  glm_model <- glm(model_formula, data = Wage)
  cv_results <- cv.glm(Wage, glm_model, K = 10)

  # Store cross-validation error (MSE)
  cv_errors[d] <- cv_results$delta[1]
}

# Identify optimal degree with minimal CV error
optimal_degree <- degrees[which.min(cv_errors)]
```

```r
cat("Optimal polynomial degree from CV:", optimal_degree, "\n")
```

```
## Optimal polynomial degree from CV: 4
```

```r
# fit models for deg 1, ... ,4
fit_1 <- lm(wage ~ poly(age, 1), data = Wage)
fit_2 <- lm(wage ~ poly(age, 2), data = Wage)
fit_3 <- lm(wage ~ poly(age, 3), data = Wage)
fit_4 <- lm(wage ~ poly(age, 4), data = Wage)

# comp models using ANOVA
anova_results <- anova(fit_1, fit_2, fit_3, fit_4)
print(anova_results)
```

```
## Analysis of Variance Table
##
## Model 1: wage ~ poly(age, 1)
## Model 2: wage ~ poly(age, 2)
## Model 3: wage ~ poly(age, 3)
## Model 4: wage ~ poly(age, 4)
##   Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1   2998 5022216
## 2   2997 4793430  1    228786 143.6025 < 2.2e-16 ***
## 3   2996 4777674  1     15756   9.8894  0.001679 **
## 4   2995 4771604  1      6070   3.8101  0.051039 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the output it would seem that cv of the polynomial regression tells us that a 4th degree polynomial is the optimal choice. ANOVA analysis however suggest that a 3rd degree is the best as it is the highest degree that retains a p-value below 0.05.

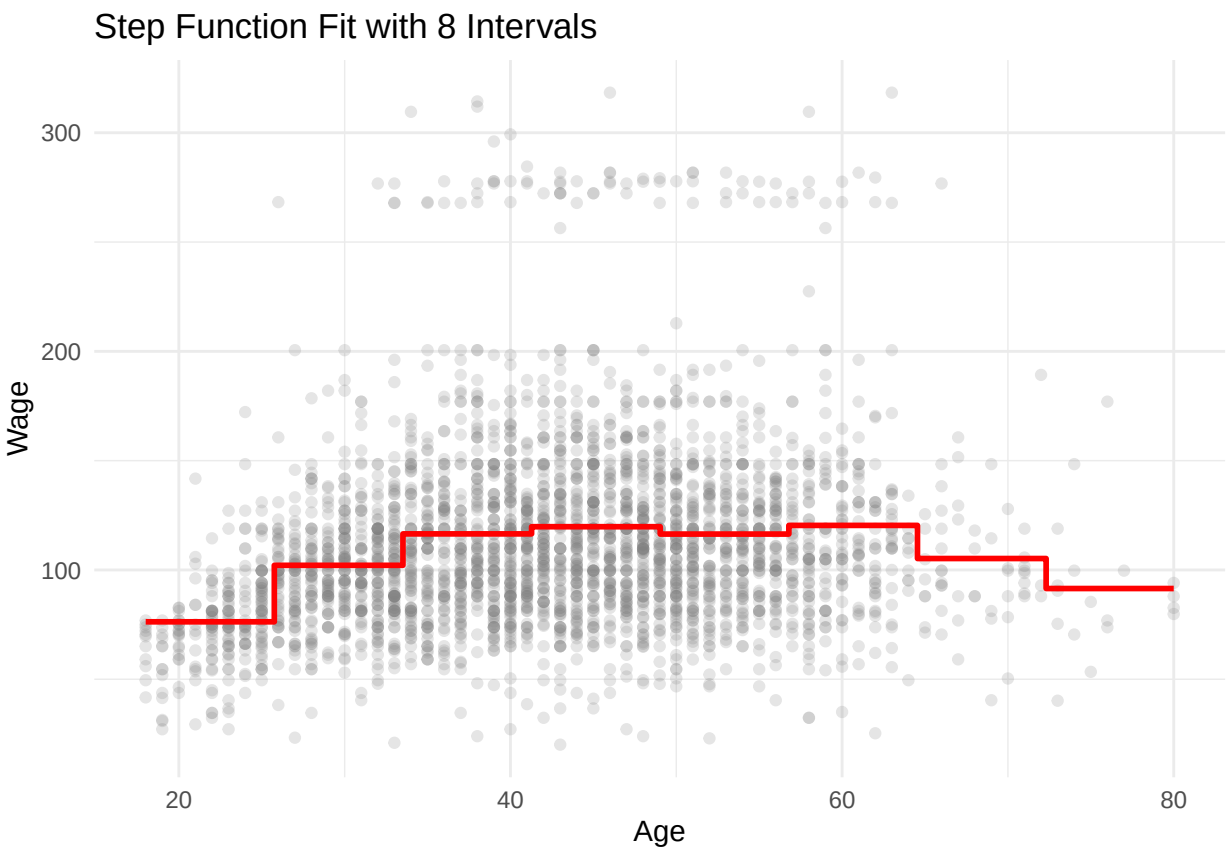
(b) Fit a step function to predict wage using age, and perform crossvalidation to choose the optimal number of cuts. Make a plot of the fit obtained.

```r
set.seed(488)
K_values <- 2:10
cv_errors_step <- numeric(length(K_values))

# age range from full dataset
age_full <- Wage$age
min_age <- min(age_full)
max_age <- max(age_full)

for (k in K_values) {
  # breaks based on dataset age range
  breaks <- seq(min_age, max_age, length.out = k + 1)

  # factor variable using breaks
  Wage$age_cut <- cut(age_full, breaks = breaks, include.lowest = TRUE)

  # 10-X CV
  cv_results <- cv.glm(
    data = Wage,
    glmfit = glm(wage ~ age_cut, data = Wage),
    K = 10
```

```
  )
  cv_errors_step[k - 1] <- cv_results$delta[1]
}

# get optimal K
opt_k <- K_values[which.min(cv_errors_step)]
cat("Optimal number of intervals from CV:", opt_k, "\n")
```

```
## Optimal number of intervals from CV: 8
```

```
# fit model using opt k on full data
optimal_breaks <- seq(min_age, max_age, length.out = opt_k + 1)
Wage$age_cut <- cut(age_full, breaks = optimal_breaks, include.lowest = TRUE)
optimal_step_model <- glm(wage ~ age_cut, data = Wage)

# get predictions for plot
pred_data <- data.frame(age = seq(min_age, max_age, length.out = 1000))
pred_data$age_cut <- cut(pred_data$age, breaks = optimal_breaks, include.lowest = TRUE)
pred_data$wage_pred <- predict(optimal_step_model, newdata = pred_data)

# plot step func fit
ggplot(Wage, aes(x = age, y = wage)) +
  geom_point(alpha = 0.2, color = "gray50") +
  geom_step(
    data = pred_data,
    aes(x = age, y = wage_pred),
    color = "red",
    linewidth = 1
  ) +
  labs(
    title = paste("Step Function Fit with", opt_k, "Intervals"),
    x = "Age",
    y = "Wage"
  ) +
  theme_minimal()
```



Exercise 9: 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.

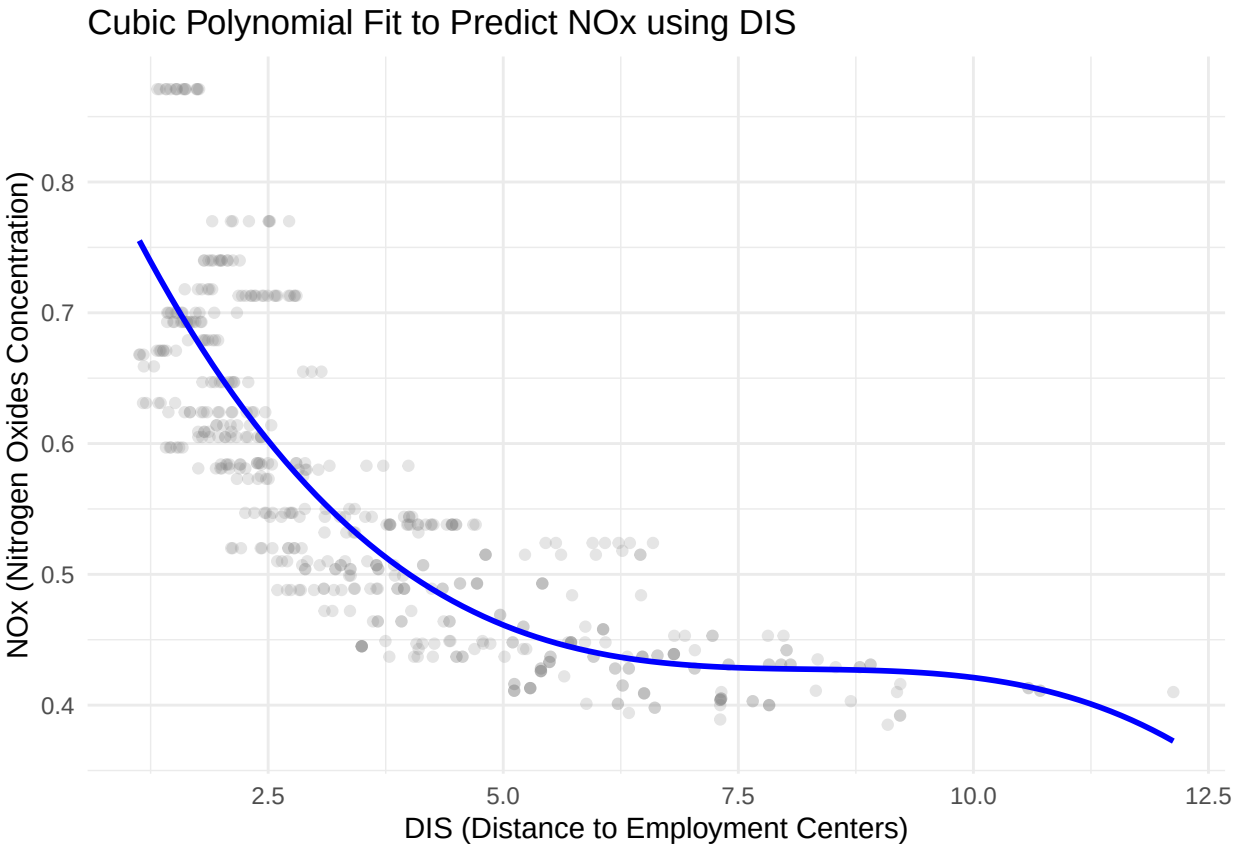
(a) Use the poly() function to fit a cubic polynomial regression to predict nox using dis. Report the regression output, and plot the resulting data and polynomial fits.

```
data(Boston)
set.seed(488)
# cubic poly regression
cubic_model <- lm(nox ~ poly(dis, 3), data = Boston)
summary(cubic_model)
```

```
##
## 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
```

```r
# predictions for plot
pred_data <- data.frame(dis = seq(min(Boston$dis), max(Boston$dis), length.out = 100))
pred_data$nox_pred <- predict(cubic_model, newdata = pred_data)
# plot data , polynomial fit
ggplot(Boston, aes(x = dis, y = nox)) +
  geom_point(alpha = 0.2, color = "gray50") +
  geom_line(data = pred_data, aes(x = dis, y = nox_pred), color = "blue", linewidth = 1) +
  labs(title = "Cubic Polynomial Fit to Predict NOx using DIS",
       x = "DIS (Distance to Employment Centers)",
       y = "NOx (Nitrogen Oxides Concentration)") +
  theme_minimal()
```

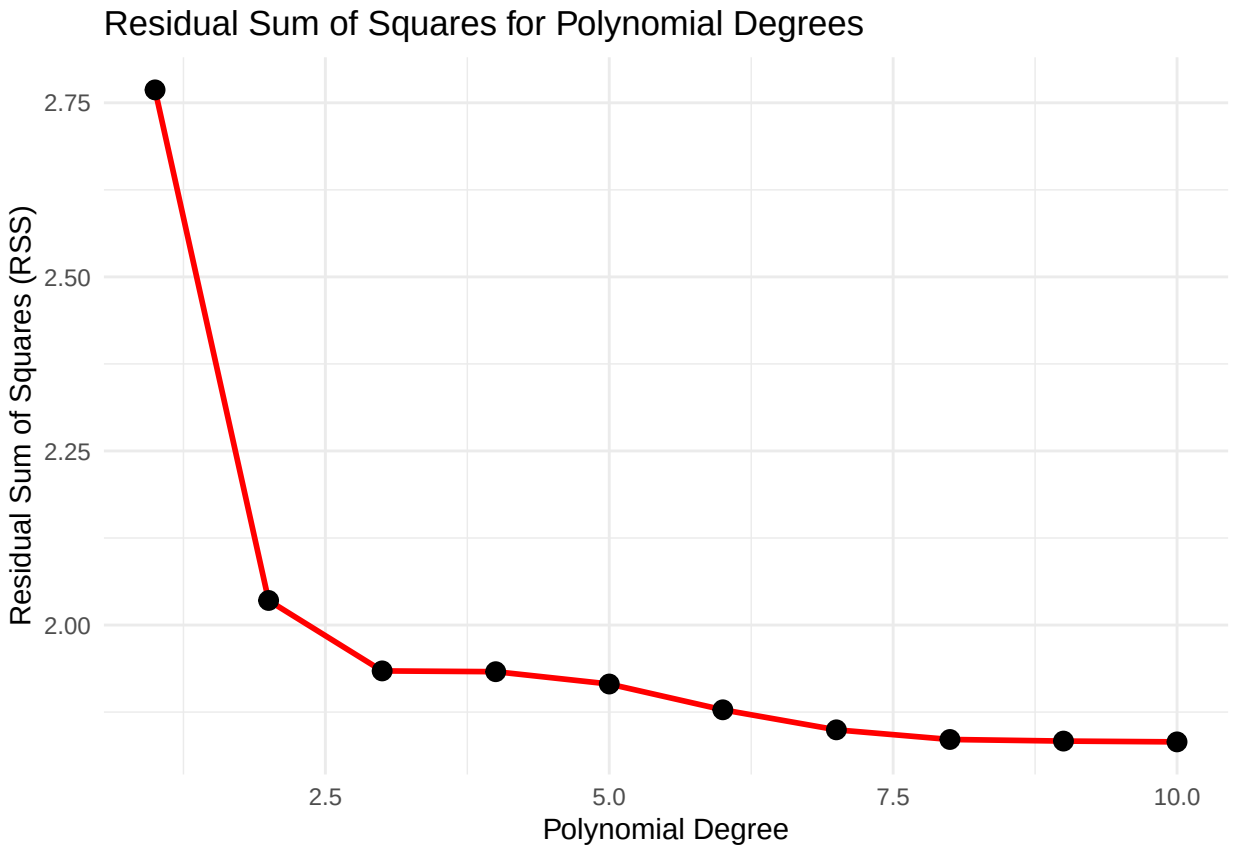


(b) Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

```r
# polynomial regression for degrees 1 to 10
degrees_poly <- 1:10
rss_poly <- numeric(length(degrees_poly))
for (d in degrees_poly) {
  # fit polynomial model
  poly_model <- lm(nox ~ poly(dis, d), data = Boston)
```

```
  # rss
  rss_poly[d] <- sum(residuals(poly_model)^2)
}
#  RSS for different polynomial degrees
rss_data <- data.frame(Degree = degrees_poly, RSS = rss_poly)
ggplot(rss_data, aes(x = Degree, y = RSS)) +
  geom_line(color = "red", size = 1) +
  geom_point(size = 3) +
  labs(title = "Residual Sum of Squares for Polynomial Degrees",
       x = "Polynomial Degree",
       y = "Residual Sum of Squares (RSS)") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
# RSS values
print(rss_data)
```

```
##    Degree      RSS
## 1       1 2.768563
## 2       2 2.035262
## 3       3 1.934107
## 4       4 1.932981
## 5       5 1.915290
```

```
## 6       6 1.878257
## 7       7 1.849484
## 8       8 1.835630
## 9       9 1.833331
## 10     10 1.832171
```

```
# RSS values for each degree
for (i in 1:length(degrees_poly)) {
  cat("Degree", degrees_poly[i], "RSS:", rss_poly[i], "\n")
}
```

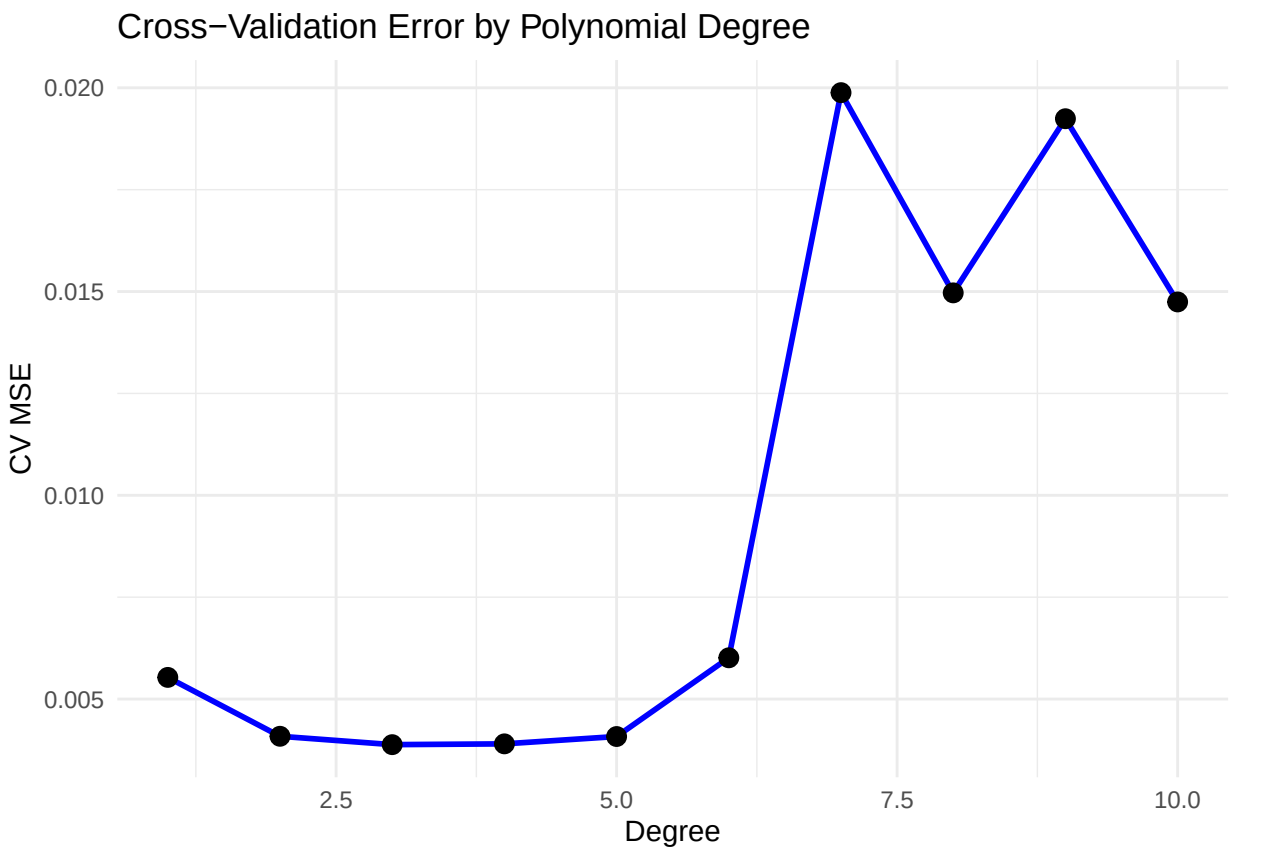
```
## Degree 1 RSS: 2.768563
## Degree 2 RSS: 2.035262
## Degree 3 RSS: 1.934107
## Degree 4 RSS: 1.932981
## Degree 5 RSS: 1.91529
## Degree 6 RSS: 1.878257
## Degree 7 RSS: 1.849484
## Degree 8 RSS: 1.83563
## Degree 9 RSS: 1.833331
## Degree 10 RSS: 1.832171
```

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

```
set.seed(488)
degrees_poly <- 1:10
cv_errors_poly <- numeric(length(degrees_poly))

for (d in degrees_poly) {
  model_formula <- as.formula(paste("nox ~ poly(dis,", d, ")"))

  # GLM and perform 10-X CV
  glm_model <- glm(model_formula, data = Boston)
  cv_results <- cv.glm(Boston, glm_model, K = 10)

  #  CV error
  cv_errors_poly[d] <- cv_results$delta[1]
}

# optimal degree
opt_poly <- degrees_poly[which.min(cv_errors_poly)]
cat("Optimal polynomial degree from CV:", opt_poly, "\n")
```

```
## Optimal polynomial degree from CV: 3
```

```
# CV errors
ggplot(data.frame(Degree = degrees_poly, CV_Error = cv_errors_poly),
       aes(x = Degree, y = CV_Error)) +
  geom_line(color = "blue", linewidth = 1) +
  geom_point(size = 3) +
  labs(title = "Cross-Validation Error by Polynomial Degree",
       x = "Degree",
       y = "CV MSE") +
  theme_minimal()
```



From the result of the cross-validation, degree 3 balances bias and variance, yielding the lowest mean CV error, so it is the best selection over higher-degree polynomials which may have extra flexibility however this doesnt necessarily mean they will result in a better prediction.

(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.

```
# spline with df=4
spline_4df <- lm(nox ~ bs(dis, df = 4), data = Boston)

# knots & model summary
knots_4df <- attr(bs(Boston$dis, df = 4), "knots")
cat("Knot locations for df=4:", knots_4df, "\n")
```

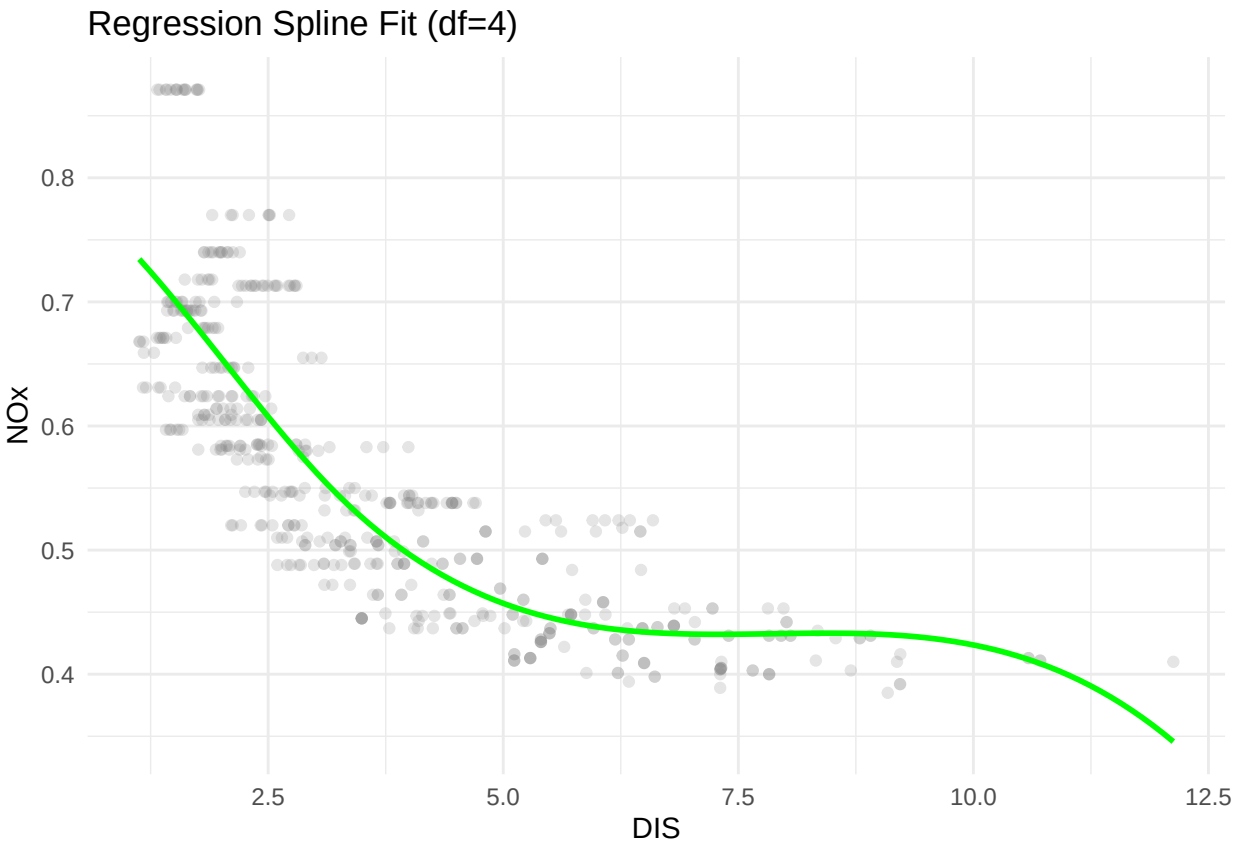
```
## Knot locations for df=4: 3.20745
```

```
summary(spline_4df)
```

```
##
## Call:
## lm(formula = nox ~ 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 ***
## 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
```

```r
# spline fit
pred_data <- data.frame(dis = seq(min(Boston$dis), max(Boston$dis), length.out = 100))
pred_data$nox_pred <- predict(spline_4df, newdata = pred_data)

ggplot(Boston, aes(x = dis, y = nox)) +
  geom_point(alpha = 0.2, color = "gray50") +
  geom_line(data = pred_data, aes(x = dis, y = nox_pred), color = "green", linewidth = 1) +
  labs(title = "Regression Spline Fit (df=4)",
       x = "DIS",
       y = "NOx") +
  theme_minimal()
```



The placement of knots is determined by the data distribution and the number of degrees of freedom. In this case, we used 4 degrees of freedom, which allows for a flexible fit while avoiding overfitting. The knots are placed at quantiles of the predictor variable by default this may help the spline better convey the underlying structure in the data.

(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.

```r
df_values <- 3:10
rss_spline <- numeric(length(df_values))

for (i in seq_along(df_values)) {
  spline_model <- lm(nox ~ bs(dis, df = df_values[i]), data = Boston)
  rss_spline[i] <- sum(residuals(spline_model)^2)
}

# RSS vs df
ggplot(data.frame(DF = df_values, RSS = rss_spline), aes(x = DF, y = RSS)) +
  geom_line(color = "purple", linewidth = 1) +
  geom_point(size = 3) +
  labs(title = "RSS for Regression Splines",
       x = "Degrees of Freedom",
       y = "Residual Sum of Squares") +
  theme_minimal()
```


```r
# RSS values
cat("RSS for Splines:\n")
```

```
## RSS for Splines:
```

```r
print(data.frame(DF = df_values, RSS = rss_spline))
```

```
##   DF      RSS
## 1  3 1.934107
## 2  4 1.922775
## 3  5 1.840173
```

```
## 4  6 1.833966
## 5  7 1.829884
## 6  8 1.816995
## 7  9 1.825653
## 8 10 1.792535
```

The RSS values indicate that as the degrees of freedom increase, the RSS tends to decrease, suggesting a better fit to the training data. However, this levels off, suggesting that additional knots beyond ≈ 6 capture little extra signal and mostly fit noise. The optimal degree of freedom should balance bias and variance.

(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.

```r
set.seed(488)
cv_errors_spline <- numeric(length(df_values))

for (i in seq_along(df_values)) {
  # model formula
  model_formula <- as.formula(paste("nox ~ bs(dis, df =", df_values[i], ")"))

  # fit GLM , perform 10-X CV
  glm_model <- glm(model_formula, data = Boston)
  cv_results <- cv.glm(Boston, glm_model, K = 10)

  # CV error
  cv_errors_spline[i] <- cv_results$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%` = 3.2157), 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(1.1296, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

```
## Warning in bs(dis, degree = 3L, knots = c(`50%` = 3.1222), 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%` = 3.1222), Boundary.knots =
## c(1.137, : some 'x' values beyond boundary knots may cause ill-conditioned
## bases
```

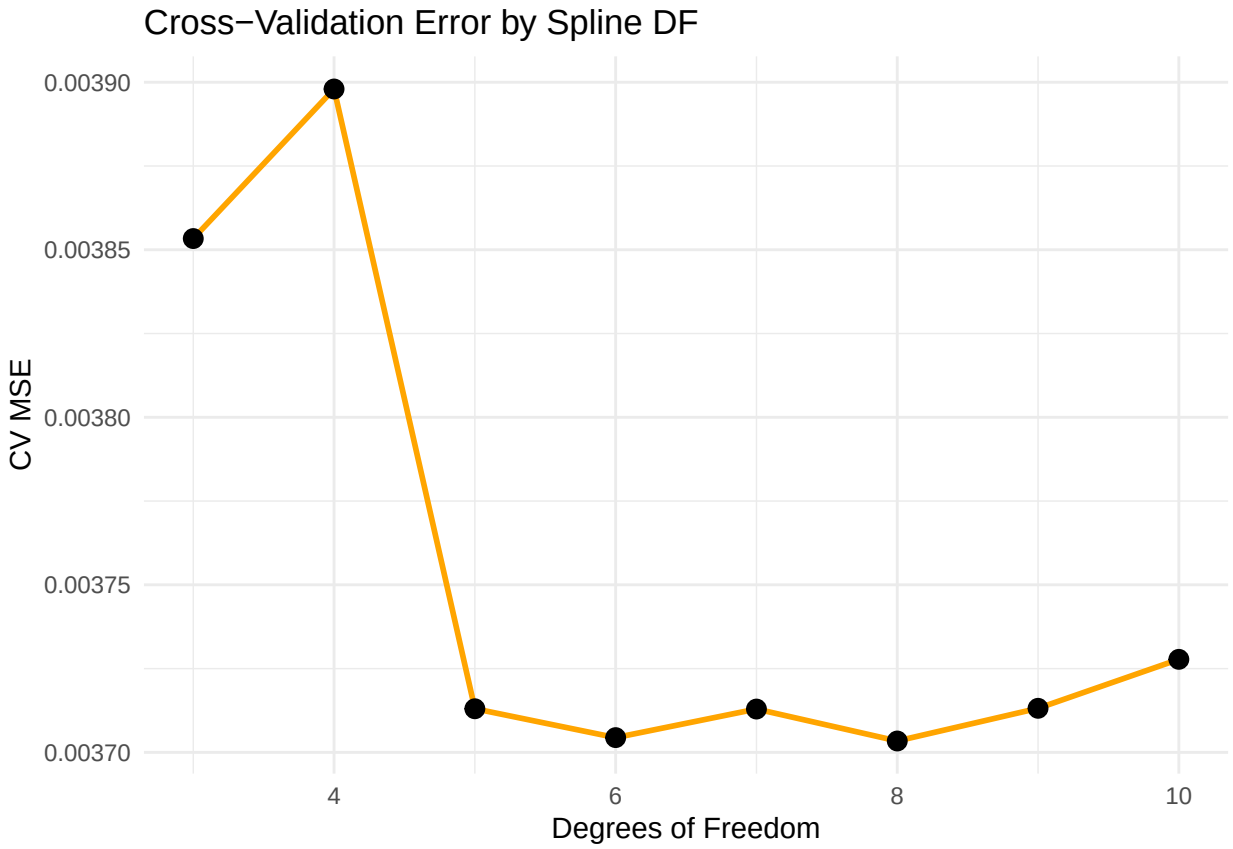
```
## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.3546, `66.66667%` =
## 4.23906666666667: some 'x' values beyond boundary knots may cause
## ill-conditioned bases
## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.3546, `66.66667%` =
## 4.23906666666667: some 'x' values beyond boundary knots may cause
## ill-conditioned bases
```

```
## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.38876666666667, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning in bs(dis, degree = 3L, knots = c(`33.33333%` = 2.38876666666667, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.1084, `50%` = 3.2157, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.1084, `50%` = 3.2157, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.10915, `50%` = 3.2759, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning in bs(dis, degree = 3L, knots = c(`25%` = 2.10915, `50%` = 3.2759, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.96794, `40%` = 2.6439, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.96794, `40%` = 2.6439, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.9709, `40%` = 2.6775, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning in bs(dis, degree = 3L, knots = c(`20%` = 1.9709, `40%` = 2.6775, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.82226666666667, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.82226666666667, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.8301, `33.33333%` =
## 2.354, : some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning in bs(dis, degree = 3L, knots = c(`16.66667%` = 1.8301, `33.33333%` =
## 2.354, : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.79877142857143, :
## some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning in bs(dis, degree = 3L, knots = c(`14.28571%` = 1.79877142857143, :
## 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%` = 1.743225, `25%` = 2.08285,
## : some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.743225, `25%` = 2.08285,
## : some 'x' values beyond boundary knots may cause ill-conditioned bases

## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.751575, `25%` = 2.10525,
## : some 'x' values beyond boundary knots may cause ill-conditioned bases
## Warning in bs(dis, degree = 3L, knots = c(`12.5%` = 1.751575, `25%` = 2.10525,
## : some 'x' values beyond boundary knots may cause ill-conditioned bases
```

```r
# get optimal df
opt_df_spline <- df_values[which.min(cv_errors_spline)]
cat("Optimal spline df from CV:", opt_df_spline, "\n")
```

```
## Optimal spline df from CV: 8
```

```r
#  CV errors
ggplot(data.frame(DF = df_values, CV_Error = cv_errors_spline),
       aes(x = DF, y = CV_Error)) +
  geom_line(color = "orange", linewidth = 1) +
  geom_point(size = 3) +
  labs(title = "Cross-Validation Error by Spline DF",
       x = "Degrees of Freedom",
       y = "CV MSE") +
  theme_minimal()
```



10-X cross-validation was performed to select the optimal degrees of freedom for the regression spline. It reaches its lowest mean-squared error at $df = 8$, suggesting that an eight-degree-of-freedom spline offers the best bias–variance trade-off, with $df = 7$ or 9 performing nearly as well.