Ken Ye
jy294

STA 325
HW 3

#1

a) $RSS(\beta_0, \beta_1) \equiv \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$

$$\frac{\partial RSS(\beta_0, \beta_1)}{\partial \beta_0} = -2 \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) = 0$$

$$\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} \hat{\beta}_0 - \sum_{i=1}^{n} \hat{\beta}_1 x_i = 0$$

$$n\bar{y} - n\hat{\beta}_0 - n\hat{\beta}_1 \bar{x} = 0$$

$$\boxed{\hat{\beta}_0^{LS} = \bar{y} - \hat{\beta}_1^{LS} \bar{x}}$$

$$\frac{\partial RSS(\beta_0, \beta_1)}{\partial \beta_1} = -2 \sum_{i=1}^{n} x_i (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) = 0$$

$$\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \hat{\beta}_0 - \sum_{i=1}^{n} \hat{\beta}_1 x_i^2 = 0$$

$$\sum_{i=1}^{n} x_i y_i - \hat{\beta}_0 n\bar{x} - \hat{\beta}_1 \sum_{i=1}^{n} x_i^2 = 0 \quad (\text{sub } \hat{\beta}_0)$$

$$\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} + \hat{\beta}_1 n\bar{x}^2 - \hat{\beta}_1 \sum_{i=1}^{n} x_i^2 = 0$$

$$\hat{\beta}_1 \left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2 \right) = \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}$$

$$= \sum_{i=1}^{n} (x_i - \bar{x})^2 = 1$$

$$\boxed{\hat{\beta}_1^{LS} = \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}$$

Scanned with CamScanner

#1

b) $\frac{\partial\left(\sum_{i=1}^{n}(y_i-\beta_0-\beta_1 x_i)^2+\lambda\beta_1^2\right)}{\partial\beta_0} = -2\sum_{i=1}^{n}(y_i-\hat\beta_0-\hat\beta_1 x_i)=0$

$\sum_{i=1}^{n} y_i-\sum_{i=1}^{n}\hat\beta_0-\sum_{i=1}^{n}\hat\beta_1 x_i=0$

similar to part a) $\Rightarrow$

$n\bar y-n\hat\beta_0-n\hat\beta_1\bar x=0$

$$\boxed{\hat\beta_{0,\lambda}^{R}=\bar y-\hat\beta_{1,\lambda}^{R}\bar x}$$

$\frac{\partial\left(\sum_{i=1}^{n}(y_i-\beta_0-\beta_1 x_i)^2+\lambda\beta_1^2\right)}{\partial\beta_1} = -2\sum_{i=1}^{n}x_i(y_i-\hat\beta_0-\hat\beta_1 x_i)+2\lambda\hat\beta_1=0$

$\sum_{i=1}^{n}x_i y_i-\hat\beta_0 n\bar x-\hat\beta_1\sum x_i^2-\lambda\hat\beta_1=0$ (sub $\hat\beta_0$)

$\sum_{i=1}^{n}x_i y_i-n\bar x\bar y+\hat\beta_1 n\bar x^2-\hat\beta_1\sum_{i=1}^{n}x_i^2-\lambda\hat\beta_1=0$

$(-n\bar x^2+\sum_{i=1}^{n}x_i^2+\lambda)\hat\beta_1=\sum x_i y_i-n\bar x\bar y$

$-n\bar x^2+\sum_{i=1}^{n}x_i^2=\sum_{i=1}^{n}(x_i-\bar x)^2=1$

$\hat\beta_1^{LS}=\sum x_i y_i-n\bar x\bar y$

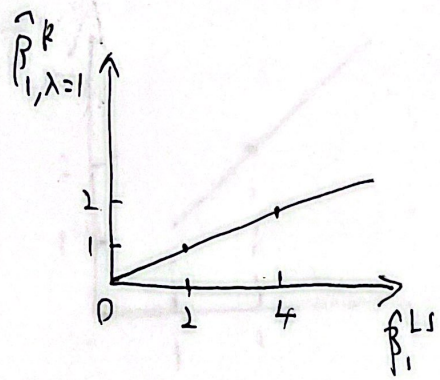
$(1+\lambda)\hat\beta_1=\hat\beta_1^{LS}$

$$\boxed{\hat\beta_{1,\lambda}^{R}=\frac{\hat\beta_1^{LS}}{1+\lambda}}$$

Scanned with CamScanner

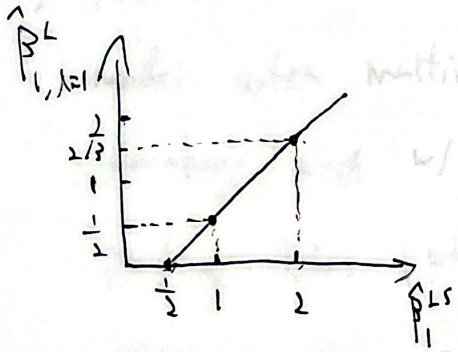
#4)

c) $\lambda = 1$, $\hat{\beta}^R_{1,\lambda} = \frac{\hat{\beta}^{LS}_1}{2}$



There is a linear relationship b/w $\hat{\beta}^R_{1,\lambda}$ and $\hat{\beta}^{LS}_1$. Since $\hat{\beta}^R_{1,\lambda=1}$ does not equal to 0 unless $\hat{\beta}^{LS}_1$ equal to 0, it doesn't have the ability to select important variables.

e) $\lambda = 1$, $\hat{\beta}^L_{1,\lambda} = (\hat{\beta}^{LS}_1 - \frac{1}{2})_+$



$\hat{\beta}^L_{1,\lambda=1}$ can perform variable selection. predictors with $\hat{\beta}^{LS}_1 < \frac{1}{2}$ will be considered inactive & shrunk to 0 when $\hat{\beta}^L_{1,\lambda}$ is used.

f) $\min_{\beta_0,\beta_1} \{RSS(\beta_0,\beta_1) + \lambda I(\beta_i \neq 0)\}$

⇓

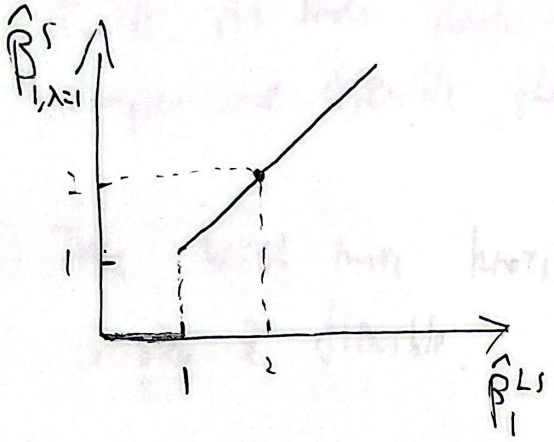
minimize $\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$, subject to $I(\beta_i \neq 0) \leq S$

When $S=1$, this is equivalent of minimizing RSS, subject to $\beta_i \neq 0$. Best subset selection finds model w/ smallest RSS, here we are doing sth. similar to best subset selection, while in most cases β_i may be 0.

Scanned with CamScanner

#1

h) $\lambda=1,\ \hat{\beta}^{S}_{1,\lambda} = \hat{\beta}^{LS}_{1} \cdot I\left(\hat{\beta}^{LS}_{1} \geq 1\right)$



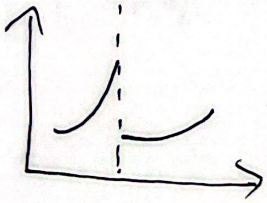
$\hat{\beta}^{S}_{1,\lambda=1}$ can perform variable selection as it's shrunk to 0 when $\hat{\beta}^{LS}_{1} < 1$. Variables w/ $\hat{\beta}^{LS}_{1} < 1$ is considered inactive.

#2

a) False. LS estimation produces a high-variance and unbiased model when multicollinearity exists. Ridge regression performs shrinkage, and w/ an optimal λ value chosen by CV, trading bias (which becomes higher) for variance (which becomes lower), a smaller MSPE could be achieved.

b) True. Lasso can perform variable selection, but not Ridge. Therefore, Lasso would result in a smaller model with fewer active predictors.

c) True. Without constraints, piecewise polynomial may not be continuous at breaking points (knots), like the graph below. Constraints can be imposed to ensure continuity if desired.



Scanned with CamScanner

#2

d) False. As more knots are added, the model becomes more complex and flexible, thus has higher variance.

e) True. With more knots, spline models can be more complex & flexible.

f) False. A model with more degrees of freedom has more effective variables that can be adjusted to fit the data, so the model can be more complex & flexible. This leads to higher variance and lower bias.

#3

a) A quartic spline with k knots can be modeled as

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \beta_3 b_3(x_i) + \dots + \beta_{k+4} b_{4}(x_i) + \varepsilon_i$$

where $b_1(x_i) = x_i$

$b_2(x_i) = x_i^2$

$b_3(x_i) = x_i^3$

$b_4(x_i) = x_i^4$

$b_{k+4}(x_i) = (x_i - \xi)_+^4 = \begin{cases} (x-\xi)^4 & \text{if } x > \xi \\ 0 & \text{otherwise} \end{cases}$ for $k = 1, \dots, K$

1 intercept + 4 basis functions + k knots

$\Downarrow$

k+5 d.f.s

Scanned with CamScanner

#3

b) property (i): it is quartic polynomial b/w any two neighboring knots.

① Before first knot: $y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 + \varepsilon_i$

② At m^{th} knot: $y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 + \ldots + \beta_{m+4}(x_i - \xi_m)^4 + \varepsilon_i$
$1 < m < k$

③ After last knot: $y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 + \ldots + \beta_{k+4}(x_i - \xi_k)^4 + \varepsilon_i$
(k^{th})

After the intercept and the basis functions, all polynomials being added with the addition of knot $k = 1 \ldots k$ are polynomial of degree 4.

property (ii): continuous derivatives of up to order 3 at each knot.

Suppose $x_i > \xi_{k=0}$

Left

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4$$

$$\frac{dy_i}{dx_i} = \beta_1 + 2\beta_2 x_i + 3\beta_3 x_i^2 + 4\beta_4 x_i^3$$

$$\frac{d^2 y_i}{dx_i^2} = 2\beta_2 + 6\beta_3 x_i + 12\beta_4 x_i^2$$

$$\frac{d^3 y_i}{dx_i^3} = 6\beta_3 + 24\beta_4 x_i$$

$$\frac{d^4 y_i}{dx_i^4} = 24\beta_4$$

Right

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 + \underline{\beta_5 (x_i - \xi_1)^4}$$

$$\frac{dy_i}{dx_i} = \beta_1 + 2\beta_2 x_i + 3\beta_3 x_i^2 + 4\beta_4 x_i^3 + \underline{4\beta_5 (x_i - \xi_1)^3}$$

$$\frac{d^2 y_i}{dx_i^2} = 2\beta_2 + 6\beta_3 x_i + 12\beta_4 x_i^2 + \underline{12\beta_5 (x_i - \xi_1)^2}$$

$$\frac{d^3 y_i}{dx_i^3} = 6\beta_3 + 24\beta_4 x_i + \underline{24\beta_5 (x_i - \xi_1)}$$

$$\frac{d^4 y_i}{dx_i^4} = 24\beta_4 + 24\beta_5$$

✓ These become 0 when approach from right.

Therefore, the 1st, 2nd, and 3rd derivative from left & right side equal to each other, thus continuous. However, the 4th derivatives are different.

Scanned with CamScanner

#3

c) I disagree. Quartic splines, as formulated in part a), can be efficiently computed just like linear regression. We can also easily compute coefficient estimation, standard errors, confidence intervals, hypothesis testings. Fundamentally, it can be represented as a linear model. Therefore, quartic splines can be fit as efficiently as a single linear model.

d) I disagree.

First, higher-order polynomial (such as degree 15) model is unstable and can perform poorly at the boundaries (Runge's phenomenon), whereas quartic spline is more stable.

Second, splines often give a better predictive performance over polynomial models because the true regression function is typically never exactly a k^{th} order polynomial over the whole domain. And splines give a more flexible model that allow for rapid changes in certain regions, but not in others. In fact, true regression function f can often be well-approximated by low-order polynomials in local regions.

Scanned with CamScanner

HW3

Ken Ye

2023-11-08

Question 4 (ISL Chapter 7, 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.

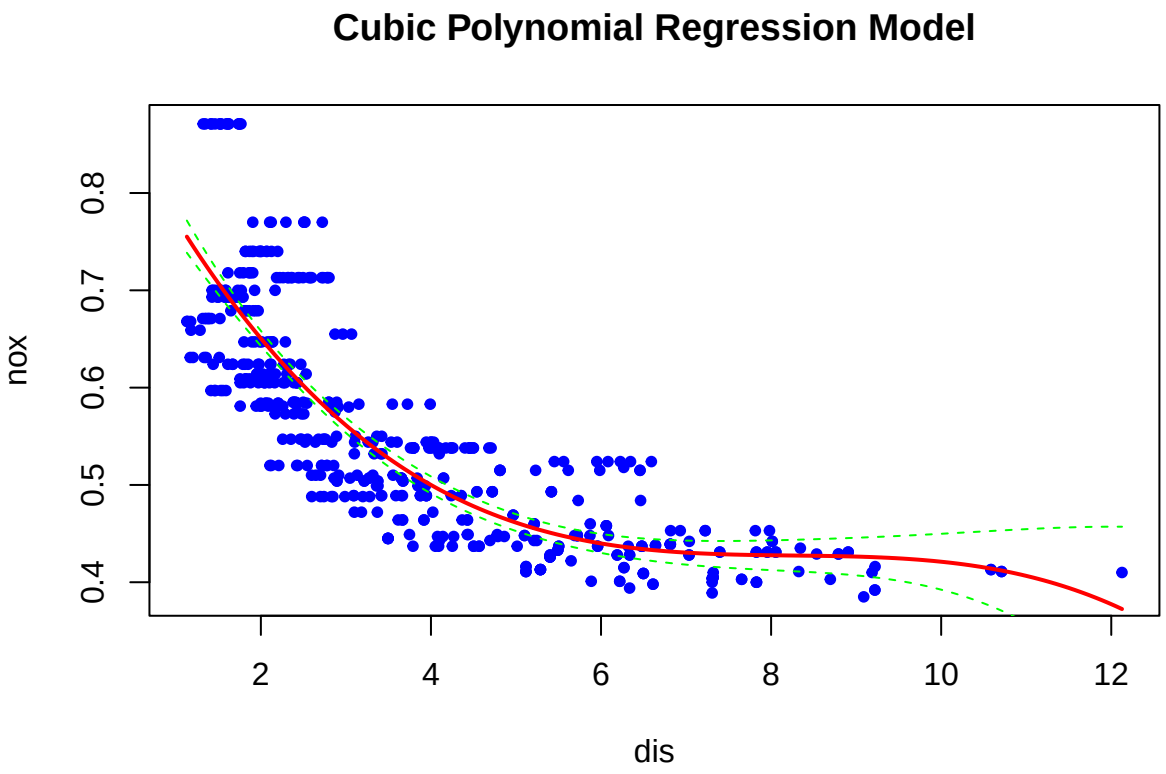
```
library(MASS)
library(boot)
library(splines)
data("Boston")
attach(Boston)
```

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.

```
# Fit cubic polynomial regression model
cubic_model <- lm(nox ~ poly(dis, 3))
summary(cubic_model)
```

```
##
## Call:
## lm(formula = nox ~ poly(dis, 3))
##
## 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
```

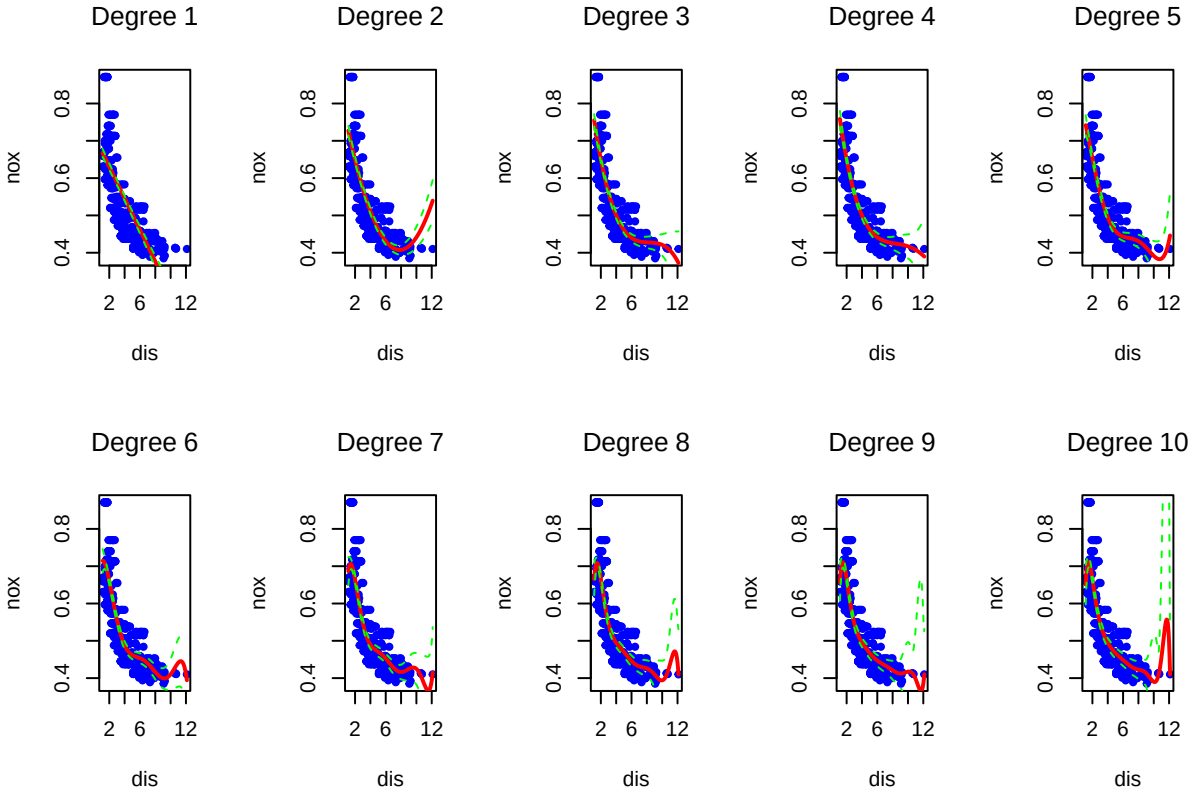
```r
# Plot
dis_seq <- seq(min(dis), max(dis), length.out = nrow(Boston))

predictions <- predict(cubic_model,
                       newdata = data.frame(dis = dis_seq),
                       se.fit = TRUE)

predicted_nox <- predictions$fit
se <- predictions$se.fit

plot(dis, nox, pch = 20, col = "blue")
lines(dis_seq, predicted_nox, col = "red", lwd = 2)
lines(dis_seq, predicted_nox + 2 * se, col = "green", lty = 2)
lines(dis_seq, predicted_nox - 2 * se, col = "green", lty = 2)
title("Cubic Polynomial Regression Model")
```



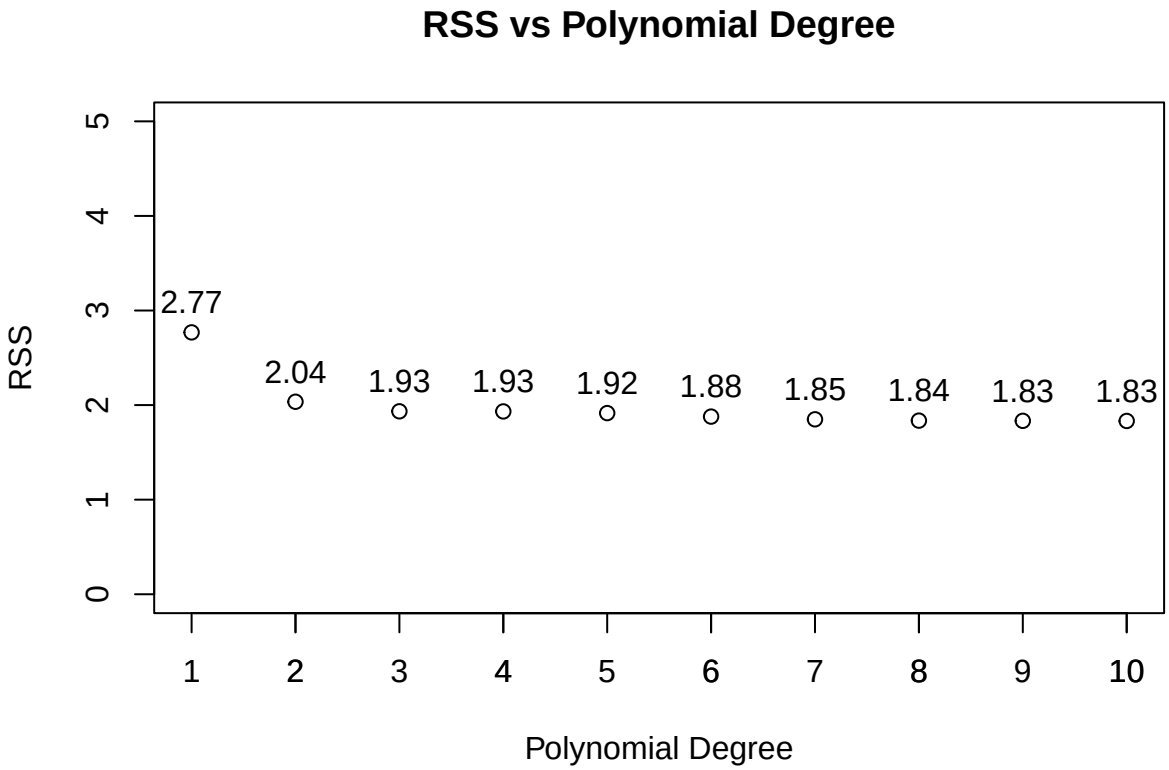
(b) Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

```r
# Fit degree 1-10 models and plot
par(mfrow = c(2, 5))
rss_vals <- numeric(10)
for (i in 1:10){
  model <- lm(nox ~ poly(dis, i))

  dis_seq <- seq(min(dis), max(dis), length.out = nrow(Boston))

  predictions <- predict(model,
```

```
                          newdata = data.frame(dis = dis_seq),
                          se.fit = TRUE)

  predicted_nox <- predictions$fit
  se <- predictions$se.fit

  plot(dis, nox, pch = 20, col = "blue",
       main = bquote("Degree" ~ .(i)))
  lines(dis_seq, predicted_nox, col = "red", lwd = 2)
  lines(dis_seq, predicted_nox + 2 * se, col = "green", lty = 2)
  lines(dis_seq, predicted_nox - 2 * se, col = "green", lty = 2)

  rss_vals[i] = sum(model$residuals^2)
}
```




```
# Plot RSS
plot(rss_vals,
     xlab = "Polynomial Degree",
     ylab = "RSS",
     main = "RSS vs Polynomial Degree",
     ylim = c(0, 5))
axis(1, at = 1:10, labels = 1:10)
text(1:10, rss_vals, labels = round(rss_vals, digits = 2), pos = 3)
```



The above graph shows the RSS values for polynomial degree 1-10.

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

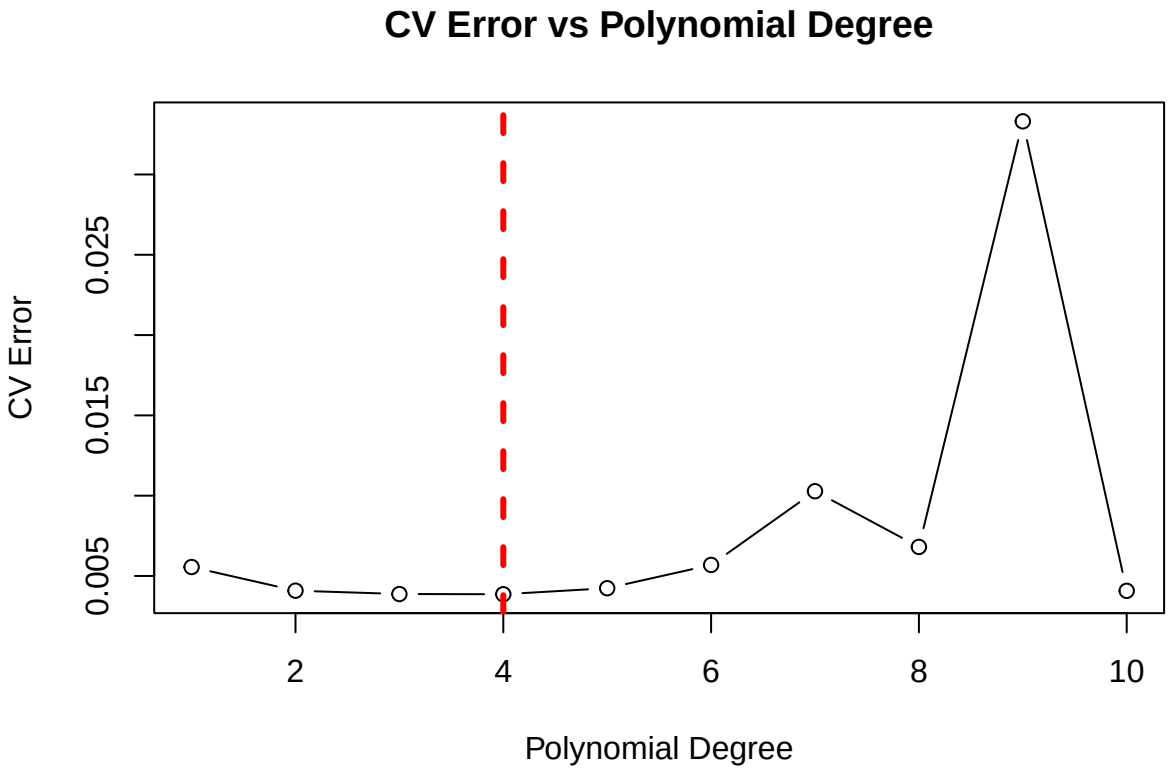
```
# Perform CV
set.seed(1)
cv.error <- numeric(10)
for (i in 1:10){
  fit = glm(nox ~ poly(dis,i))
  cv.error[i] = cv.glm(Boston,fit, K =10)$delta[1]
}

print(cv.error)
```

```
##  [1] 0.005558263 0.004085706 0.003876521 0.003863342 0.004237452 0.005686862
##  [7] 0.010278897 0.006810868 0.033308607 0.004075599
```

```
best_degree <- which.min(cv.error)
cat("The degree with the smallest cross-validation error is", best_degree)
```

```
## The degree with the smallest cross-validation error is 4
```

```
# Plot CV
plot(cv.error,
     type = "b",
     xlab = "Polynomial Degree",
     ylab = "CV Error",
     main = "CV Error vs Polynomial Degree")
abline(v = 4, col = "red", lwd = 3, lty = 2)
```



We performed 10-fold cross validation and computed the validation error for different polynomial degrees (1-10). The degree that yielded the minimum CV error is degree 4, which had an error of 0.003863342, and this implies degree 4 might be the optimal degree for this question, but test MSPE needs to be assessed to prove it.

(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.

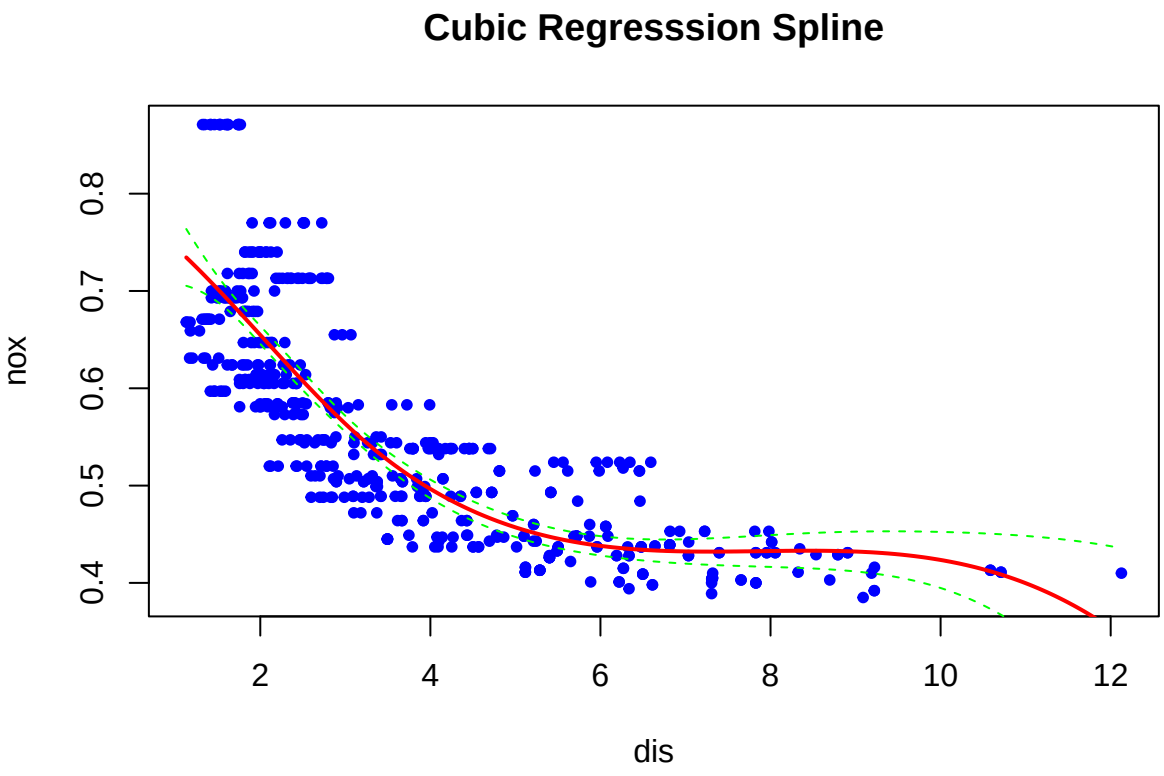
To choose the knots, we know that cubic spline with K knots would have K + 4 parameters or degrees of freedom. Therefore, with four degrees of freedom we can only have a cubic spline with zero knot.

```r
# Fit cubic spline
cs.fit <- lm(nox ~ bs(dis, df = 4))
summary(cs.fit)
```

```
##
## Call:
## lm(formula = nox ~ bs(dis, df = 4))
##
## 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
```

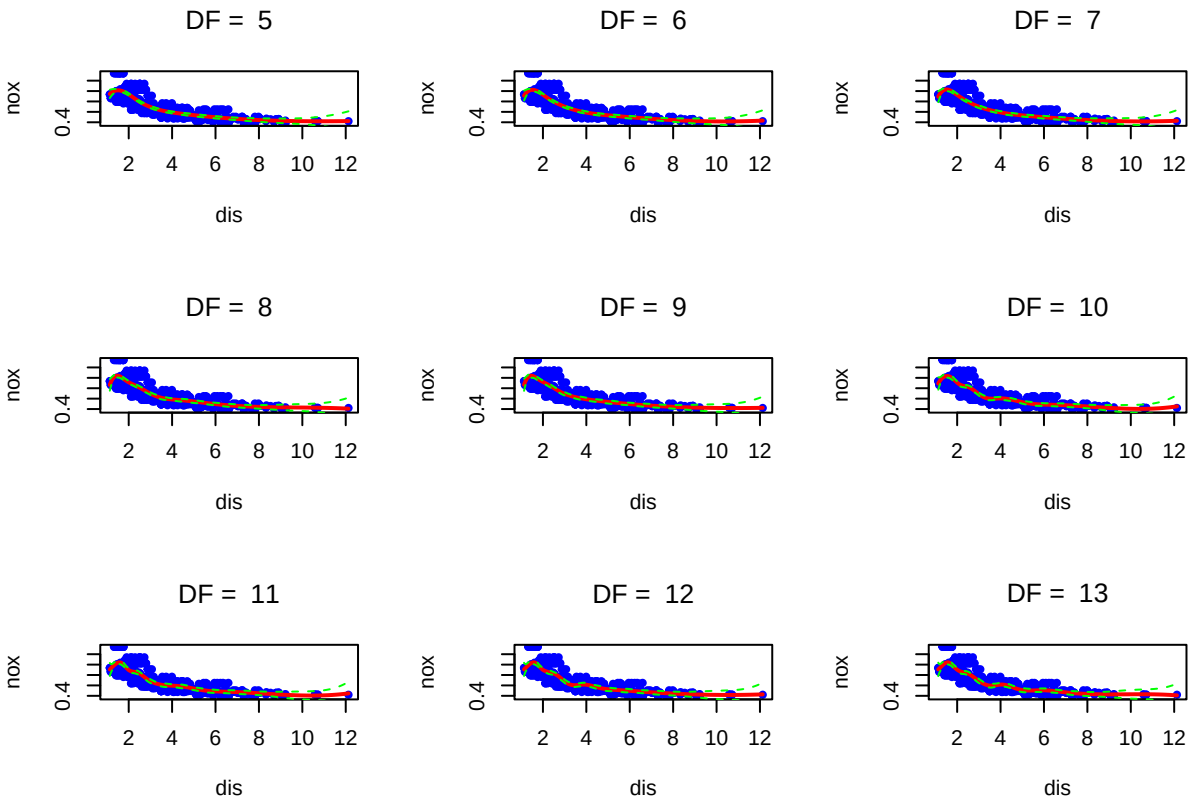
```r
# Plot
dis_seq <- seq(min(dis), max(dis), length.out = nrow(Boston))

predictions <- predict(cs.fit,
                       newdata = data.frame(dis = dis_seq),
                       se.fit = TRUE)

predicted_nox <- predictions$fit
se <- predictions$se.fit

plot(dis, nox, pch = 20, col = "blue")
lines(dis_seq, predicted_nox, col = "red", lwd = 2)
lines(dis_seq, predicted_nox + 2 * se, col = "green", lty = 2)
lines(dis_seq, predicted_nox - 2 * se, col = "green", lty = 2)
title("Cubic Regresssion Spline")
```



(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
# Fit df 5-13 splines and plot
par(mfrow = c(3, 3))
rss_vals <- numeric(9)
for (i in 5:13){
```

```
  cs.fit <- lm(nox ~ bs(dis, df = i))

  dis_seq <- seq(min(dis), max(dis), length.out = nrow(Boston))

  predictions <- predict(cs.fit,
                         newdata = data.frame(dis = dis_seq),
                         se.fit = TRUE)

  predicted_nox <- predictions$fit
  se <- predictions$se.fit

  plot(dis, nox, pch = 20, col = "blue",
       main = bquote("DF = " ~ .(i)))
  lines(dis_seq, predicted_nox, col = "red", lwd = 2)
  lines(dis_seq, predicted_nox + 2 * se, col = "green", lty = 2)
  lines(dis_seq, predicted_nox - 2 * se, col = "green", lty = 2)

  rss_vals[i-4] = sum(cs.fit$residuals^2)
}
```



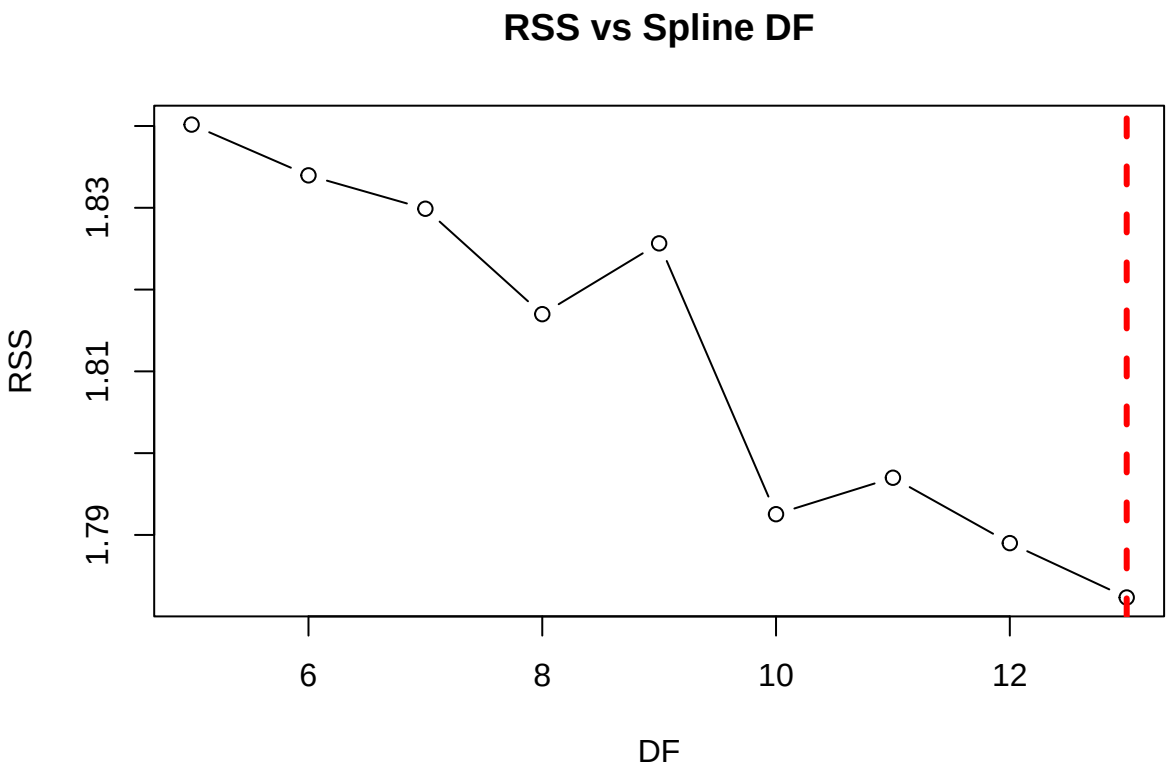

```
print(rss_vals)
```

```
## [1] 1.840173 1.833966 1.829884 1.816995 1.825653 1.792535 1.796992 1.788999
## [9] 1.782350
```

```r
best_df <- which.min(rss_vals) + 4
cat("The df with the smallest RSS is", best_df)
```

```
## The df with the smallest RSS is 13
```

```r
# Plot RSS
degrees_of_freedom <- 5:13
plot(degrees_of_freedom, rss_vals,
     type = "b",
     xlab = "DF",
     ylab = "RSS",
     main = "RSS vs Spline DF")
abline(v = 13, col = "red", lwd = 3, lty = 2)
```



We tried degree of freedom 5-13 for the cubic splines and computed the RSS for each. The DF that yielded the minimum RSS is 13 (the largest we tried), which had an error of 1.782350. For cubic spline models with higher degrees of freedom (more knots), the model generally has less RSS. However, they are generally more wiggly than models with fewer degrees of freedom. This illustrates the bias-variance tradeoff — more complex & flexible models (cubic splines with higher degrees of freedom) have lower bias (lower RSS) but higher variance, and vice versa.

(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
# Perform CV
set.seed(1)
cv.error <- numeric(9)
for (i in 5:13){
  fit <- glm(nox ~ bs(dis, df = i))
```

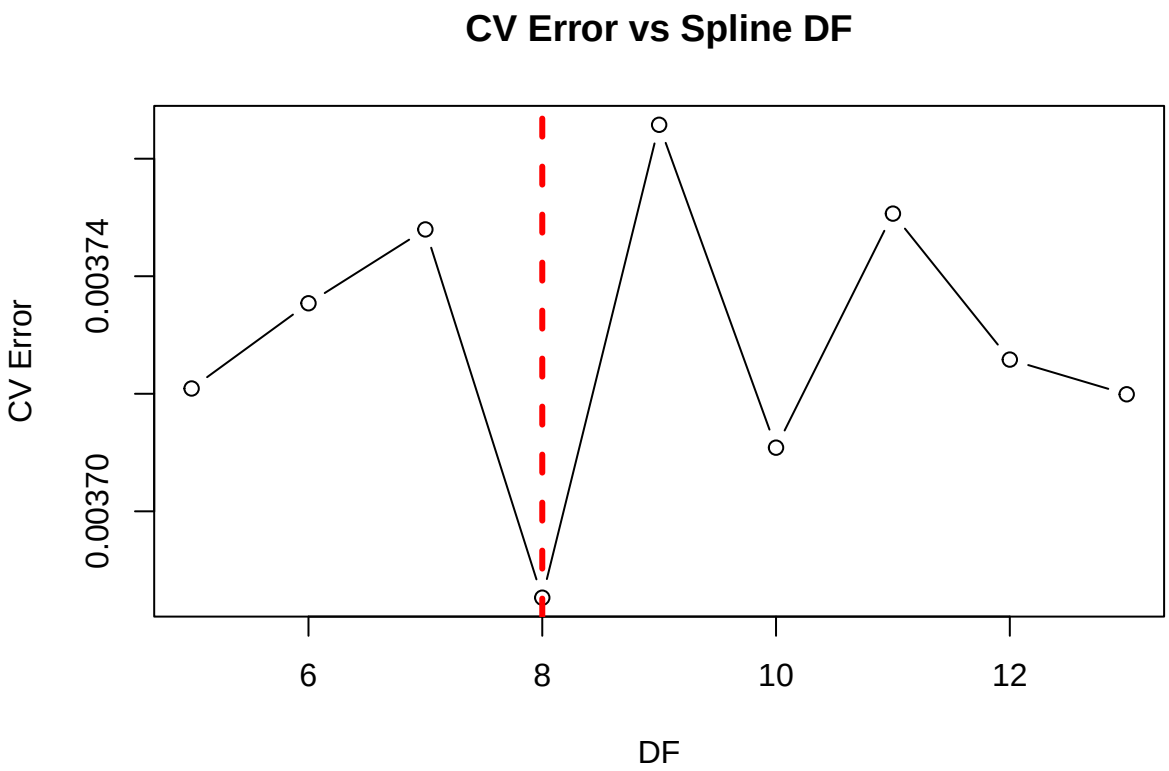
```r
  cv.error[i-4] = cv.glm(Boston,fit, K = 10)$delta[1]
}

print(cv.error)
```

```
## [1] 0.003720901 0.003735400 0.003747967 0.003685305 0.003765768 0.003710831
## [7] 0.003750657 0.003725816 0.003719913
```

```r
best_degree <- which.min(cv.error) + 4
cat("The df with the smallest cross-validation error is", best_degree)
```

```
## The df with the smallest cross-validation error is 8
```

```r
# Plot CV
degrees_of_freedom <- 5:13
plot(degrees_of_freedom, cv.error,
     type = "b",
     xlab = "DF",
     ylab = "CV Error",
     main = "CV Error vs Spline DF")
abline(v = 8, col = "red", lwd = 3, lty = 2)
```



We performed 10-fold cross validation and computed the validation error for different degrees of freedom (5-13) for the cubic splines. The df that yielded the minimum CV error is 8, which had an error of 0.003685305, and this implies `df = 8` might be the optimal df for this question, but test MSPE needs to be assessed to prove it. Also, a caveat is that the difference in CV errors for these df's being examined is actually very small, and we know that complex models might be harder to interpret. Therefore, in reality, model with smaller df might be chosen.