Question 1

#a $RSS(\beta_0, \beta_1) = \sum (y_i - \beta_0 - \beta_1 x_i)^2$

$$\frac{d}{d\beta_0} = -\sum 2(y_i - \hat{\beta_0} - \hat{\beta_1} x_i) = 0$$
$$-2\sum (y_i - \hat{\beta_0} - \hat{\beta_1} x_i) = 0$$
$$\sum (y_i - \hat{\beta_0} - \hat{\beta_1} x_i) = 0$$
$$\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} \hat{\beta_0} - \sum_{i=1}^{n} \hat{\beta_1} x_i = 0$$
$$n\bar{y} - n\hat{\beta_0} - n\hat{\beta_1}\bar{x} = 0$$
$$n\hat{\beta_0} = n\bar{y} - n\hat{\beta_1}\bar{x}$$
$$\hat{\beta_0}^{LS} = \bar{y} - \hat{\beta_1}^{LS}\bar{x}$$

$$\frac{d}{d\beta_1} = -\sum 2x_i(y_i - \hat{\beta_0} - \hat{\beta_1} x_i) = 0$$
$$\sum (2x_i y_i - 2\hat{\beta_0} x_i - 2\hat{\beta_1} x_i^2) = 0$$
$$\sum 2x_i y_i - \sum 2\hat{\beta_0} x_i - \sum 2\hat{\beta_1} x_i^2 = 0$$
$$\sum x_i y_i - \hat{\beta_0} n\bar{x} - \hat{\beta_1} \sum x_i^2 = 0$$
$$\sum x_i y_i - (\bar{y} - \hat{\beta_1}\bar{x}) n\bar{x} - \hat{\beta_1} \sum x_i^2 = 0$$
$$\sum x_i y_i - n\bar{x}\bar{y} + n\hat{\beta_1}\bar{x}^2 - \hat{\beta_1} \sum x_i^2 = 0$$
$$\hat{\beta_1}\left(\sum x_i^2 - n\bar{x}^2\right) = \sum x_i y_i - n\bar{x}\bar{y}$$
$$\hat{\beta_1}^{LS} = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sum x_i^2 - n\bar{x}^2} = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sum (x_i - \bar{x})^2} = \sum x_i y_i - n\bar{x}\bar{y}$$

equals one by question setting

#b $\min \{RSS(\beta_0, \beta_1) + \lambda \beta_1^2\}$

$$= \sum (y_i - \beta_0 - \beta_1 x_i)^2 - \lambda \beta_1^2$$

$$\frac{d}{d\beta_0} = -\sum 2(y_i - \hat{\beta_0} - \hat{\beta_1} x_i) = 0$$
$$-2\sum (y_i - \hat{\beta_0} - \hat{\beta_1} x_i) = 0$$
$$\sum (y_i - \hat{\beta_0} - \hat{\beta_1} x_i) = 0$$
$$\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} \hat{\beta_0} - \sum_{i=1}^{n} \hat{\beta_1} x_i = 0$$
$$n\bar{y} - n\beta_0 - n\hat{\beta_1}\bar{x} = 0$$
$$n\hat{\beta_0} = n\bar{y} - n\hat{\beta_1}\bar{x}$$
$$\hat{\beta_{0,\lambda}}^{R} = \bar{y} - \hat{\beta_{1,\lambda}}^{R}\bar{x}$$

$$\frac{\partial}{\partial \beta_1} = -\sum 2x_i(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) - 2\lambda\hat{\beta}_1 = 0$$

$$\sum 2x_i y_i - \sum 2x_i \hat{\beta}_0 - \sum 2x_i^2 \hat{\beta}_1 - 2\lambda\hat{\beta}_1 = 0$$

$$= \sum x_i y_i - \hat{\beta}_0 n\bar{x} - \hat{\beta}_1 \sum x_i^2 - \lambda\hat{\beta}_1 = 0$$

$$= \sum x_i y_i - (\bar{y} - \hat{\beta}_1\bar{x})n\bar{x} - \hat{\beta}_1 \sum x_i^2 - \lambda\hat{\beta}_1 = 0$$

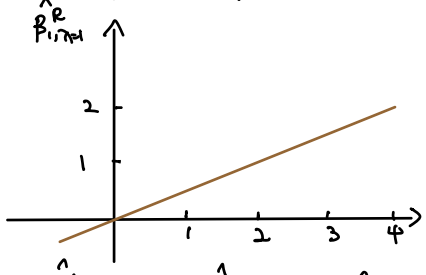
$$= \sum x_i y_i - n\bar{y}\bar{x} + n\hat{\beta}_1\bar{x}^2 - \hat{\beta}_1 \sum x_i^2 - \lambda\hat{\beta}_1 = 0$$

$$(-n\bar{x}^2 + \sum x_i^2 + \lambda)\hat{\beta}_1 = \sum x_i y_i - n\bar{x}\bar{y} = \hat{\beta}_1^{LS}$$

$$\hat{\beta}_1 = \frac{\hat{\beta}_1^{LS}}{(-n\bar{x}^2 + \sum x_i^2 + \lambda)} = \frac{\hat{\beta}_1^{LS}}{1+\lambda}$$

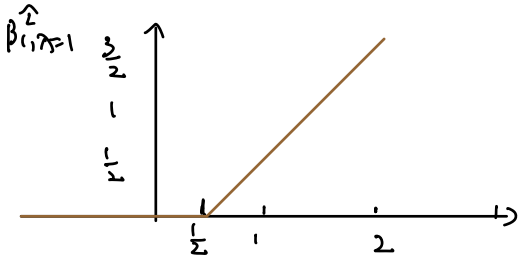
$= \sum (x_i - \bar{x})^2$

#c as $\lambda = 1$ $\quad \hat{\beta}_{1,\lambda}^R = \frac{\hat{\beta}_1^{LS}}{2}$



Comment: I don't think the ridge estimator at $\lambda=1$ is capable of selecting important variables as there's a linear relationship with $\hat{\beta}_1^{LS}$ and $\hat{\beta}_{1,\lambda=1}^R$ does not reach to zero unless $\hat{\beta}_1^{LS}$ is zero, which is almost never the case

#e $\quad \hat{\beta}_{0\lambda}^L = \bar{y} - \hat{\beta}_{1\lambda}\bar{x}$, $\hat{\beta}_{1\lambda}\bar{x} = \max\{\hat{\beta}_1^{LS} - \frac{\lambda}{2}, 0\} = \max\{\beta_1 - \frac{1}{2}, 0\}$



Comment: I can tell that Lasso is capable of model shrinkage at $\lambda=1$. From the plot, all predictors whose $\hat{\beta}_1^{LS} < \frac{1}{2}$ is considered inactive and shrunk to zero.

#f $\quad \min\{RSS(\beta_0,\beta_1) + \lambda L(\beta_1 \neq 0)\}$

$= \min\{RSS(\beta_0,\beta_1) + \|\beta\|_0\}$

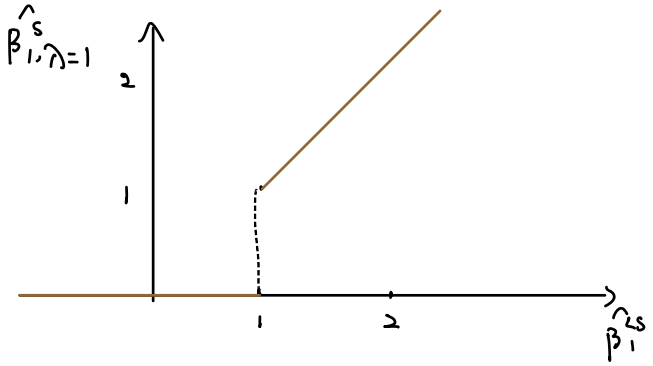
$\Leftrightarrow$ minimize $\sum(y_i - \beta_0 - \beta_1 x_i)^2$ subject to $\|\beta\|_0$, or $I(\beta_1 \neq 0) \leq s$

Interpretation: When $s=1$, this is basically minimizing RSS, subject to $\beta_1 \neq 0$. Since best subset selection is to find the model with smallest RSS, while in most of the cases β_1 wouldn't be zero, what we're doing here is very alike best subset selection

take $s=1$ See what problem it is $\Rightarrow$ "Best subset selection"

h

$$\lambda = 1 \Rightarrow \hat{\beta}^{S}_{1,\lambda} = \hat{\beta}^{S}_{1,\lambda} \cdot I(\hat{\beta}_1^{LS} \geq 1)$$



(comment: from this plot I can tell that it's capable of selecting import predictors as $\hat{\beta}^{S}_{1,\lambda=1}$ is shrunk to zero for when $\hat{\beta}_1^{LS} < 1$. So those variables with $\hat{\beta}_1^{LS} < 1$ is considered inactive

d (Bonus)

$$\min \{ RSS(\beta_1, \beta_0) + \lambda|\beta_1| \}$$

Lasso estimators $(\hat{\beta}^{L}_{0,\lambda}, \hat{\beta}^{L}_{1,\lambda})$ solve two equations

$$\begin{cases} -2\Sigma(y_i - \beta_0 - \beta_1 x_i) = 0 \\ -2\Sigma x_i (y_i - \beta_0 - \beta_1 x_i) + \lambda \partial\beta_1 \ni 0 \end{cases} \quad (1)$$

where $\partial\beta_1$ is the sub-differential of β_1

$$\partial\beta_1 = \begin{cases} -1, & \beta_1 < 0 \\ [-1, +1], & \beta_1 = 0 \\ +1, & \beta_1 > 0 \end{cases}$$

- Suppose Least-□ estimate $\hat{\beta}_1^{LS} > \frac{\lambda}{2}, > 0$

  estimators in (1) simplify to

  $$\hat{\beta}^{L}_{0,\lambda} = \bar{y} - \hat{\beta}^{L}_{1,\lambda} \bar{x}$$

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

  $$= 2\hat{\beta}_1^{LS} - \lambda$$

→ check if they solve equation (1)

1) $-2\Sigma(y_i - \beta_0 - \beta_1 x_i)$

$= -2(\Sigma y_i - \Sigma\beta_0 - \Sigma\beta_1 x_i)$

$= -2(n\bar{y} - n\beta_0 - n\bar{x}\beta_1)$

$= -2(n\bar{y} - (n\bar{y} - n\hat{\beta}^{L}_{1,\lambda}\bar{x}) - n\bar{x}\hat{\beta}^{L}_{1,x})$

$= -2 \times 0 = 0$ ✓

2) $-2\sum x_i(y_i - \beta_0 - \beta_1 x_i) + \lambda\partial\beta_1$

$= -2[\sum x_i y_i - \sum x_i\beta_0 - \sum x_i\beta_1] + \lambda\partial\beta_1$

$= -2[\sum x_i y_i - n\bar{x}\beta_0 - n\bar{x}\beta_1] + \lambda\partial\beta_1$

$= -2[\sum x_i y_i - n\bar{x}(\bar{y} - \hat{\beta}_{1,\lambda}^{L}\bar{x}) - n\bar{x}\hat{\beta}_{1,\lambda}^{L}] - \lambda$

$= -2[\sum x_i y_i - n\bar{x}\bar{y} + n\bar{x}\hat{\beta}_{1,\lambda}^{L}\bar{x} - n\bar{x}\hat{\beta}_{1,\lambda}^{L}] - \lambda$

$= -2[\sum x_i y_i - n\bar{x}\bar{y} + n\bar{x}^2(2\sum x_i y_i - 2n\bar{x}\bar{y} - \lambda)$

$- n\bar{x}(2\sum x_i y_i - 2n\bar{x}\bar{y} - \lambda) - \lambda$

$= -2$??

- Suppose Least-□ estimate $\hat{\beta}_1^{LS} < \frac{\lambda}{2}$, 0 ?? 与0的关系

estimators in (1) simplify to

$\hat{\beta}_{0,\lambda}^{L} = \bar{y} - \hat{\beta}_{1,\lambda}^{L}\bar{x}$

$\hat{\beta}_{1,\lambda}^{L} = (\hat{\beta}_1^{LS} - \lambda/2) - (\hat{\beta}_1^{LS} - \frac{\lambda}{2})$

$= 0$

#9. (Bonus)

$\min \{RSS(\beta_0,\beta_1) + \lambda\mathbb{1}(\beta_1 \neq 0)\}$, suppose $\hat{\beta}_1^{LS} > 0$

$= \Sigma(y_i - \beta_0 - \beta_1 x_i)^2 - \lambda\mathbb{1}(\beta_1 \neq 0)$

$\frac{\partial}{\partial\beta_0} = -\sum 2(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) = 0$

$-2\Sigma(y_i - \hat{\beta_0} - \hat{\beta_1}x_i) = 0$

$\Sigma(y_i - \hat{\beta_0} - \hat{\beta_1}x_i) = 0$

$\sum_{i=1}^{n} y_i - \sum_{i=1}^{n}\hat{\beta}_0 - \sum_{i=1}^{n}\beta_1\hat{x}_i = 0$

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

$n\hat{\beta_0} = n\bar{y} - n\hat{\beta_1}\bar{x}$

$\hat{\beta_0}^{S} = \bar{y} - \hat{\beta_1}^{S}\bar{x}$

$\frac{\partial}{\partial\beta_1} = -\sum 2x_i(y_i - \hat{\beta_0} - \hat{\beta_1}x_i) = 0$

$= -2[\sum x_i y_i - \sum x_i\hat{\beta_0} - \sum x_i^2\hat{\beta_1}] = 0$

$$\Rightarrow \quad \sum x_i y_i - n\bar{x}\hat{\beta_0} - \hat{\beta_1}\sum x_i^2 = 0$$

$$\sum x_i y_i - n\bar{x}(\bar{y} - \hat{\beta_1}\bar{x}) - \hat{\beta_1}\sum x_i^2 = 0$$

$$\sum x_i y_i - n\bar{x}\bar{y} + \hat{\beta_1} n\bar{x}^2 - \hat{\beta_1}\sum x_i^2 = 0$$

$$\hat{\beta_1}(n\bar{x}^2 - \sum x_i^2) = n\bar{x}\bar{y} - \sum x_i y_i$$

$$\hat{\beta_1}^{LS} = \frac{n\bar{x}\bar{y} - \sum x_i y_i}{(n\bar{x} - \sum x_i^2)} = \sum x_i y_i - n\bar{x}\bar{y}$$

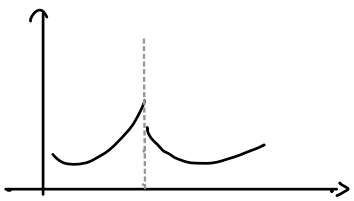
$$= \hat{\beta}_1^{LS}$$ how's $I(\hat{\beta_1}^{LS} \geq \sqrt{\lambda})$??

Question 2

(a) False: when there is high multicollinearity in data, Least-Square estimation would have high variance, and Ridge regression should be used to reduce that variance with a little sacrifice in bias to achieve better MSE

(b) True: Lasso, unlike ridge regression, can perform variable selection. Therefore it would result in smaller model with few predictors

(c) True: without constrains, piece wise polynomials may not start/end the same at the breaking point, resulting in plot like below



Therefore it may not be continuous

(d) False. As more knots are added, model becomes more flexible therefore the fitted model can have higher variance not less.

(e) True: With more more knot, spline models get more complex & flexible. You can fit different polynomials that are specific to the local regio

(f) False: A model with more degree of freedom has more effective variables It is therefore more complex & flexible. Therefore it has less bias in its fit not more.

#3 Quartic Spline

(a) Full Model Specification

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \beta_3 b_3(x_i) + \cdots + \beta_{k+4} b_4(x_i) + \varepsilon_i$$

$$b_1(x_i) = x_i$$

$$b_2(x_i) = x_i^2$$

$$b_3(x_i) = x_i^3$$

$$b_4(x_i) = x_i^4$$

$$b_{k+4}(x_i) = (x_i - \xi_k)_+^4, \quad k = 1, 2, \cdots, K$$

where $(x_i - \xi_k)^3 = \begin{cases} (x_i - \xi_k)^3 & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$

There are k+4 degrees of freedom in my model

(b) property i : it is quartic polynomial btw 2 neighor knots

1) Before 1st 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) Btw m & n knots : $y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 + \cdots + \beta_{m+4}(x_i - \xi_m)_+^4 + \varepsilon_i$

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

We can see that in all scenarios, the highest degree polynomial is 4

property ii: Continuous derivative up to order 3

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

$$\frac{\partial^0 y_i}{\partial x_i^0} = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 + \beta_5 (x_i - \xi_1)^4$$

left side

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

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

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

$$\frac{\partial^4 y_i}{\partial x^4} = 24\beta_4 \neq 0$$

right side $\frac{\partial^0 y_i}{\partial x_i^0} = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \beta_4 x_i^4 + \beta_5 (x_i - \xi_1)^4$

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

$\frac{\partial^2 y_i}{\partial x_i^2} = 2\beta_2 + 6\beta_3 x_i + 12\beta_4 x_i^2 + 12\beta_5 (x_i - \xi_1)^2$ when approach from right

$\frac{\partial^3 y_i}{\partial x_i^3} = 6\beta_3 + 24\beta_4 x_i + 24\beta_5 (x_i - \xi_1)$ they become zero

$\frac{\partial^4 y_i}{\partial x_i^4} = 24\beta_5$

Therefore the 1st, 2nd, 3rd derivatives equal one another from both sides

We can tell that $\frac{\partial^4 y_i}{\partial x_i^4}$ does not equal to one another except $\beta_4 = \beta_5$, which eleminates local fitting & makes it global quartic fitting.

(c) I don't agree with her.

With cubic splines, we can easily compute estimates, prediction, confidence intervals, SE ... just like we did for Linear regression Therefore cubic splines are not much more computationally expensive than simple linear models

(d) I don't agree with her.

The true regression model that we want is almost never a 15 degree polynomial over the full input space. On the other hand, true regression model can typically be fit by lower order polynomial in local regions.

In addition, high order polynomial can have wild tail behavior making extrapolation almost infeasible.

Therefore splines would with 16 d.f. would be a better choice.

STA 325 hw3

Archie Ju

2022-10-17

Question 4, ISH Chapter 7 Exercise 9

Part A)

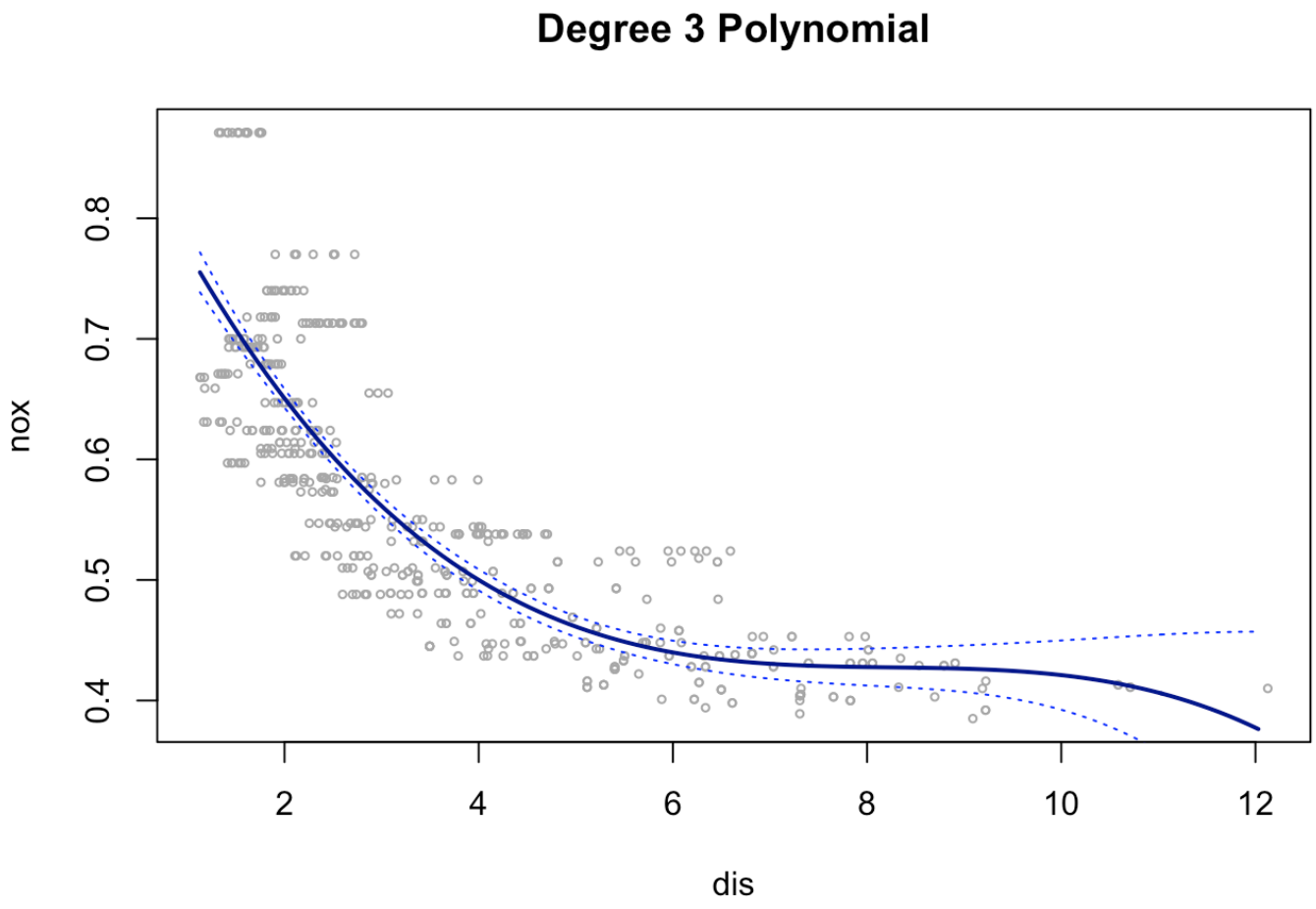
```
library(MASS)
data("Boston")
attach(Boston)
```

```
poly.fit = lm(nox ~ poly(dis, 3, raw = TRUE))
coef(summary(poly.fit))
```

```
##                                Estimate   Std. Error    t value      Pr(>|t|)
## (Intercept)                 0.9341280720 0.0207076150  45.110365 9.853624e-179
## poly(dis, 3, raw = TRUE)1 -0.1820816950 0.0146973264 -12.388763  6.078843e-31
## poly(dis, 3, raw = TRUE)2  0.0219276580 0.0029329119   7.476412  3.428917e-13
## poly(dis, 3, raw = TRUE)3 -0.0008849957 0.0001727172  -5.123959  4.274950e-07
```

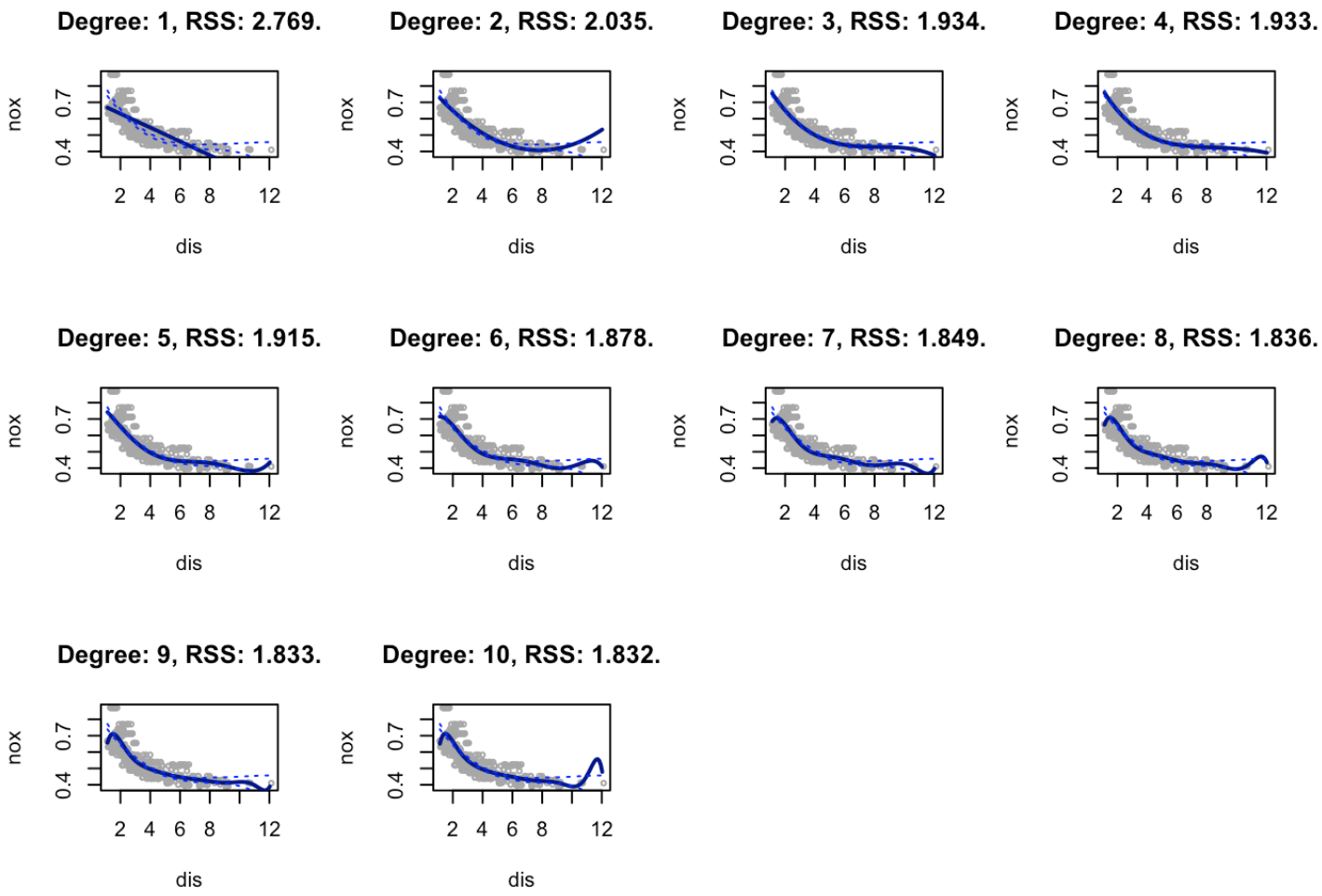
```
dislims= range(dis)
dis.grid <- seq(from = dislims[1], to = dislims[2], by = 0.1)
preds = predict(poly.fit, newdata = list(dis = dis.grid), se = TRUE)
se.bands = cbind(preds$fit + 2*preds$se.fit, preds$fit - 2*preds$se.fit)
```

```
plot(dis, nox, xlim = dislims, cex = 0.5, col = "darkgrey")
title("Degree 3 Polynomial")
lines(dis.grid, preds$fit, lwd = 2, col = "darkblue")
matlines(dis.grid, se.bands, lwd = 1, col = "blue", lty = 3)
```



Part B)

```
par(mfrow = c(3,4))
for (i in 1:10){
  fit = lm(nox ~ poly(dis, i, raw = TRUE))
  preds = predict(fit, newdata = list(dis = dis.grid), se = TRUE)
  plot(dis, nox, cex = 0.5, col = "darkgrey")
  lines(dis.grid, preds$fit, lwd = 2, col = "darkblue")
  matlines(dis.grid, se.bands, lwd = 1, col = "blue", lty = 3)
  title(sprintf("Degree: %s, RSS: %s.",i, round(sum(fit$residuals^2), 3)))
}
```



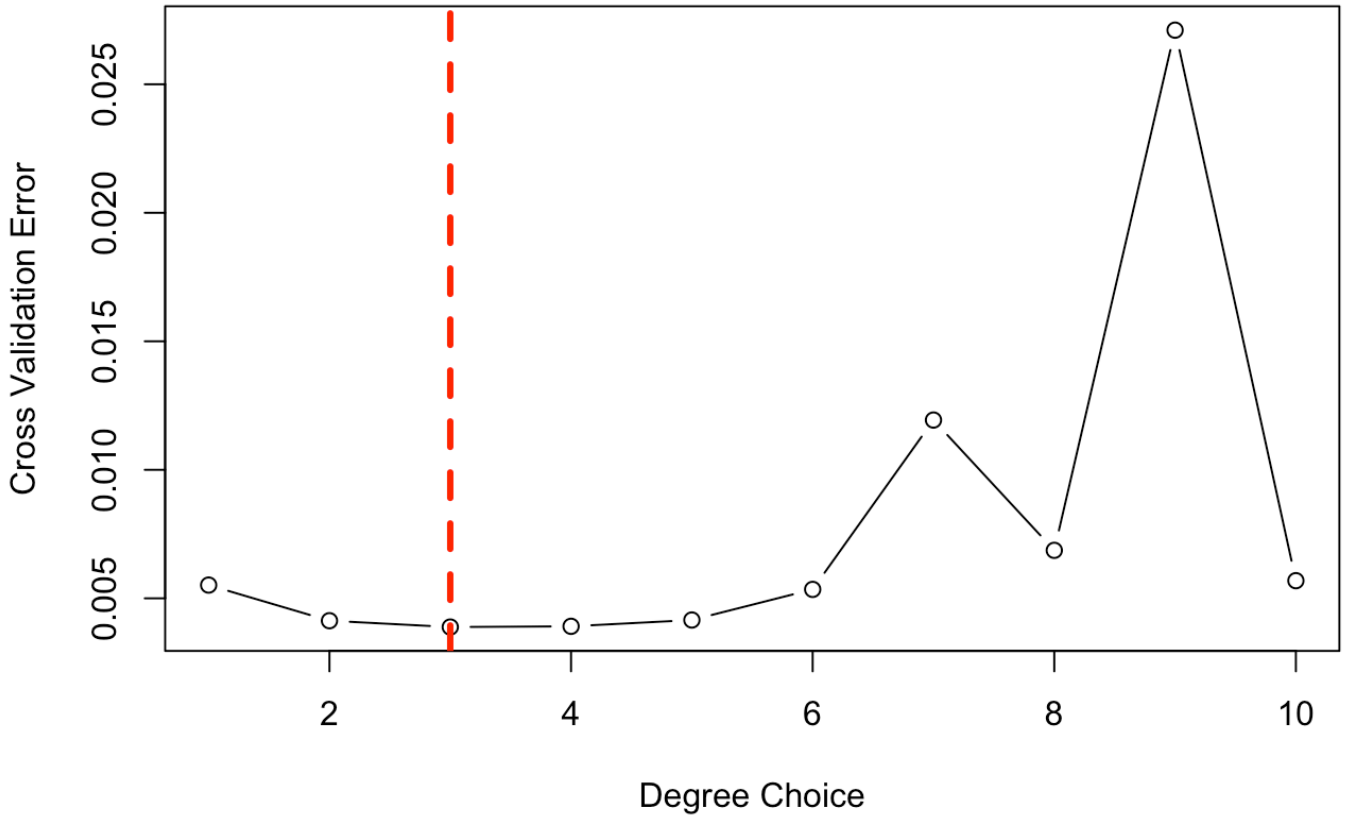
You may refer back to the plot for Residual Sum of Square (RSS).

Part C)

```
library(boot)
set.seed(325)
degree=1:10
cv.error=rep(0,10)
for (i in degree){
    fit = glm(nox ~ poly(dis ,i, raw =T), data=Boston)
    cv.error[i] = cv.glm(Boston,fit, K=10)$delta[1]
}
cv.error
```

```
##  [1] 0.005515136 0.004128439 0.003882659 0.003911212 0.004156136 0.005347354
##  [7] 0.011936081 0.006866949 0.027106214 0.005686458
```

```
plot(cv.error, type = "b", xlab = "Degree Choice", ylab = "Cross Validation Error"
)
abline(v = 3, col = "red", lwd=3, lty=2)
```




```
min(cv.error)
```

```
## [1] 0.003882659
```

I take a 10-fold cross validation approach by computing the validation error for different polynomial degree settings (from 1-10) According to the cross validation error plot, minmium cv.error occurs at polynomial degree of 3, which implies cubic polynomial fitting would work the best and degree 3 would be the optimal degree.

Part D)

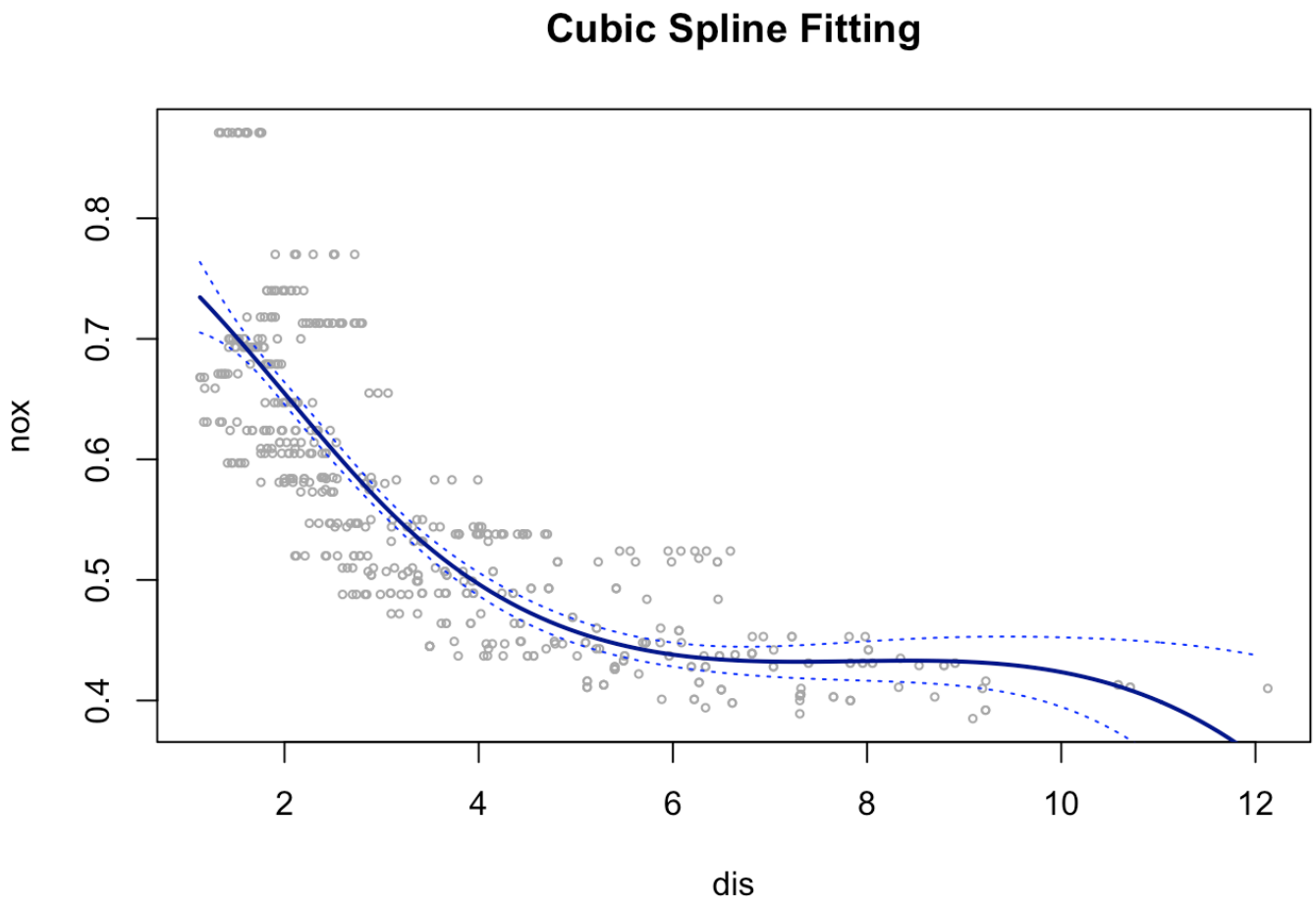
A cubic spline with K knots woulld have K+4 parameters or degrees of freedom. Therefore, with four degrees of freedom we can only have a cubic spline with zero knot.

```
library(splines)
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
```

```
preds = predict(cs.fit, newdata = list(dis = dis.grid), se = TRUE)
se.bands = cbind(preds$fit + 2*preds$se.fit, preds$fit - 2*preds$se.fit)
```

```
plot(dis, nox, xlim = dislims, cex = 0.5, col = "darkgrey")
title("Cubic Spline Fitting")
lines(dis.grid, preds$fit, lwd = 2, col = "darkblue")
matlines(dis.grid, se.bands, lwd = 1, col = "blue", lty = 3)
```



Part E)

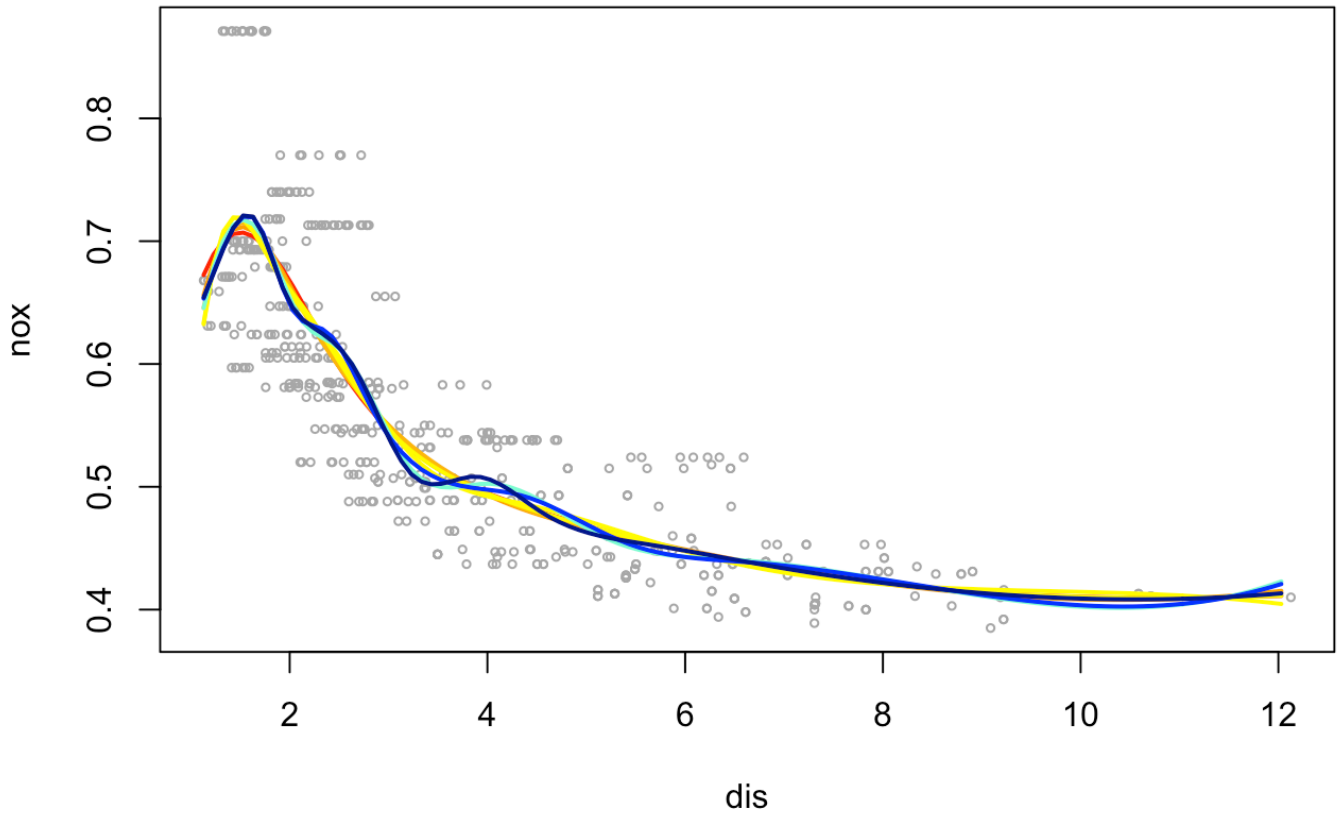
We first plot out the cubic spline regressions of varying degrees (from 4-12) on the same plot. As below:

```
colors = c("red", "orange", "darkgoldenrod1", "yellow", "yellow", "aquamarine", "b
lue",
           "dark blue", "purple", "black")

plot(dis, nox, xlim = dislims, cex = 0.5, col = "darkgrey")

for (i in 4:12) {
  cs.fit = lm(nox ~ bs(dis, i))
  preds = predict(cs.fit, newdata = list(dis = dis.grid), se = T)

  lines(dis.grid, preds$fit, lwd = 2, col = colors[i-4])
}
```

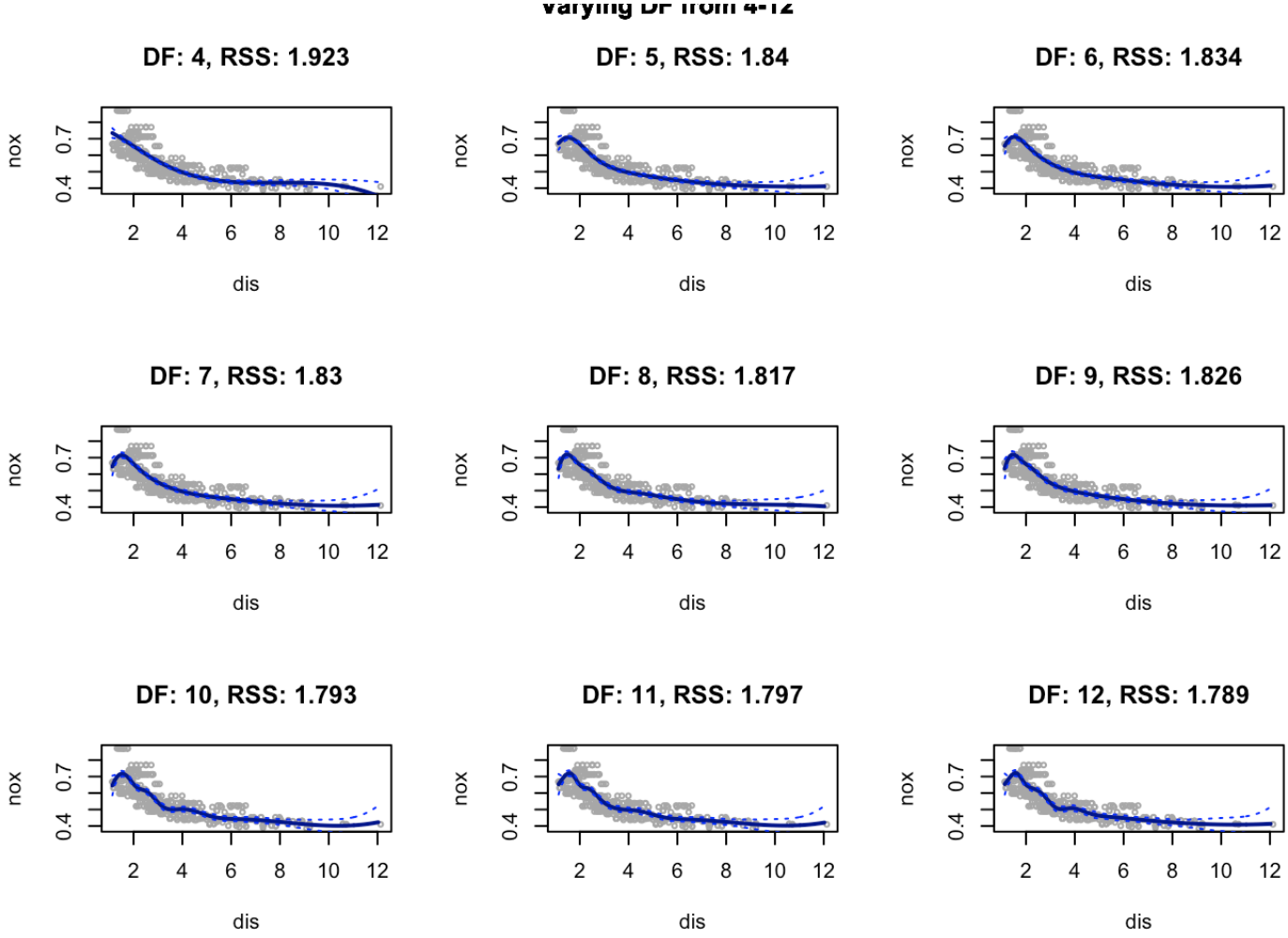


It appears to be a little chunked and difficult to tell which line is which, so I decide to plot out the splines side by side, as below:

```
par(mfrow = c(3,3))

for (i in 4:12) {
  cs.fit = lm(nox ~ bs(dis, df = i))
  preds = predict(cs.fit, newdata = list(dis = dis.grid), se = TRUE)
  se.bands = cbind(preds$fit + 2*preds$se.fit, preds$fit - 2*preds$se.fit)

  plot(dis, nox, xlim = dislims, cex = 0.5, col = "darkgrey")
  title("Varying DF from 4-12", outer = TRUE)
  title(sprintf("DF: %s, RSS: %s", i, round(sum(cs.fit$residuals^2), 3)))
  lines(dis.grid, preds$fit, lwd = 2, col = "darkblue")
  matlines(dis.grid, se.bands, lwd = 1, col = "blue", lty = 3)
}
```



(You may refer to the title of plots for number of knots and RSS information.)

In plots above I made 9 cubic splines with varying degrees of freedom, starting from 4 ending at 12. We can tell that for cubic spline models with higher degrees of freedom (more knots), the model generally has less RSS. However, they typically are more wiggly than models with few degrees of freedom at the top left of the graph.

This confirms the bias-variance trade off where more flexible complex models (cubic splines with higher degrees of freedom, more knots) have lower bias (lower RSS) but higher variance (wiggly) in prediction.

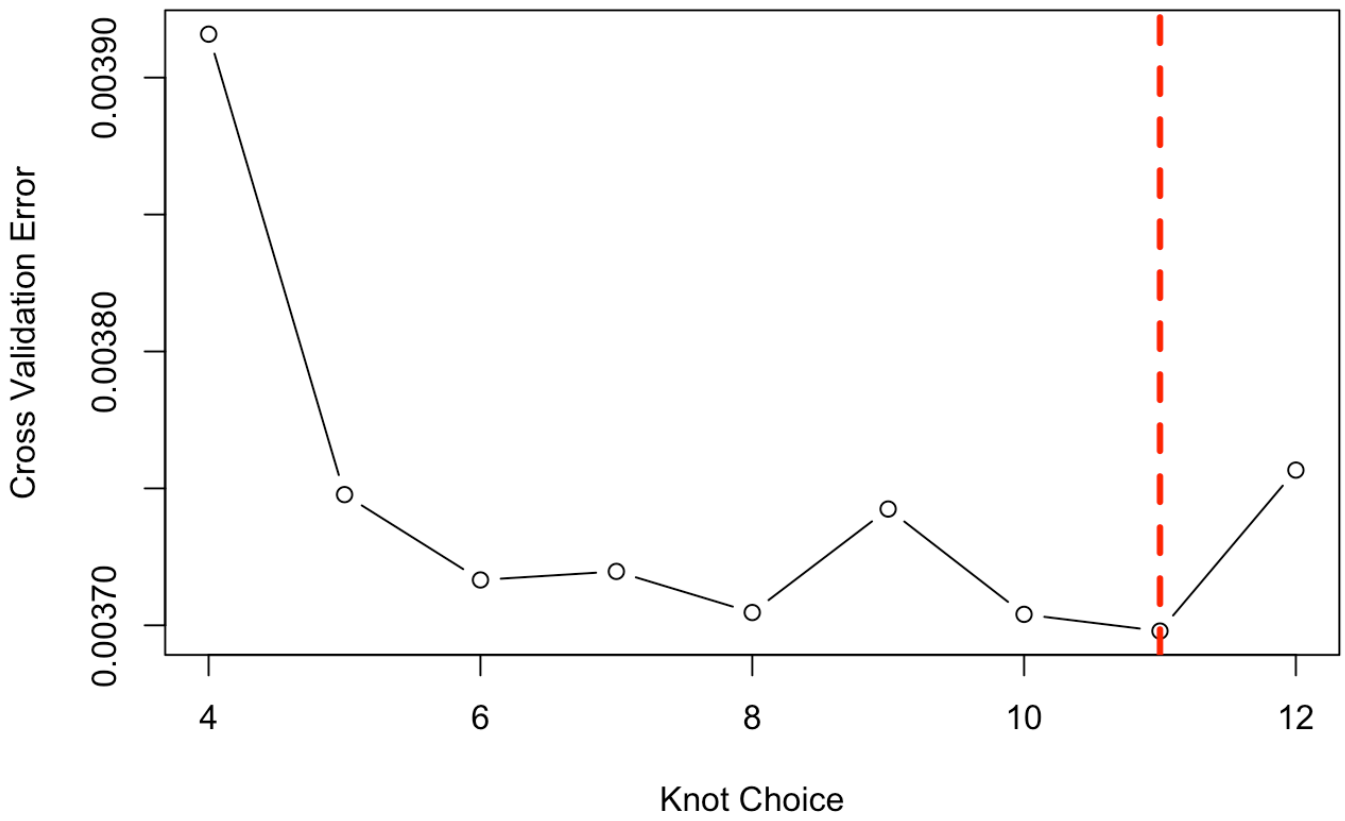
Part F)

```
library(boot)
set.seed(325)
df=4:12
cv.error=rep(0,9)
for (i in df){
    cs.fit = glm(nox ~ bs(dis, df = i), data = Boston)
    cv.error[i-3] = cv.glm(Boston,cs.fit, K=10)$delta[1]
}
cv.error
```

```
## [1] 0.003915860 0.003747738 0.003716571 0.003719681 0.003704685 0.003742470
## [7] 0.003704016 0.003697914 0.003756672
```

```
cv.errormat = data.frame(matrix(NA,nrow=9, ncol=2))
cv.errormat[1] = cv.error
cv.errormat[2] = 4:12
names(cv.errormat)[names(cv.errormat) == "X1"] <- "cv.error"
names(cv.errormat)[names(cv.errormat) == "X2"] <- "knot.choice"
```

```
plot(cv.errormat$knot.choice, cv.errormat$cv.error, type = "b", xlim = range(cv.er
rormat$knot.choice),
     xlab = "Knot Choice", ylab = "Cross Validation Error")
abline(v = 11, col = "red", lwd=3, lty=2)
```




```
min(cv.error)
```

```
## [1] 0.003697914
```

According to the 10-fold cross-validation approach, having a cubic spline with degree = 11 would results in the minimal validation error.

However, since the difference between cross validation error is very minimal (within 0.0001), I would actually prefer small knot choice for not over fitting the data. So realistically speaking, I would prefer 6-8 knots instead of 11 knots.