# Bruce Campbell ST-617 Homework 2

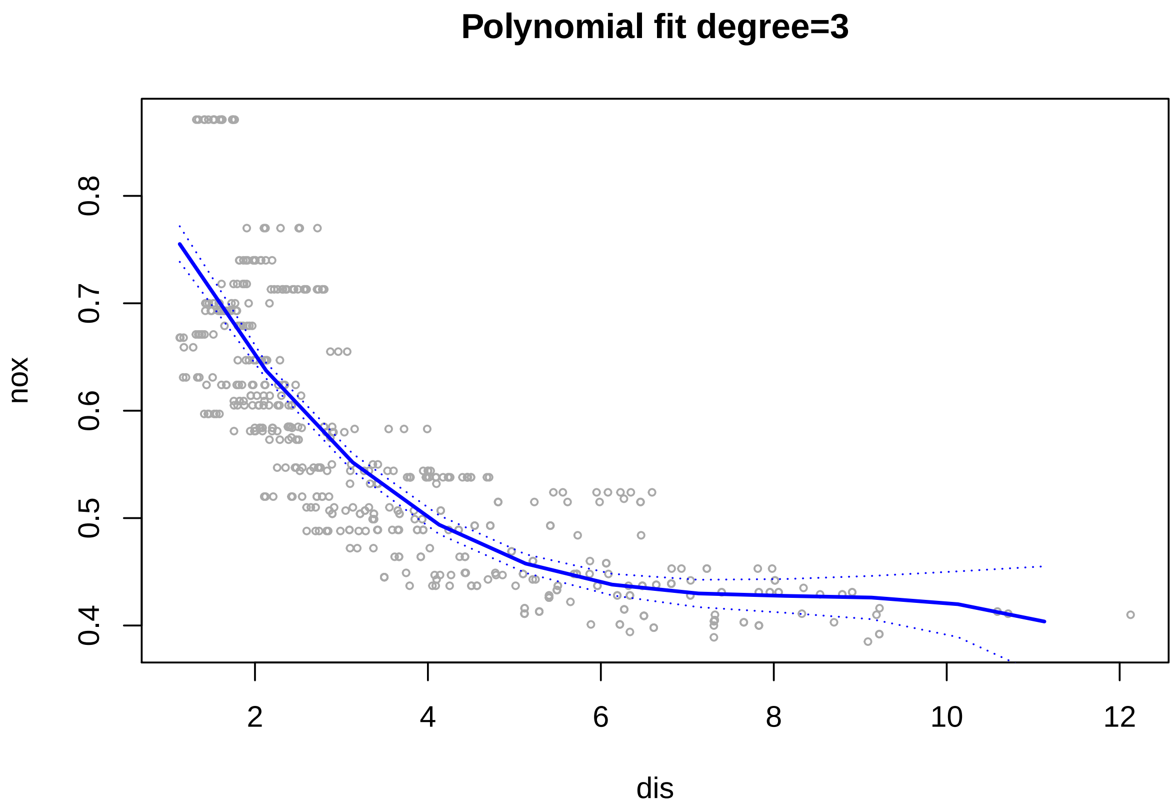
Tue Jul 12 23:35:02 2016

## Chapter 7

### Problem 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.

```
rm(list = ls())
library(MASS)
attach(Boston)
lmpoly.fit <- lm(nox ~ poly(dis, degree = 3))

dislims = range(dis)
dis.grid = seq(from = dislims[1], to = dislims[2])

preds = predict(lmpoly.fit, newdata = data.frame(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 ")

lines(dis.grid, preds$fit, lwd = 2, col = " blue")
matlines(dis.grid, se.bands, lwd = 1, col = " blue", lty = 3)
title("Polynomial fit degree=3")
```

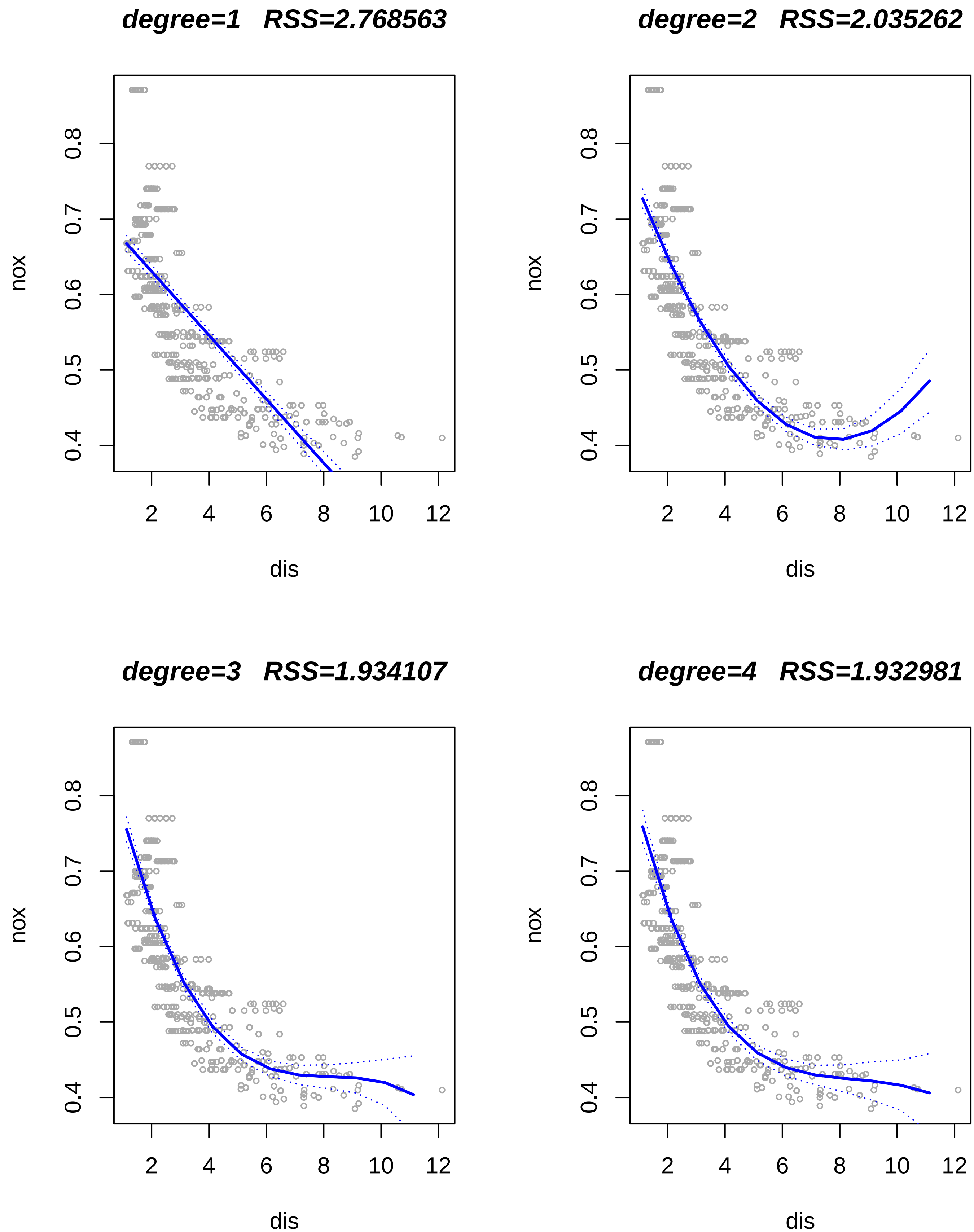


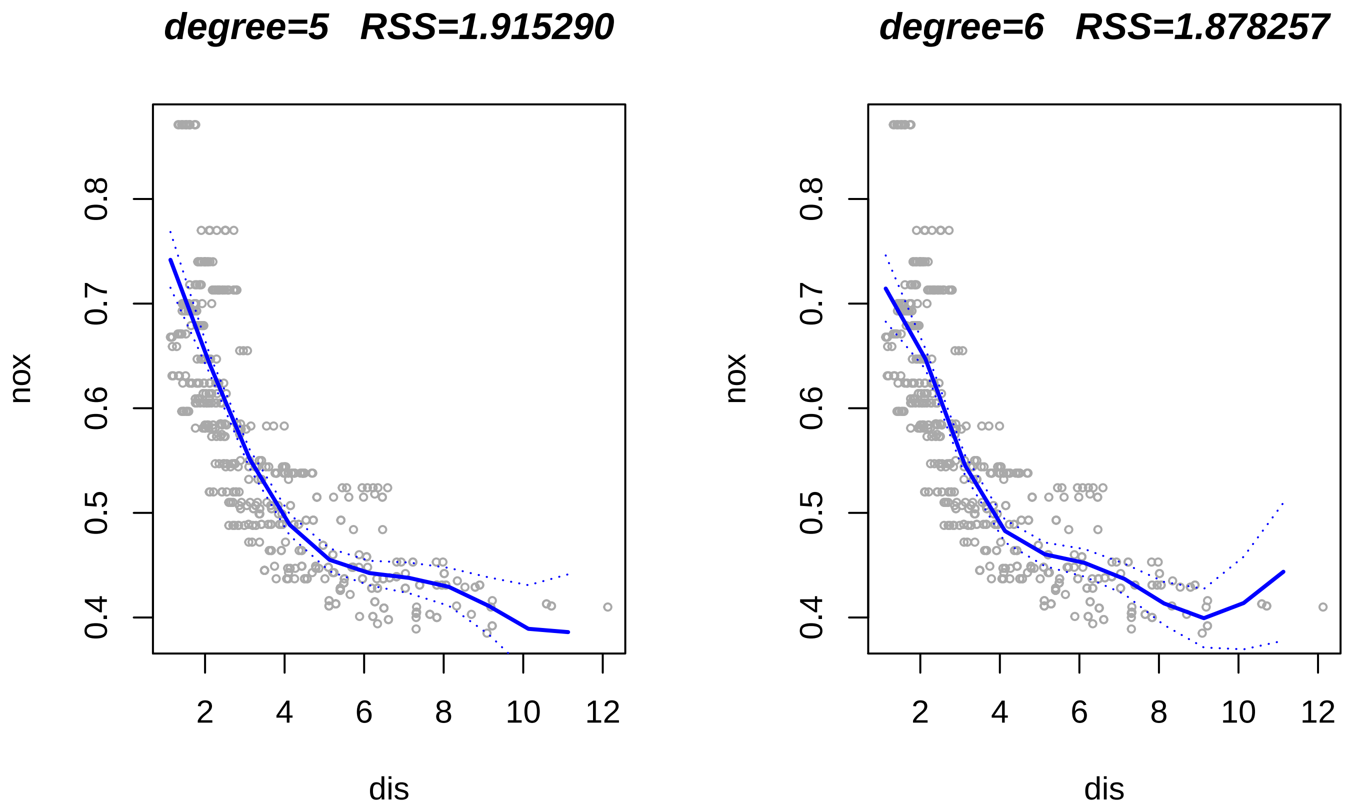

**b)**

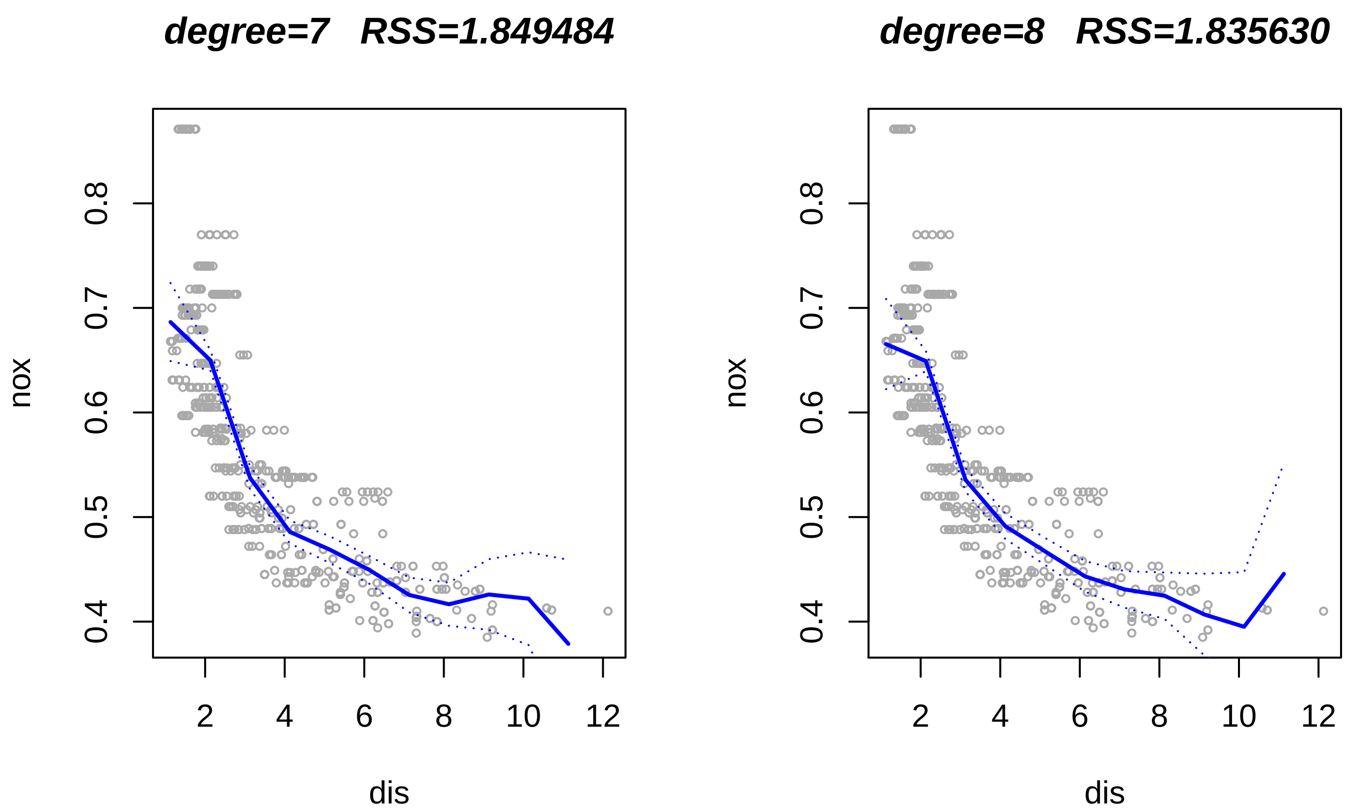
Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

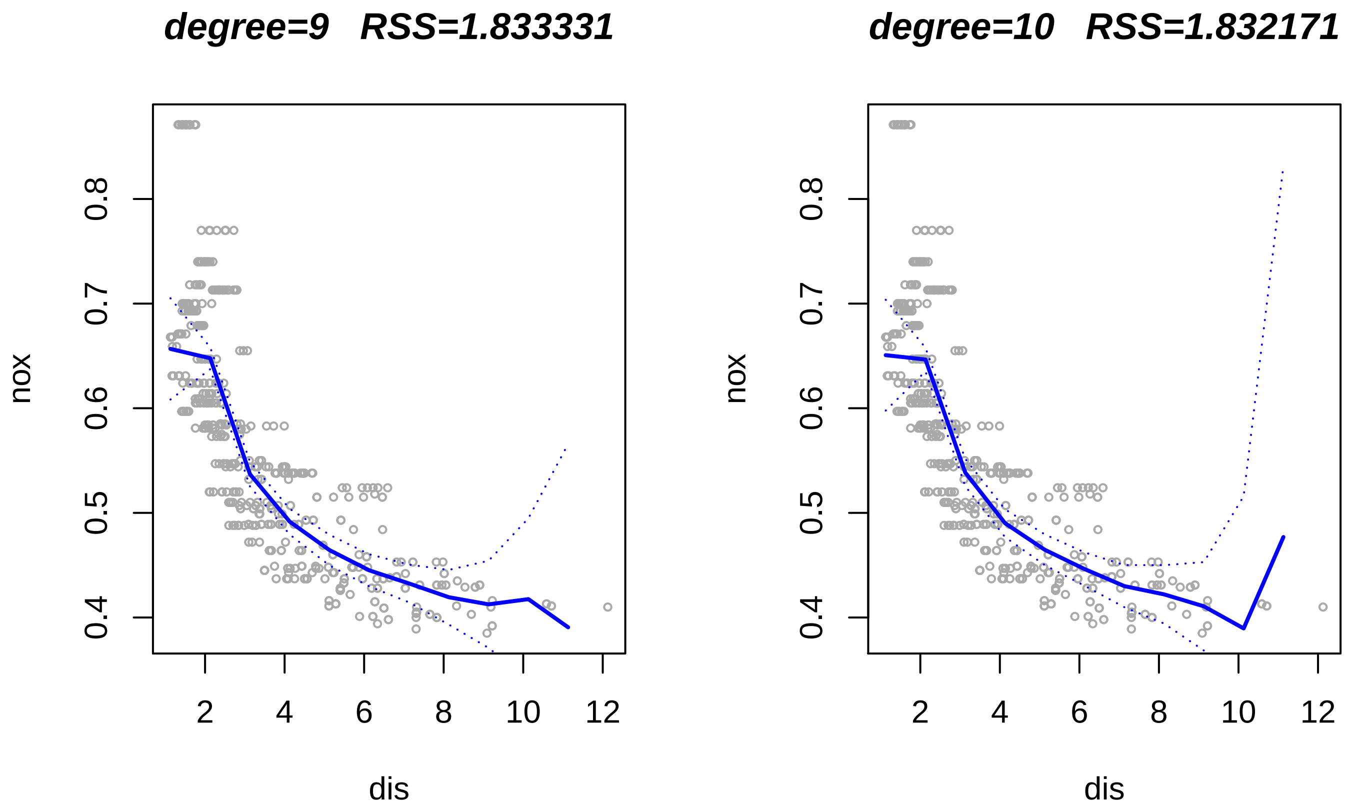
```
max.poly <- 10
nox <- Boston$nox
dis <- Boston$dis
plotFunc <- function(dis, nox, degree_val) {
    lmpoly.fit <- lm(nox ~ poly(dis, degree = degree_val))

    dislims = range(dis)
    dis.grid = seq(from = dislims[1], to = dislims[2])

    preds = predict(lmpoly.fit, newdata = data.frame(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 ")

    lines(dis.grid, preds$fit, lwd = 2, col = " blue")
    matlines(dis.grid, se.bands, lwd = 1, col = " blue", lty = 3)
    title(c(sprintf("degree=%d   RSS=%f", degree_val, sum(lmpoly.fit$residuals^2))),
        cex = 0.5, font.main = 4)
```

```
}

for (i in 1:max.poly) {
    plotFunc(dis, nox, degree_val = i)
}
```


degree=1 RSS=2.768563
degree=2 RSS=2.035262
degree=3 RSS=1.934107
degree=4 RSS=1.932981
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12


degree=5 RSS=1.915290
degree=6 RSS=1.878257
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12


degree=7 RSS=1.849484
nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
dis
degree=8 RSS=1.835630
nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
dis




**c)**

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

```
max.poly <- 10

dis = Boston$dis
nox = Boston$nox


cvFunc <- function(dis, nox, degree) {
    preds <- numeric(length(dis))
    for (i in 1:length(dis)) {
        # Here is where we exclude i from the training set
        dis.in <- dis[-i]
        dis.out <- dis[i]

        # Single test point
        nox.in <- nox[-i]
        nox.out <- dis[i]

        fit <- lm(nox.in ~ poly(dis.in, degree = degree))
        preds[i] <- predict(fit, newdata = data.frame(dis.in = dis.out))
```

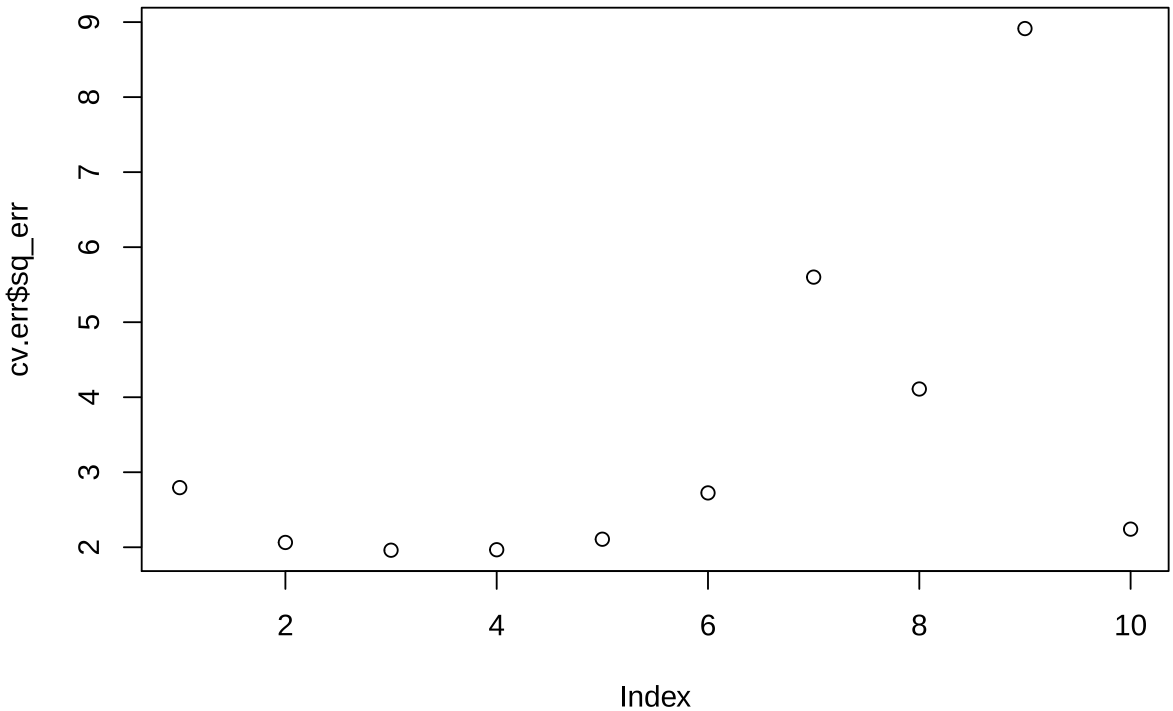
```
    }
    return(sum((nox - preds)^2))
}

cv.err <- data.frame(sq_err = numeric(max.poly))
for (i in 1:max.poly) {
    cv.err[i, 1] <- cvFunc(dis, nox, degree = i)
}

which.min(cv.err$sq_err)
```

```
## [1] 3
```

```
library(pander)
pander(cv.err)
```

| sq_err |
|---|
| 2.795 |
| 2.064 |
| 1.961 |
| 1.967 |
| 2.107 |
| 2.724 |
| 5.601 |
| 4.109 |
| 8.914 |
| 2.242 |

```
plot(cv.err$sq_err)
```

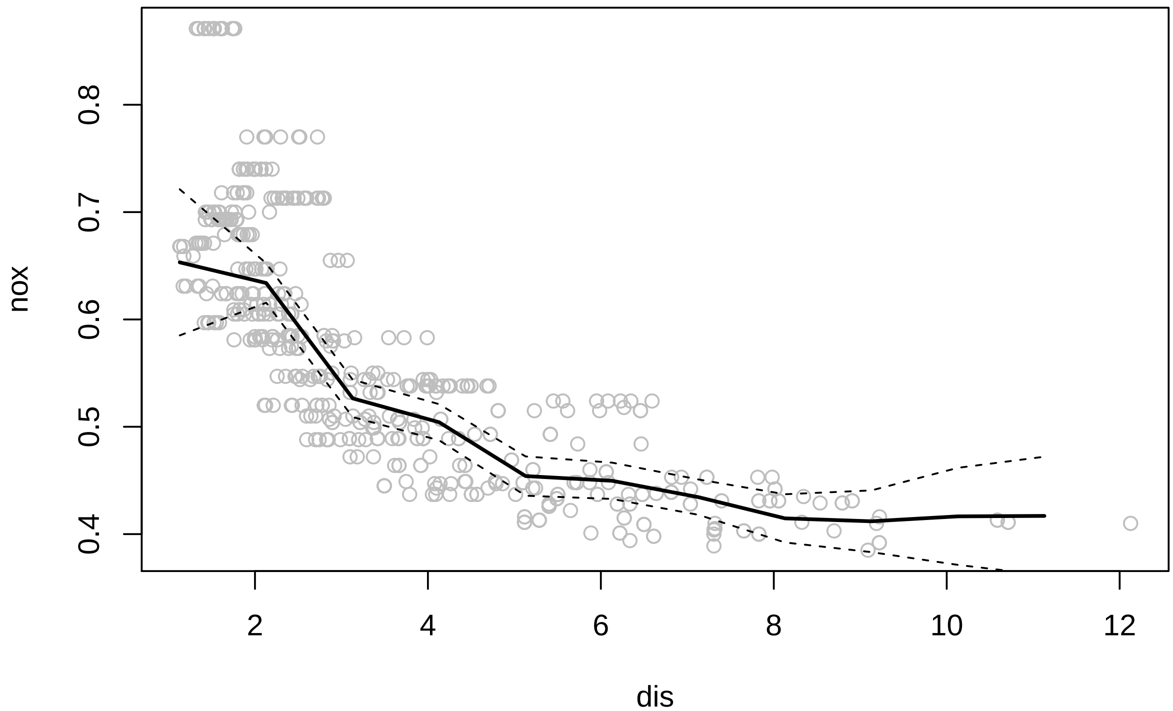



We've used leave one out cross validation above to select the best degree based on the $CV_n$ error rate. Interestingly, there is a marked drop in the $CV_n$ at degree of 10. The best degree based on the LOOCV algrithm is 3.

**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.

```
library(splines)
quantileLocs <- quantile(dis, ppoints(10))
fit = lm(nox ~ bs(dis, knots = quantileLocs), data = Boston)
dislims = range(dis)
dis.grid = seq(from = dislims[1], to = dislims[2])
pred = predict(fit, newdata = list(dis = dis.grid), se = T)
plot(dis, nox, col = " gray ")
lines(dis.grid, pred$fit, lwd = 2)
lines(dis.grid, pred$fit + 2 * pred$se, lty = "dashed")
lines(dis.grid, pred$fit - 2 * pred$se, lty = "dashed")
```

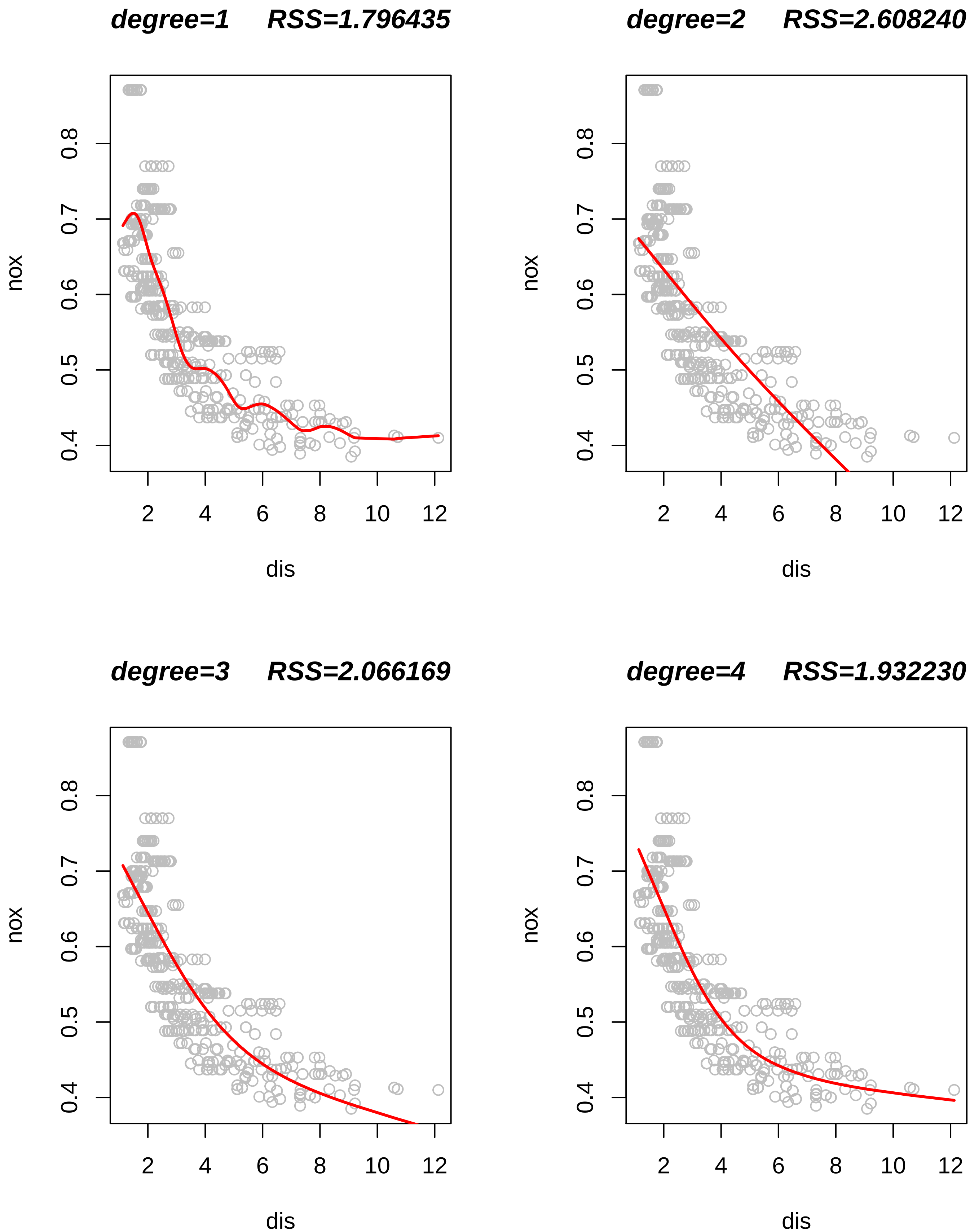


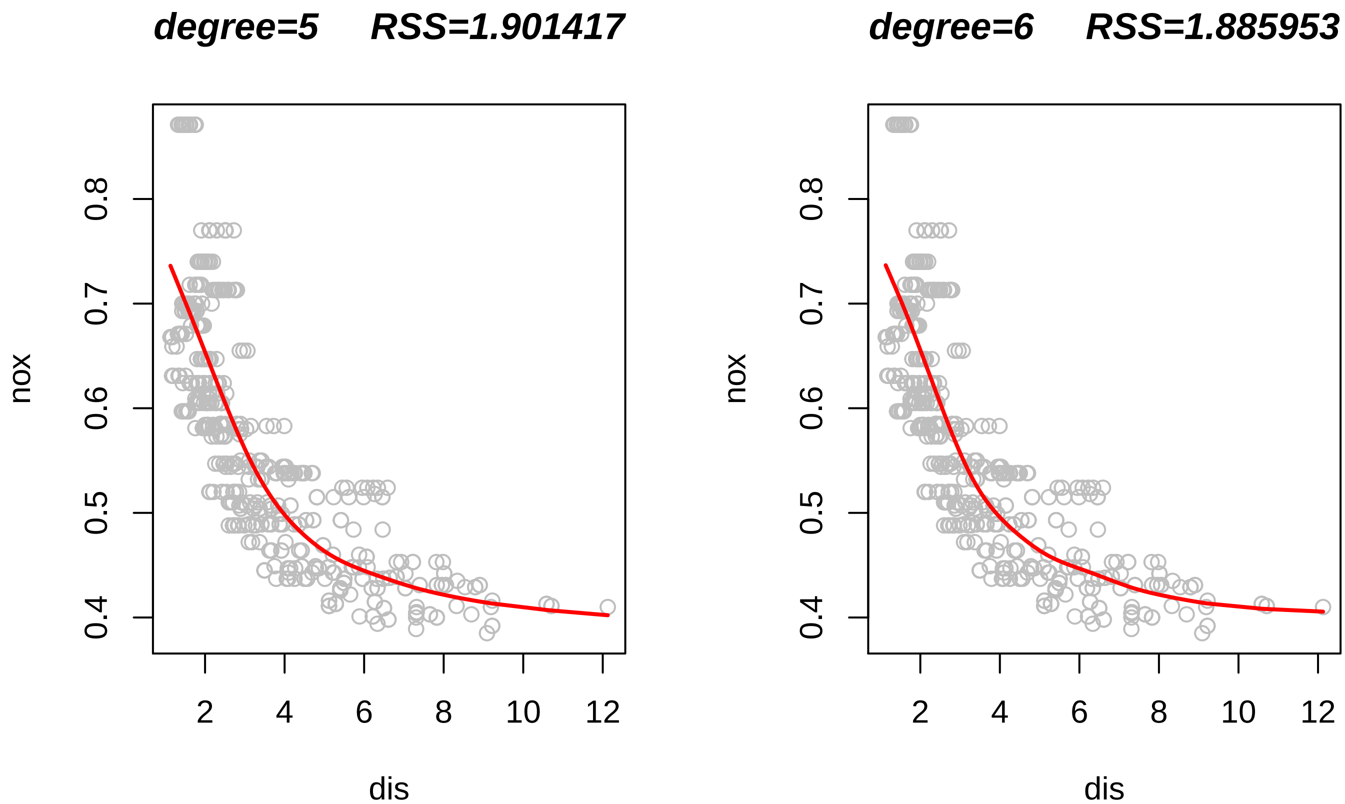

We chose the knots to be distributed accoring to the deciles of the data.

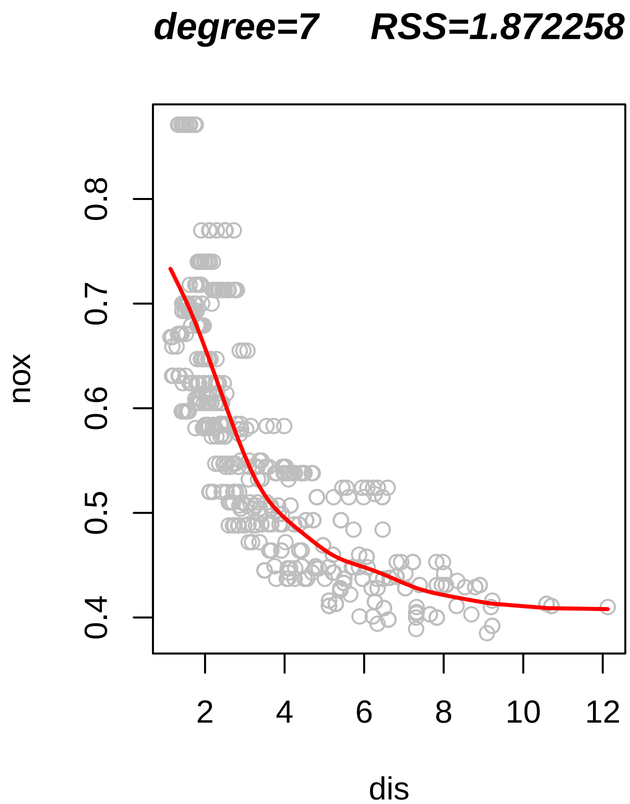
**e)**

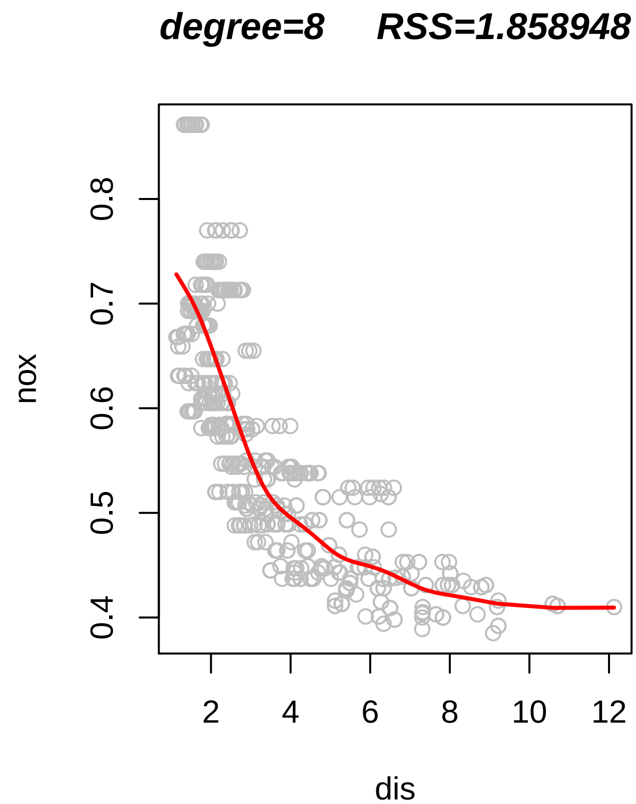
Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.

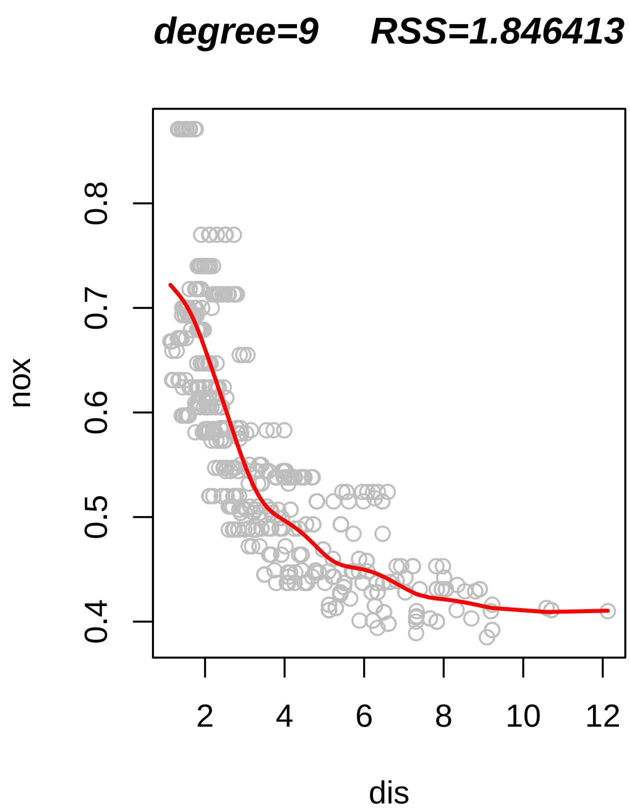
```
max.df <- 10
nox <- Boston$nox
dis <- Boston$dis
plotFunc <- function(dis, nox, degree_val) {
    fit = smooth.spline(dis, nox, df = degree_val)
    pred = predict(fit, se = T)
    plot(dis, nox, col = " gray ")
    lines(fit, col = "red ", lwd = 2)
    RSS <- sum(residuals(fit)^2)

    title(c(sprintf("degree=%d      RSS=%f", degree_val, RSS)), cex = 0.5, font.main = 4)

}

for (i in 1:max.poly) {
    plotFunc(dis, nox, degree_val = i)
}
```

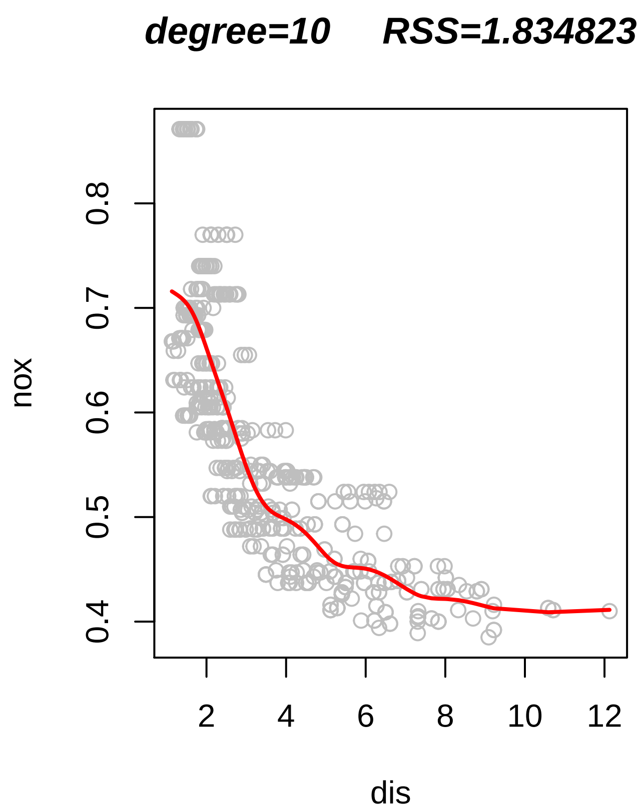

degree=1 RSS=1.796435
degree=2 RSS=2.608240
degree=3 RSS=2.066169
degree=4 RSS=1.932230
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12


degree=5 RSS=1.901417
degree=6 RSS=1.885953
nox
dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12


degree=7 RSS=1.872258
nox
0.4
0.5
0.6
0.7
0.8
dis
2
4
6
8
10
12


degree=8 RSS=1.858948
nox
0.4
0.5
0.6
0.7
0.8
dis
2
4
6
8
10
12




**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.

```
dis = Boston$dis
nox = Boston$nox

cvFunc <- function(dis, nox, degree_val) {
    preds <- numeric(length(dis))
    for (i in 1:length(dis)) {
        # Here is where we exclude i from the training set
        dis.in <- dis[-i]
        dis.out <- dis[i]

        # Single test point
        nox.in <- nox[-i]
        nox.out <- dis[i]

        fit = smooth.spline(dis.in, nox.in, df = degree_val)

        preds[i] <- predict(fit, dis.out)$y
    }
    return(sum((nox - preds)^2))
```

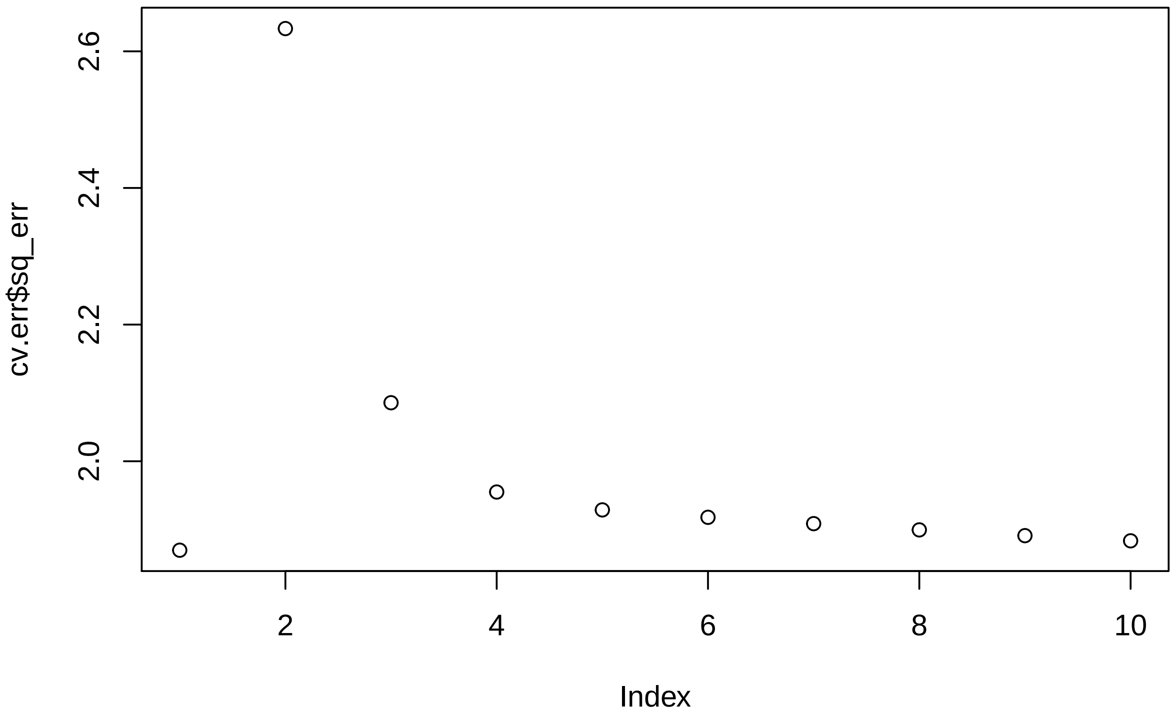
```r
}

cv.err <- data.frame(sq_err = numeric(max.poly))
for (i in 1:max.df) {
    cv.err[i, 1] <- cvFunc(dis, nox, degree_val = i)
}

which.min(cv.err$sq_err)
```

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

```r
library(pander)
pander(cv.err)
```

| sq_err |
|---|
| 1.87 |
| 2.633 |
| 2.086 |
| 1.955 |
| 1.929 |
| 1.918 |
| 1.909 |
| 1.9 |
| 1.891 |
| 1.884 |

```r
plot(cv.err$sq_err)
```




We've used leave one out cross validation above to select the best degree based on the $CV_n$ error rate. There are two condidates for best degree based on $CV_n$ degree 1 and degree 10.