Solutions to Assignment

Rongfei Jin

March 21, 2025

Conceptual 1

(a)

$$\begin{aligned} f(x) &= f_1(x) \\ \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)_+^3 &= a_1 + b_1 x + c_1 x^2 + d_1 x^3 \\ (\beta_0 - a_1) + (\beta_1 - b_1)x + (\beta_2 - c_1)x^2 + (\beta_3 - d_1)x^3 &= 0 \qquad (x-\xi)_+^3 = 0 \end{aligned}$$

$$\begin{aligned} \beta_0 &= a_1 \\ \beta_1 &= b_1 \\ \beta_2 &= c_1 \\ \beta_3 &= d_1 \end{aligned}$$

(b)

$$\begin{aligned} f(x) &= f_2(x) \\ \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)_+^3 &= a_2 + b_2 x + c_2 x^2 + d_2 x^3 \\ \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x^3 - 3\xi x^2 + 3\xi^2 x - \xi^3) &= a_2 + b_2 x + c_2 x^2 + d_2 x^3 \\ (\beta_0 - a_2 - \beta_4\xi^3) + (\beta_1 - b_2 + 3\beta_4\xi^2)x + (\beta_2 - c_2 - 3\beta_4\xi)x^2 + (\beta_3 - d_2 + \beta_4)x^3 &= 0 \end{aligned}$$

$$\begin{aligned} a_2 &= \beta_0 - \beta_4\xi^3 \\ b_2 &= \beta_1 + 3\beta_4\xi^2 \\ c_2 &= \beta_2 - 3\beta_4\xi \\ d_2 &= \beta_3 + \beta_4 \end{aligned}$$

(c)

$$\begin{aligned} f_2(\xi) &= \beta_0 - \beta_4\xi^3 + (\beta_1 + 3\beta_4\xi^2)\xi + (\beta_2 - 3\beta_4\xi)\xi^2 + (\beta_3 + \beta_4)\xi^3 \\ &= \beta_0 + \beta_1\xi + \beta_2\xi^2 + \beta_3\xi^3 \\ &= f_1(\xi) \end{aligned}$$

(d)

$$\begin{aligned} f_1'(\xi) &= \beta_1 + 2\beta_2\xi + 3\beta_3\xi^2 \\ f_2'(\xi) &= \beta_1 + 3\beta_4\xi^2 + 2(\beta_2 - 3\beta_4\xi)\xi + 3(\beta_3 + \beta_4)\xi^2 \\ &= \beta_1 + 2\beta_2\xi + 3\beta_3\xi^2 \\ &= f_1'(\xi) \end{aligned}$$

(e)

$$\begin{aligned}
f_1''(x) &= 2\beta_2 + 6\beta_3 x \\
f_2''(x) &= 2\beta_2 - 6\beta_4\xi + 6(\beta_3 + \beta_4)\xi^2 \\
&= f_1''(x)
\end{aligned}$$

Conceptual 2

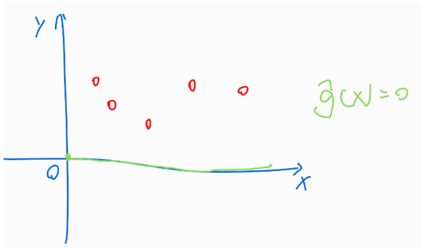

Figure 1: (a)

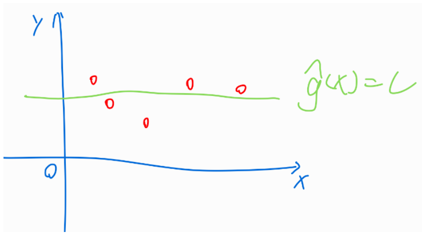

Figure 2: (b)

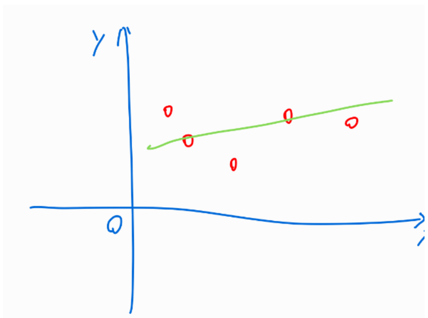

Figure 3: (c)

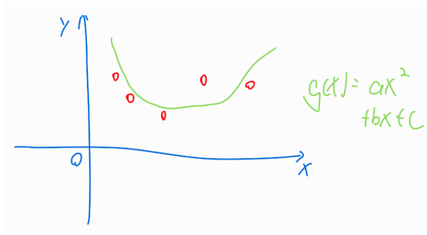

Figure 4: (d)

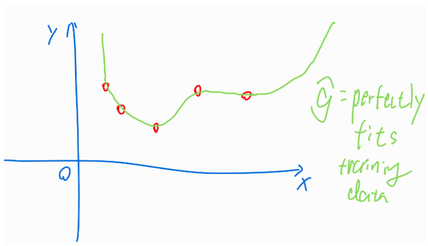

Figure 5: (e)

Conceptual 3

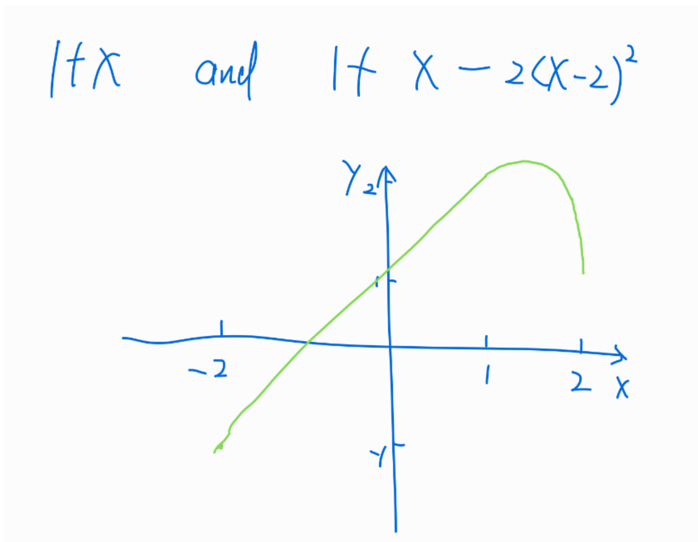

Figure 6: 3

Conceptual 5

(a) g2 because it is more flexible and can fit the data better.

(b) cannot be determined. However, if g2 overfits then g1 will perform better.

(c) same since the penalty term

Applied 6

a

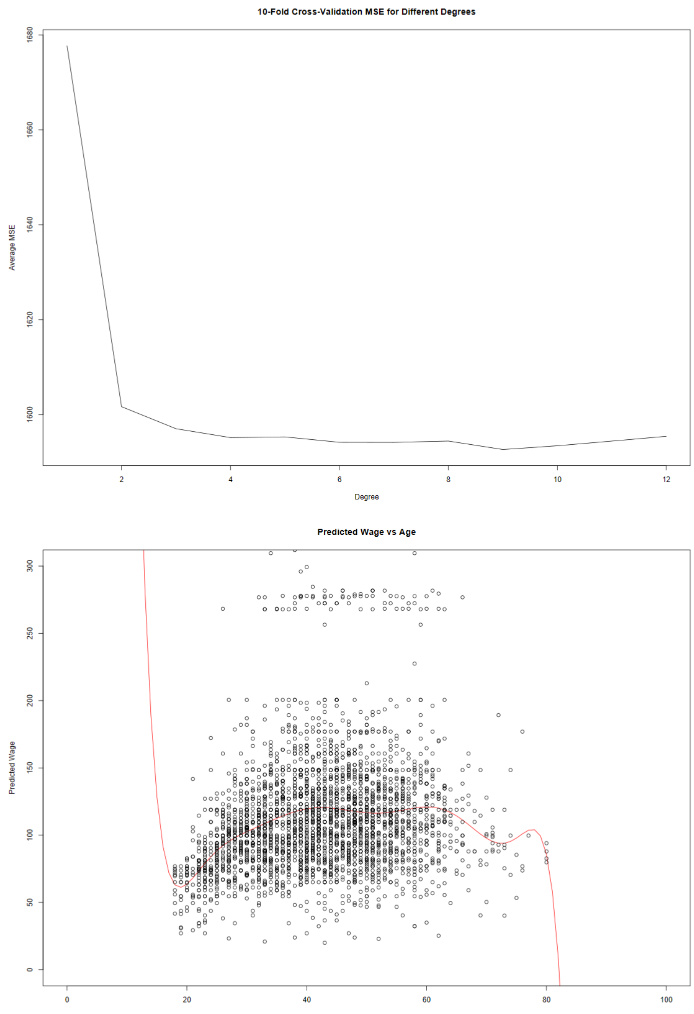

Figure 7

```r
library(ISLR2)

data(Wage)
min(Wage$age)
max(Wage$age)

cv.poly <- function(k = 10, Wage, degs) {
  # get fixed indices for folds and shuffle
  indices <- c(1:nrow(Wage))
  indices <- sample(indices)

  # split into k folds
  folds <- split(indices, rep(1:k, length.out = length(indices)))

  avg_mse <- c()
  for (deg in degs) {
```

```r
    mse <- c()
    for (fold in folds) {
      fit <- lm(wage ~ poly(age, degree = deg), data = Wage[-fold, ])
      pred <- predict(fit, Wage[fold, ])
      mse <- c(mse, mean((pred - Wage[fold, ]$wage)^2))
    }
    avg_mse <- c(avg_mse, mean(mse))
  }

  # find the best index
  best_index <- which.min(avg_mse)
  return(list(best_deg = degs[best_index], avg_mse = avg_mse, degs = degs))
}

cv_results <- cv.poly(k = 10, Wage, degs = 2:12)
cv_results$best_deg
# best degree is 9

plot(cv_results$avg_mse ~ cv_results$degs, type = "l", xlab = "Degree", ylab = "Average MSE", main = "10-Fold

# best model
best_model <- lm(wage ~ poly(age, degree = 9), data = Wage)
summary(best_model)

pred <- predict(best_model, data.frame(age = seq(0, 100, 1)))

# xlim from 0 to 100
# ylim from 0 to 300
plot(pred ~ seq(0, 100, 1), xlab = "Age", ylab = "Predicted Wage", main = "Predicted Wage vs Age", type = "l"
points(Wage$age, Wage$wage)



models <- lapply(1:12, function(deg) lm(wage ~ poly(age, degree = deg), data = Wage))
# anova all models
do.call(anova, models)
```

```
Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)              111.7036     0.7282 153.395  < 2e-16 ***
poly(age, degree = 9)1   447.0679    39.8857  11.209  < 2e-16 ***
poly(age, degree = 9)2  -478.3158    39.8857 -11.992  < 2e-16 ***
poly(age, degree = 9)3   125.5217    39.8857   3.147  0.00167 **
poly(age, degree = 9)4   -77.9112    39.8857  -1.953  0.05087 .
poly(age, degree = 9)5   -35.8129    39.8857  -0.898  0.36932
poly(age, degree = 9)6    62.7077    39.8857   1.572  0.11601
poly(age, degree = 9)7    50.5498    39.8857   1.267  0.20512
poly(age, degree = 9)8   -11.2547    39.8857  -0.282  0.77783
poly(age, degree = 9)9   -83.6918    39.8857  -2.098  0.03596 *
```

Figure 8

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)	
1	2998	5022216					
2	2997	4793430	1	228786	143.6709	< 2.2e-16	***
3	2996	4777674	1	15756	9.8941	0.001674	**
4	2995	4771604	1	6070	3.8119	0.050984	.
5	2994	4770322	1	1283	0.8054	0.369552	
6	2993	4766389	1	3932	2.4693	0.116192	
7	2992	4763834	1	2555	1.6046	0.205345	
8	2991	4763707	1	127	0.0795	0.777935	
9	2990	4756703	1	7004	4.3985	0.036054	*
10	2989	4756701	1	3	0.0017	0.967540	
11	2988	4756597	1	103	0.0648	0.799070	
12	2987	4756591	1	7	0.0043	0.947903	

Figure 9

the best degree is 9. The anova tests show that only degree 2 3 and 9 significantly improve the degree 1 model. This is consistent with summary output

b

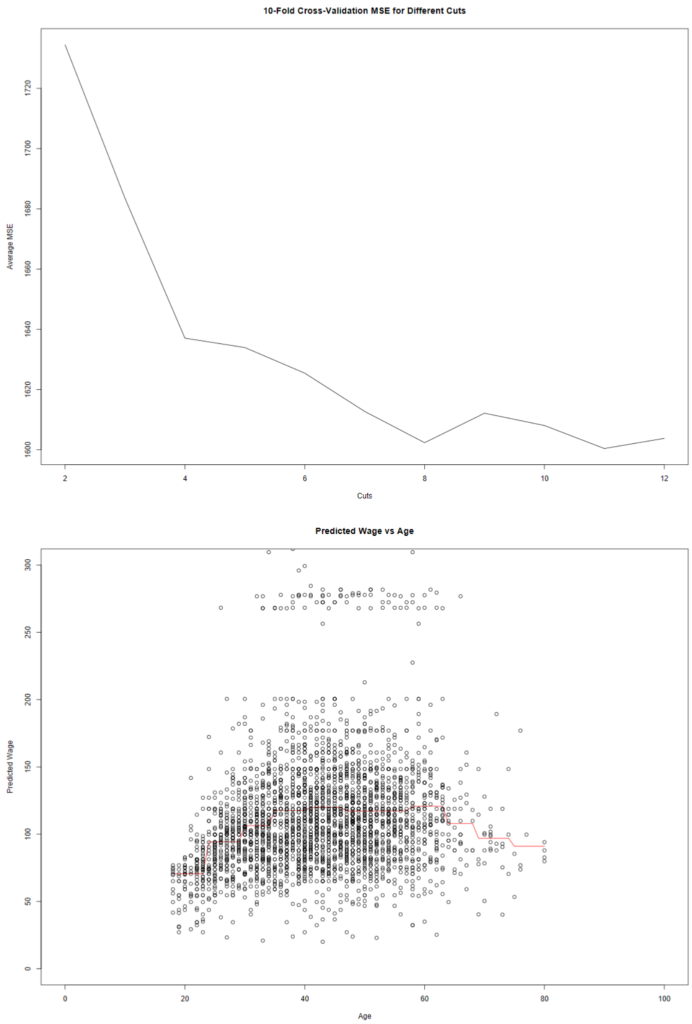

Figure 10

```r
library(ISLR2)

data(Wage)
min(Wage$age)
max(Wage$age)

fit_step_function <- function(x, y, k) {
  breaks <- seq(min(x), max(x), length.out = k + 1)

  intervals <- cut(x, breaks = breaks, include.lowest = TRUE)

  step_heights <- tapply(y, intervals, mean)

  predict_step <- function(new_x) {
    new_intervals <- cut(new_x, breaks = breaks, include.lowest = TRUE)
    return(step_heights[as.character(new_intervals)])
  }
```

```
  return(list(
    breaks = breaks,
    heights = step_heights,
    predict = predict_step
  ))
}

cv.step <- function(k = 10, Wage, cuts) {
  # get fixed indices for folds and shuffle
  indices <- c(1:nrow(Wage))
  indices <- sample(indices)

  # split into k folds
  folds <- split(indices, rep(1:k, length.out = length(indices)))

  avg_mse <- c()
  for (cut in cuts) {
    mse <- c()
    for (fold in folds) {
      fit <- fit_step_function(Wage$age[-fold], Wage$wage[-fold], cut)
      pred <- fit$predict(Wage[fold, ]$age)
      mse <- c(mse, mean((pred - Wage[fold, ]$wage)^2))
    }
    avg_mse <- c(avg_mse, mean(mse))
  }

  # find the best index
  best_index <- which.min(avg_mse)
  return(list(best_cuts = cuts[best_index], avg_mse = avg_mse, cuts = cuts))
}

cv_results <- cv.step(k = 10, Wage, cuts = 2:12)
cv_results$best_cuts
# best degree is 9

plot(cv_results$avg_mse ~ cv_results$cuts, type = "l", xlab = "Cuts", ylab = "Average MSE", main = "10-Fold C

# best model
best_model <- fit_step_function(Wage$age, Wage$wage, cv_results$best_cuts)

pred <- best_model$predict(seq(0, 100, 1))

# xlim from 0 to 100
# ylim from 0 to 300
plot(pred ~ seq(0, 100, 1), xlab = "Age", ylab = "Predicted Wage", main = "Predicted Wage vs Age", type = "l"
points(Wage$age, Wage$wage)


models <- lapply(1:12, function(deg) lm(wage ~ poly(age, degree = deg), data = Wage))
# anova all models
do.call(anova, models)
```

The best number of cuts is 11

Applied 8

```r
library(ISLR2)
library(splines)

pairs(Auto)

# sort the data by acceleration
Auto <- Auto[order(Auto$acceleration), ]

plot(mpg ~ acceleration, data = Auto)

fit_lm <- lm(mpg ~ acceleration, data = Auto)

# fit a bs spline to the data
fit_bs <- lm(mpg ~ bs(acceleration, df = 4), data = Auto)

# fit a natural spline to the data
fit_ns <- lm(mpg ~ ns(acceleration, df = 4), data = Auto)

# fit a polynomial to the data
fit_poly <- lm(mpg ~ poly(acceleration, degree = 4), data = Auto)


# plot the data
par(mfrow = c(1, 1))
plot(mpg ~ acceleration, data = Auto)

# plot the fitted lines
lines(Auto$acceleration, predict(fit_bs), col = "red")
lines(Auto$acceleration, predict(fit_ns), col = "blue")
lines(Auto$acceleration, predict(fit_poly), col = "green")

# legend
legend("topright", legend = c("bs df = 4", "ns df = 4", "poly df = 4"), col = c("red", "blue", "green"), lty


# normality plot
par(mfrow = c(2, 1))
plot(fit_lm, which = 1)
plot(fit_lm, which = 2)
```

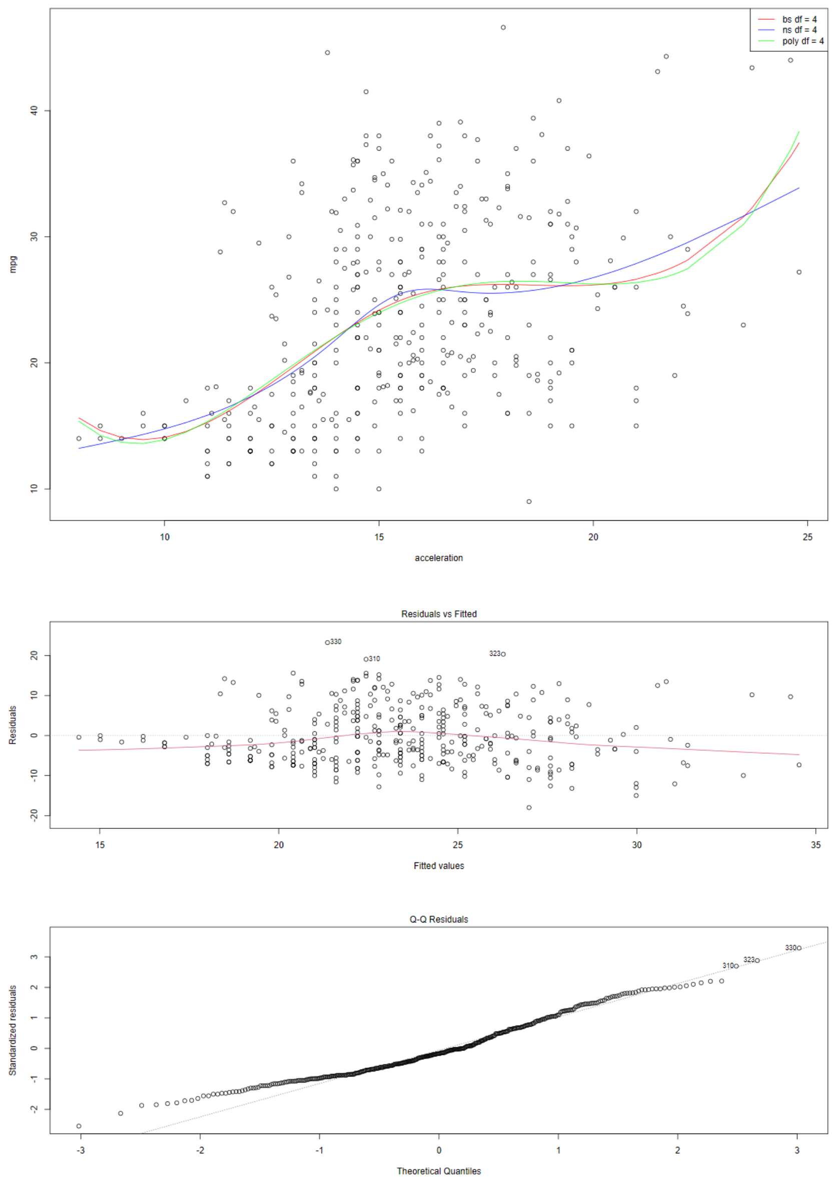

Figure 11

The normality plot shows that the some residuals are not normally distributed.

Applied 9

```
library(ISLR)

str(Boston)
# sort by dis
Boston <- Boston[order(Boston$dis), ]

# fit polynomial regression
fit_poly <- lm(nox ~ poly(dis, degree = 3), data = Boston)
summary(fit_poly)
par(mfrow = c(1, 1))
plot(Boston$dis, Boston$nox)
lines(Boston$dis, predict(fit_poly, Boston), col = "red")
title("Polynomial fit with 3 degrees of freedom")


# fit 1 to 10 degree polynomials

color_map <- rainbow(10)
legend_text <- vector(mode = "character", length = 10)

plot(Boston$dis, Boston$nox)
for (deg in 1:10) {
  fit <- lm(nox ~ poly(dis, degree = deg), data = Boston)
  ssr <- sum(summary(fit)$residuals^2)
  color <- color_map[deg]
  lines(Boston$dis, predict(fit, Boston), col = color)
  legend_text[deg] <- paste("Degree", deg, "SSR:", ssr)
}
legend_text
legend("topright", legend = legend_text, col = color_map, lty = 1)
title("Polynomial fits with degrees 1 to 10")


cv.poly <- function(k = 10, df, degs) {
  # get fixed indices for folds and shuffle
  indices <- c(1:nrow(df))
  indices <- sample(indices)

  # split into k folds
  folds <- split(indices, rep(1:k, length.out = length(indices)))

  avg_mse <- c()
  for (deg in degs) {
    mse <- c()
    for (fold in folds) {
      fit <- lm(nox ~ poly(dis, degree = deg), data = df[-fold, ])
      pred <- predict(fit, df[fold, ])
      mse <- c(mse, mean((pred - df[fold, ]$nox)^2))
    }
    avg_mse <- c(avg_mse, mean(mse))
  }

  # find the best index
  best_index <- which.min(avg_mse)
  return(list(best_deg = degs[best_index], avg_mse = avg_mse, degs = degs))
```

```r
}

cv_results <- cv.poly(k = 10, df = Boston, degs = 1:13)
cv_results$best_deg

fit.bs <- lm(nox ~ bs(dis, df = 4, knots = c(3)), data = Boston)
plot(Boston$dis, Boston$nox)
lines(Boston$dis, predict(fit.bs, Boston), col = "red")
title("B-spline fit with 4 degrees of freedom and knots at 3")

# Knot is chosen by observing a change of curvature in the plot of nox vs dis

range_of_df <- c(3:6)
par(mfrow = c(2, 2))
for (df in range_of_df) {
  fit <- lm(nox ~ bs(dis, df = df), data = Boston)
  ssr <- sum(summary(fit)$residuals^2)
  plot(Boston$dis, Boston$nox)
  lines(Boston$dis, predict(fit, Boston), col = "red")
  title(paste("B-spline fit with", df, "degrees of freedom and knots at 3", "SSR:", ssr))
}


cv.bs <- function(k = 10, df, degs) {
  # get fixed indices for folds and shuffle
  indices <- c(1:nrow(df))
  indices <- sample(indices)

  # split into k folds
  folds <- split(indices, rep(1:k, length.out = length(indices)))

  avg_mse <- c()
  for (deg in degs) {
    mse <- c()
    for (fold in folds) {
      fit <- lm(nox ~ bs(dis, df = deg), data = df[-fold, ])
      pred <- predict(fit, df[fold, ])
      mse <- c(mse, mean((pred - df[fold, ]$nox)^2))
    }
    avg_mse <- c(avg_mse, mean(mse))
  }

  # find the best index
  best_index <- which.min(avg_mse)
  return(list(best_deg = degs[best_index], avg_mse = avg_mse, degs = degs))
}

cv_results_bs <- cv.bs(k = 10, df = Boston, degs = 4:10)
cv_results_bs$best_deg
```

a, b, c

```
lm(formula = nox ~ poly(dis, degree = 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, degree = 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
poly(dis, degree = 3)2  0.856330   0.062071  13.796  < 2e-16 ***
poly(dis, degree = 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
```

Figure 12

The best degree is 3 from 10 folds cross validation with mse as the metric.

d

The knots is chosen at 3 because it is where the curvature of the plot changes.

e, f

Higher degrees of freedom leads to overfitting. the change of curvature near x=2 for df ¿ 4 does not reflect the true trend.

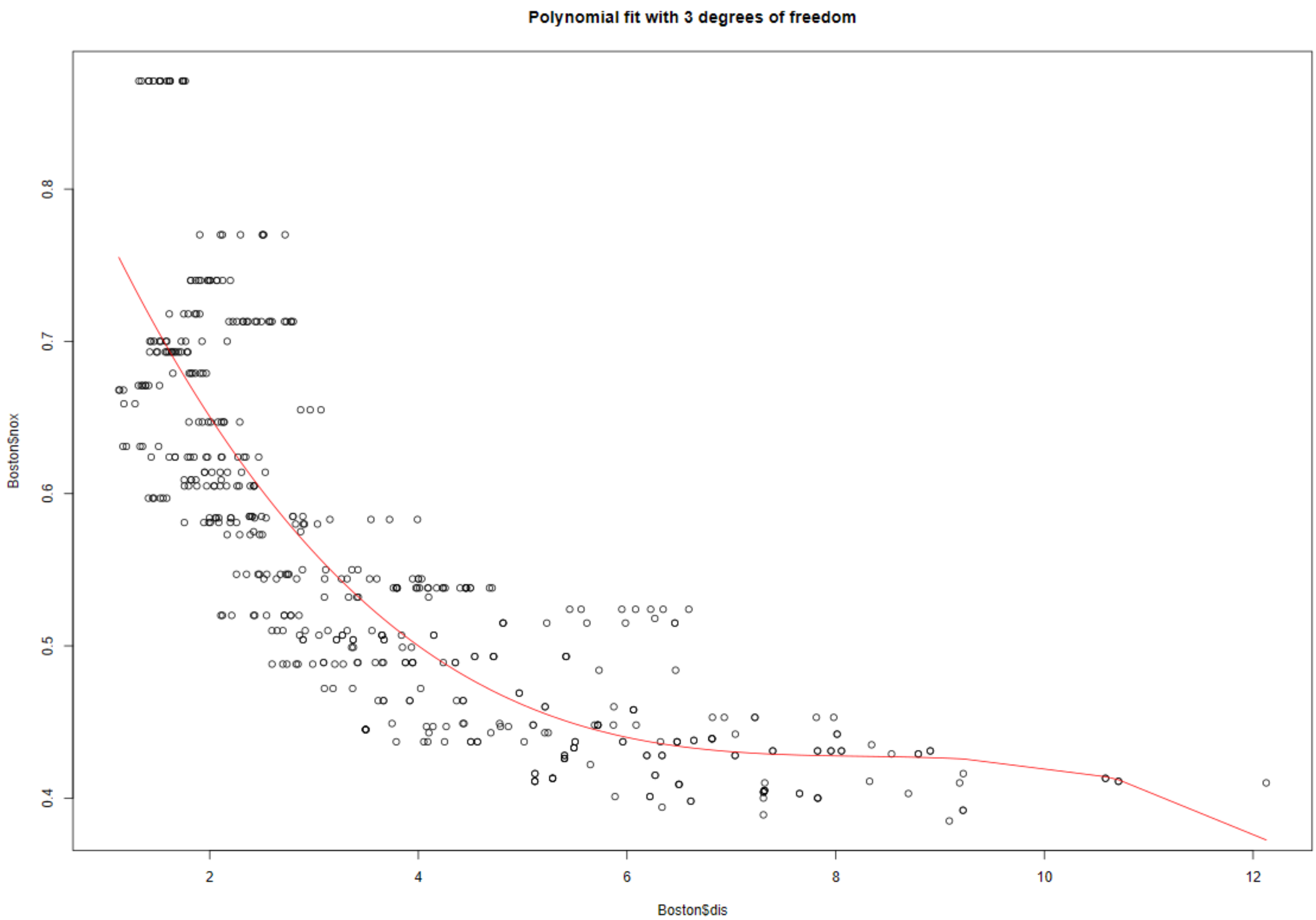
Polynomial fit with 3 degrees of freedom
Boston$nox
Boston$dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12

Figure 13

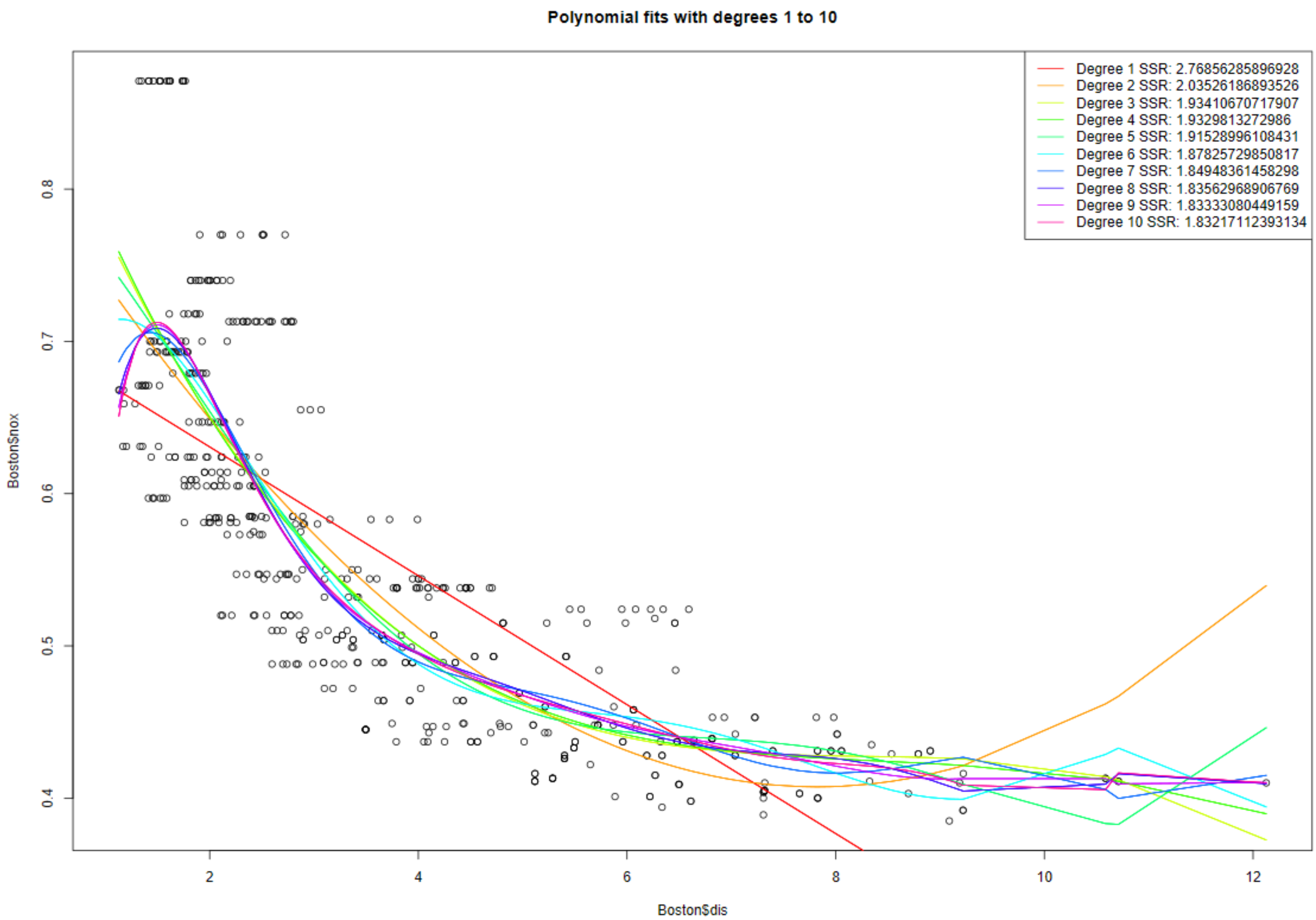
Polynomial fits with degrees 1 to 10
Degree 1 SSR: 2.76856285896928
Degree 2 SSR: 2.03526186893526
Degree 3 SSR: 1.93410670717907
Degree 4 SSR: 1.9329813272986
Degree 5 SSR: 1.91528996108431
Degree 6 SSR: 1.87825729850817
Degree 7 SSR: 1.84948361458298
Degree 8 SSR: 1.83562968906769
Degree 9 SSR: 1.83333080449159
Degree 10 SSR: 1.83217112393134
Boston$nox
Boston$dis
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12

Figure 14

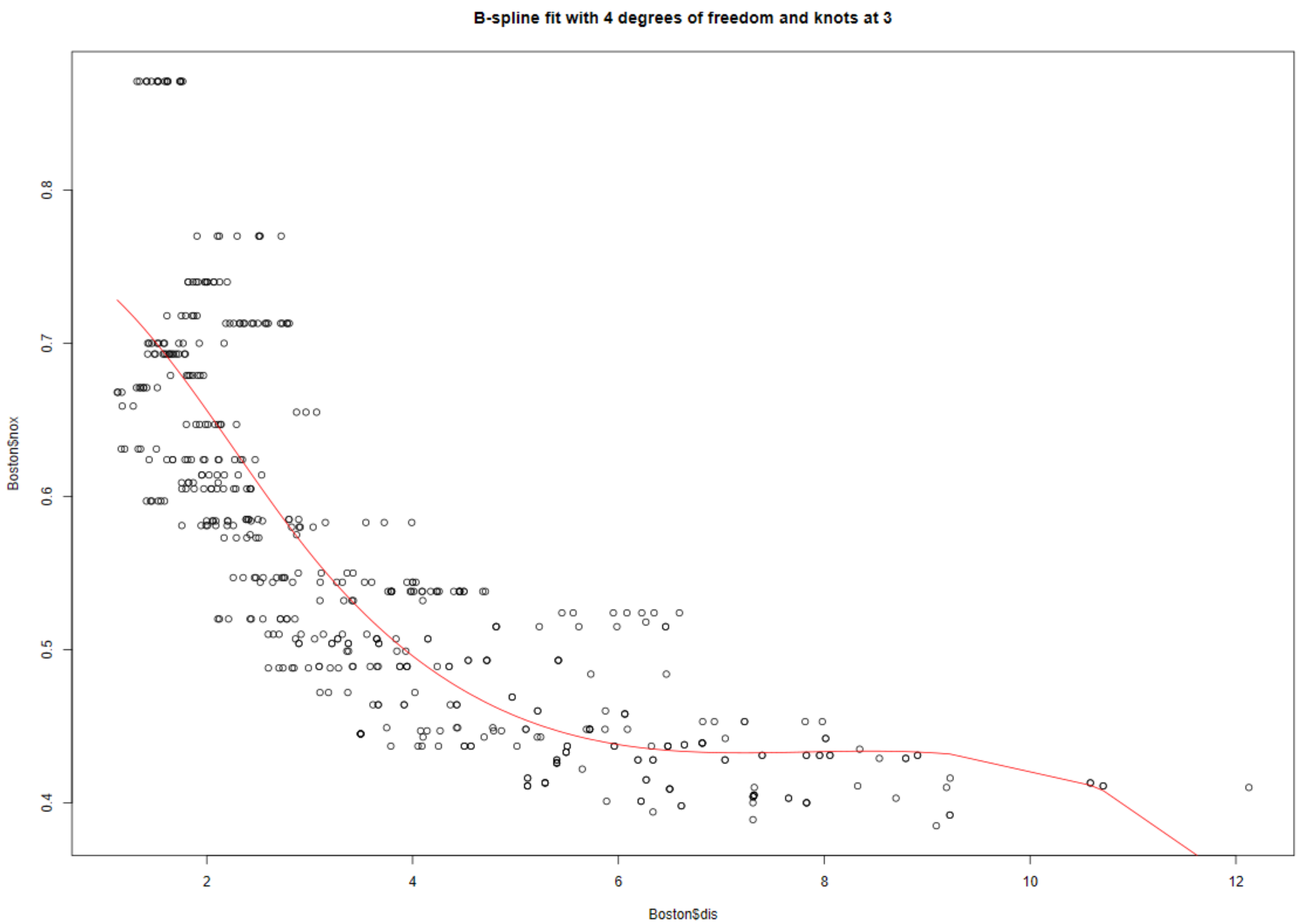
B-spline fit with 4 degrees of freedom and knots at 3
Boston$nox
0.4
0.5
0.6
0.7
0.8
Boston$dis
2
4
6
8
10
12

Figure 15

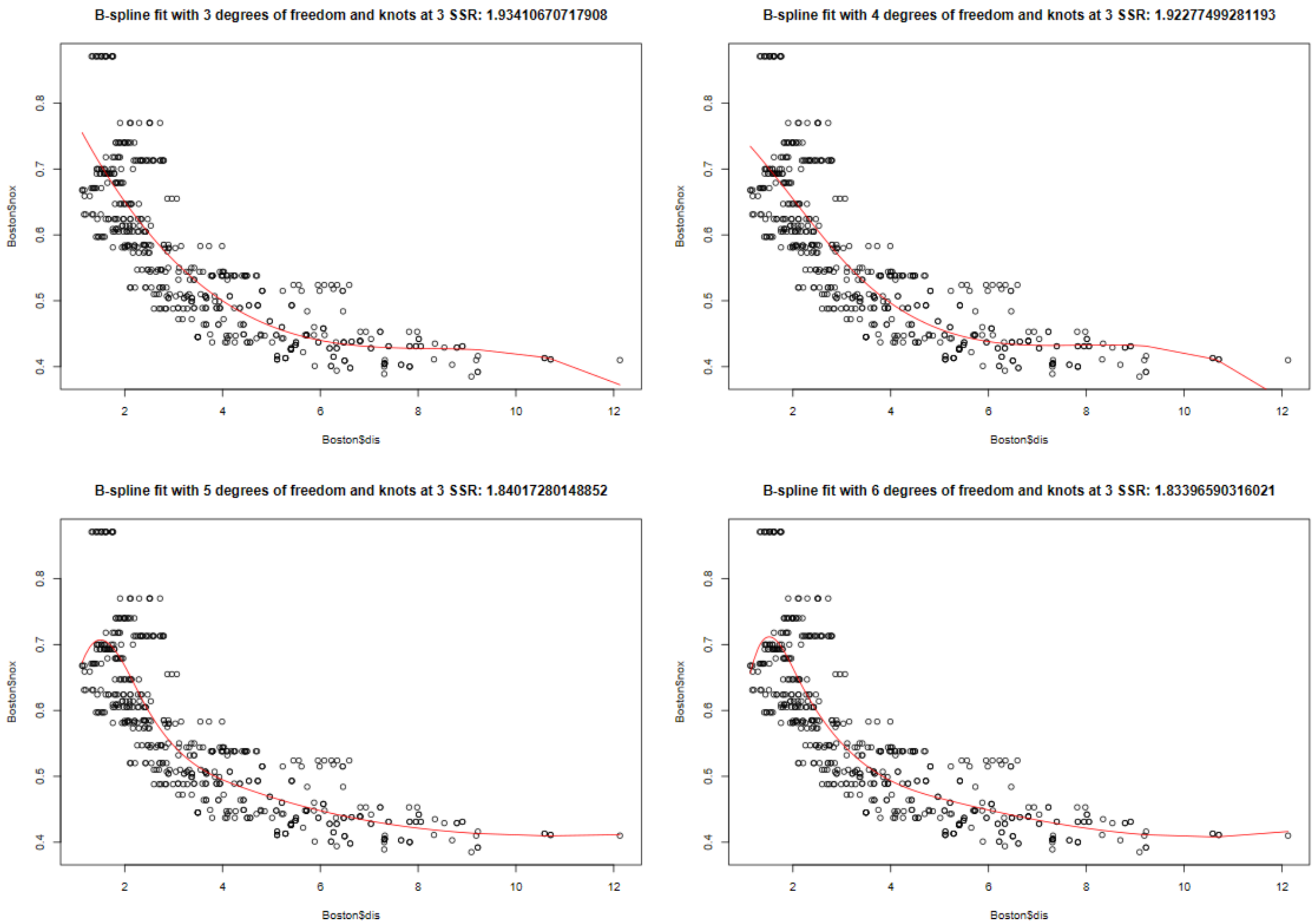
B-spline fit with 3 degrees of freedom and knots at 3 SSR: 1.93410670717908
B-spline fit with 4 degrees of freedom and knots at 3 SSR: 1.92277499281193
B-spline fit with 5 degrees of freedom and knots at 3 SSR: 1.84017280148852
B-spline fit with 6 degrees of freedom and knots at 3 SSR: 1.83396590316021
Boston$nox
Boston$dis

Figure 16