ISLR | Chapter 7 Exercises

Marshall McQuillen

9/24/2018

Conceptual

1

- **A**. The cubic piecewise polynomial:

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)^3_+ \quad where \quad (x-\xi)^3_+ = \begin{cases} 0, & x \leq \xi \\ (x-\xi)^3, & otherwise \end{cases}$$

  ...can be broken up and rewritten to be:

$$f(x) = \begin{cases} f_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3, & x \leq \xi \\ f_2(x) = a_2 + b_2 x + c_2 x^2 + d_2 x^3, & otherwise \end{cases}$$

  In $f_1(x)$, since $(x-\xi)^3_+ = 0$ (because $x \leq \xi$), the fifth term (of $f(x)$) zeroes out and the coefficients can be expresses as $a_1 = \beta_0,\ b_1 = \beta_1,\ c_1 = \beta_2$ and $d_1 = \beta_3$.

- **B**. Expanding the fifth term in $f(x)$ allows for the various powers of x to be grouped together and then recondensed. $a_2,\ b_2,\ c_2$ and d_2 are expressed in terms of the cofficients below.

$$\begin{aligned}
f_2(x) &= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)^3 && (1)\\
&= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)(x-\xi)(x-\xi) && (2)\\
&= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x^2 - 2x\xi + \xi^2)(x-\xi) && (3)\\
&= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x^3 - x^2\xi - 2x^2\xi + 2x\xi^2 + \xi^2 x - \xi^3) && (4)\\
&= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x^3 - 3x^2\xi + 3x\xi^2 - \xi^3) && (5)\\
&= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^3 - \beta_4 3x^2\xi + \beta_4 3x\xi^2 - \beta_4 \xi^3 && (6)\\
&= (\beta_0 - \beta_4 \xi^3) + (\beta_1 x + \beta_4 3x\xi^2) + (\beta_2 x^2 - \beta_4 3x^2\xi) + (\beta_3 x^3 + \beta_4 x^3) && (7)\\
&= (\beta_0 - \beta_4 \xi^3) + (\beta_1 + 3\beta_4 \xi^2)x + (\beta_2 - 3\beta_4 \xi)x^2 + (\beta_3 + \beta_4)x^3 && (8)
\end{aligned}$$

$$f_2(x) = a_2 + b_2 x + c_2 x^2 + d_2 x^3 \quad where \quad \begin{cases} a_2 = \beta_0 - \beta_4 \xi^3 \\ b_2 = \beta_1 + 3\beta_4 \xi^2 \\ c_2 = \beta_2 - 3\beta_4 \xi \\ d_2 = \beta_3 + \beta_4 \end{cases} \quad (9)$$

- **C**. Showing that $f(x)$ is continuous at ξ is illustrated by showing that $f(\xi)_1 = f(\xi)_2$.

$$
\begin{aligned}
f_1(\xi) &= a_1 + b_1(\xi) + c_1(\xi)^2 + d_1(\xi)^3 && (10)\\
&= \beta_0 + \beta_1(\xi) + \beta_2(\xi)^2 + \beta_3(\xi)^3 && (11)\\
& && (12)\\
f_2(\xi) &= a_2 + b_2(\xi) + c_2(\xi)^2 + d_2(\xi)^3 && (13)\\
&= (\beta_0 - \beta_4\xi^3) + (\beta_1 + 3\beta_4\xi^2)(\xi) + (\beta_2 - 3\beta_4\xi)(\xi)^2 + (\beta_3 + \beta_4)(\xi)^3 && (14)\\
&= (\beta_0 - \beta_4\xi^3) + (\beta_1\xi + 3\beta_4\xi^3) + (\beta_2\xi^2 - 3\beta_4\xi^3) + (\beta_3\xi^3 + \beta_4\xi^3) && (15)\\
&= \beta_0 - \beta_4\xi^3 + \beta_1\xi + 3\beta_4\xi^3 + \beta_2\xi^2 - 3\beta_4\xi^3 + \beta_3\xi^3 + \beta_4\xi^3 && (16)\\
&= \beta_0 + \beta_1\xi + \beta_2\xi^2 + \beta_3\xi^3 + 3\beta_4\xi^3 - 3\beta_4\xi^3 + \beta_4\xi^3 - \beta_4\xi^3 && (17)\\
&= \beta_0 + \beta_1\xi + \beta_2\xi^2 + \beta_3\xi^3 + (3\beta_4\xi^3 - 3\beta_4\xi^3) + (\beta_4\xi^3 - \beta_4\xi^3) && (18)\\
f_2(\xi) &= \beta_0 + \beta_1\xi + \beta_2\xi^2 + \beta_3\xi^3 && (19)
\end{aligned}
$$

$$f_2(\xi) = \beta_0 + \beta_1\xi + \beta_2\xi^2 + \beta_3\xi^3 = f_1(\xi)$$

- **D**. In order to show that $f_1'(\xi) = f_2'(\xi)$, we must first find $f'(x)$ with respect to x and then simplify both $f_1'(\xi)$ and $f_2'(\xi)$.

$$
\begin{aligned}
f(x) &= a_1 + b_1x + c_1x^2 + d_1x^3 && (20)\\
f'(x) &= b_1 + 2c_1x + 3d_1x^2 && (21)
\end{aligned}
$$

Therefore, substituting the necessary coefficients in for $b_1,\ c_1$ and d_1 in both $f_1'(\xi)$ and $f_2'(\xi)$, we get:

$$f'(x) = b_1 + 2c_1x + 3d_1x^2 \quad then \quad \begin{cases} f_1'(\xi) = \beta_1 + 2\beta_2\xi + 3\beta_3\xi^2 \\ f_2'(\xi) = (\beta_1 + 3\beta_4\xi^2) + 2(\beta_2 - 3\beta_4\xi)\xi + 3(\beta_3 + \beta_4)\xi^2 \end{cases} \tag{22}$$

$$
\begin{aligned}
f_2'(\xi) &= (\beta_1 + 3\beta_4\xi^2) + 2(\beta_2 - 3\beta_4\xi)\xi + 3(\beta_3 + \beta_4)\xi^2 && (23)\\
&= \beta_1 + 3\beta_4\xi^2 + 2\beta_2\xi - 6\beta_4\xi^2 + 3\beta_3\xi^2 + 3\beta_4\xi^2 && (24)\\
&= \beta_1 + 2\beta_2\xi + 3\beta_3\xi^2 + (3\beta_4\xi^2 + 3\beta_4\xi^2 - 6\beta_4\xi^2) && (25)\\
&= \beta_1 + 2\beta_2\xi + 3\beta_3\xi^2 + (6\beta_4\xi^2 - 6\beta_4\xi^2) && (26)\\
f_2'(\xi) &= \beta_1 + 2\beta_2\xi + 3\beta_3\xi^2 && (27)
\end{aligned}
$$

We now see that the derivative $f'(x)$ is continuous at knot ξ, which is to say $f_1'(\xi) = f_2'(\xi)$:

$$f_2'(\xi) = \beta_1 + 2\beta_2\xi + 3\beta_3\xi^2 = f_1'(\xi)$$

- **E**. In order to show that $f_1''(\xi) = f_2''(\xi)$, we must first find $f''(x)$ with respect to x and then simplify both $f_1''(\xi)$ and $f_2''(\xi)$.

$$f(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3 \tag{28}$$
$$f'(x) = b_1 + 2c_1 x + 3d_1 x^2 \tag{29}$$
$$f''(x) = 2c_1 + 6d_1 x \tag{30}$$

Therefore, substituting the necessary coefficients in for c_1 and d_1 in both $f_1''(\xi)$ and $f_2''(\xi)$, we come to:

$$f''(x) = 2c_1 + 6d_1 x \quad then \quad \begin{cases} f_1''(\xi) = 2\beta_2 + 6\beta_3\xi \\ f_2''(\xi) = 2(\beta_2 - 3\beta_4\xi) + 6(\beta_3 + \beta_4)\xi \end{cases} \tag{31}$$

$$f_2''(\xi) = 2(\beta_2 - 3\beta_4\xi) + 6(\beta_3 + \beta_4)\xi \tag{32}$$
$$= 2\beta_2 - 6\beta_4\xi + 6\beta_3\xi + 6\beta_4\xi \tag{33}$$
$$= 2\beta_2 + 6\beta_3\xi + (6\beta_4\xi - 6\beta_4\xi) \tag{34}$$
$$f_2''(\xi) = 2\beta_2 + 6\beta_3\xi \tag{35}$$

We now see that the second derivative $f''(x)$ is continuous at knot ξ, which is to say $f_1''(\xi) = f_2''(\xi)$:

$$f_2''(\xi) = 2\beta_2 + 6\beta_3\xi = f_1''(\xi)$$

2

(sketches on following page)

- **A**. With $\lambda = \infty$, the second term will dominate the above equation and the RSS will be ignored. Since $g^0 = g$, this comes out to finding $g(x)$ that minimizes the integral of $g(x)$. Therefore, $g(x) = 0$.
- **B**. With $\lambda = \infty$ and $m = 1$, the second term will dominate the above equation and the RSS will be ignored. This then becomes a problem of finding a function $g(x)$ where $\int g'(x)$ is minimized. Therefore, $g(x) = c$ (a flat line) where c is a constant, ensuring that $g'(x) = 0$.
- **C**. With $\lambda = \infty$ and $m = 2$, the second term will dominate the above equation and the RSS will be ignored. This then becomes a problem of finding a function $g(x)$ where $\int g''(x)$ is minimized.

  If we work backwards conceptually, we will see that $g(x) = \beta_0 + \beta_1 x$. Since $\int g''(x)$ must be minimized, $g''(x) = 0$. Therefore, $g'(x) = c$ where c is some constant. This implies that $g(x)$ must have a constant slope, c aka β_1. Therefore, $g(x) = \beta_0 + \beta_1 x$
- **D**. With $\lambda = \infty$ and $m = 3$, the second term will dominate the above equation and the RSS will be ignored. This then becomes a problem of finding a function $g(x)$ where $\int g'''(x)$ is minimized. Therefore, $g(x) = \beta_0 + \beta_1 x + \beta_2 x^2$, $g(x)$ will be quadratic in some sense

  Once again, working backwards conceptually, if the goal is to minimize $\int g'''(x)$, then $g'''(x) = 0$. Therefore, $g''(x) = c$, where c is some constant. This implies that $g'(x)$ must have a constant slope, c. if $g'(x)$ has a constant slope, then $g(x) = \beta_0 + \beta_1 x + \beta_2 x^2$. Having a quadratic equation means that the slope of $g(x)$ is changing at a fixed rate, which satisfies our condition that $g'(x) = c$.
- **E**. With $\lambda = 0$ and $m = 3$, the second term in the equation is completely ignored, and $g(x)$ becomes the line that interpolates all data points.

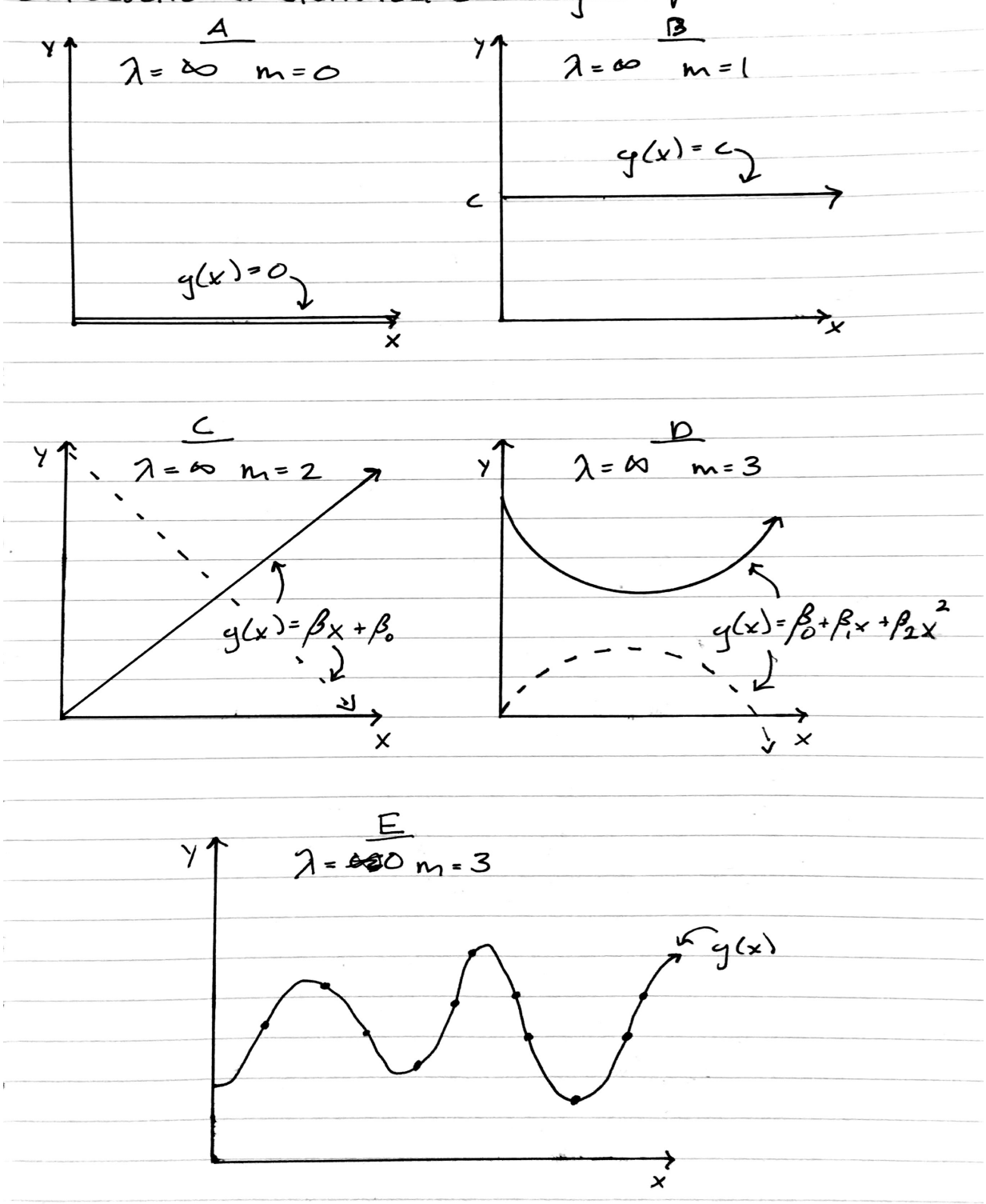
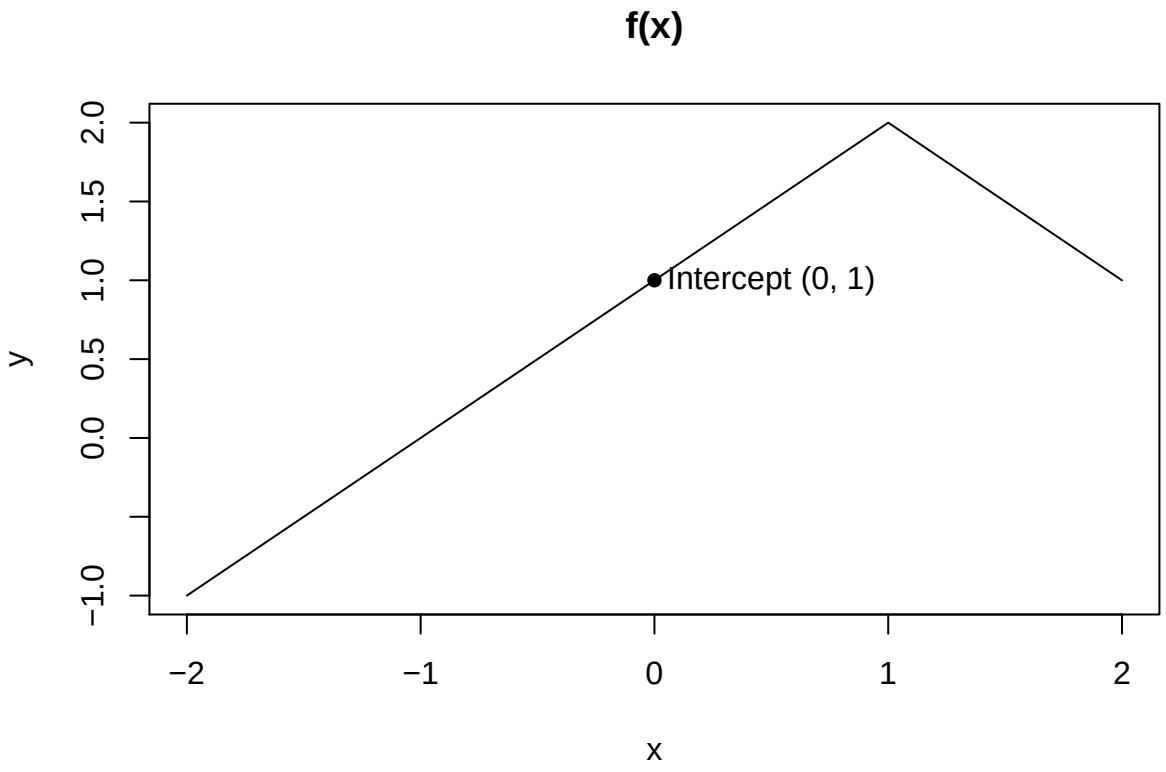


Figure 1: "Conceptual Exercise 2"

3

$$f(x) = 1 + x + \begin{cases} -2(x-1)^2, \ x \geq 1 \\ 0, \ otherwise \end{cases}$$

The intercept is at $y = 1$, $f(x)$ is linear with a slope equal to 1 up to $x = 1$, after which it becomes quadratic.

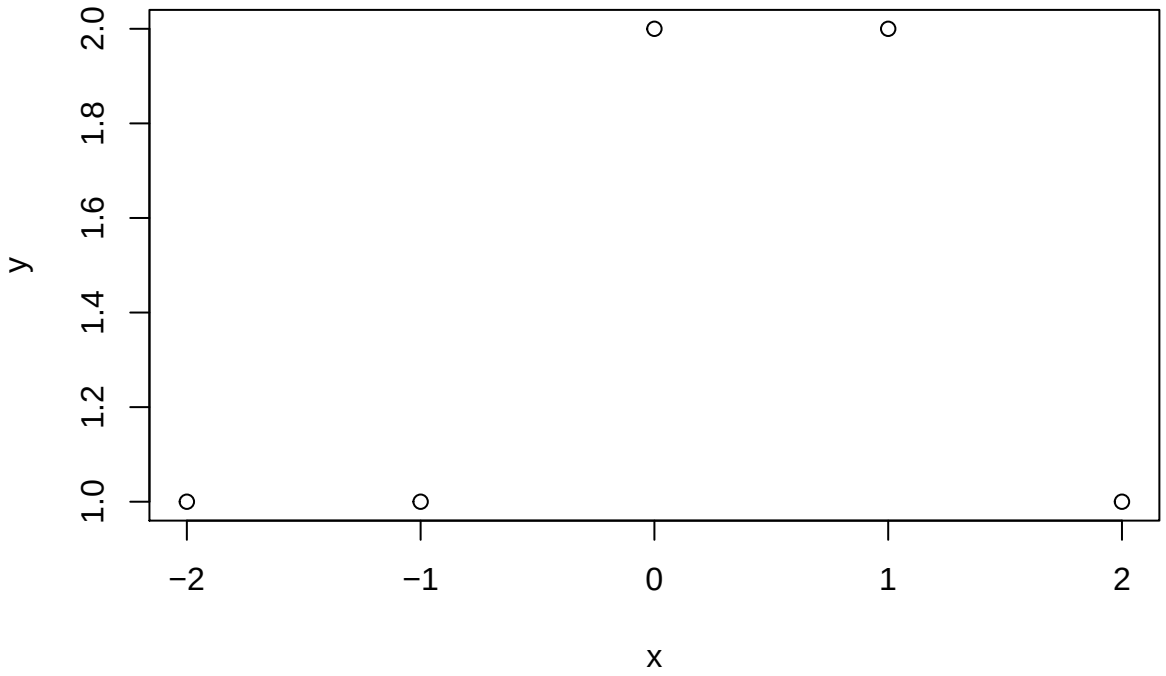


4

$$f(x) = \beta_0 + \beta_1 b_1(x) + \beta_2 b_2(x) \tag{36}$$

$$f(x) = 1 + b_1(x) + 3b_2(x) \quad where \quad \begin{cases} b_1(x) = I(0 \leq x \leq 2) - (x-1)I(1 \leq x \leq 2) \\ b_2(x) = (x-3)I(3 \leq x \leq 4) + I(4 < x \leq 5) \end{cases} \tag{37}$$

```
x <- -2:2
y <- c(1,1,2,2,1)
plot(x, y)
```



5

$$\hat{g_1} = \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \int [g^3(x)]^2 dx \right) \tag{38}$$

$$\hat{g_2} = \left(\sum_{i=1}^{n} (y_i - g(x_i))^2 + \int [g^4(x)]^2 dx \right) \tag{39}$$

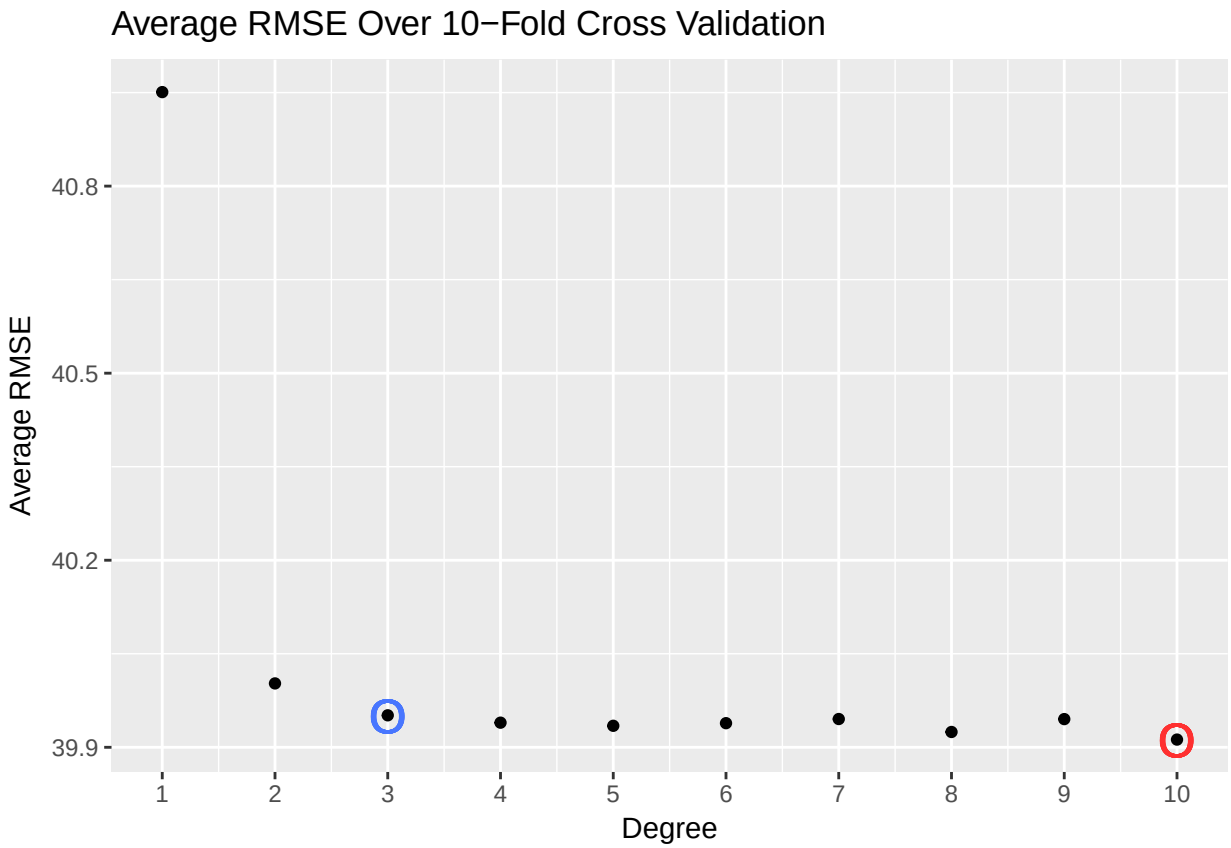
- **A**. As $\lambda \to \infty$, $\hat{g_2}$ will have a smaller training RSS. This is because $\hat{g_2}$ has one more degree of freedom than $\hat{g_1}$; in other words, it is allowed to be more flexible thatn $\hat{g_1}$.
- **B**. As $\lambda \to \infty$, $\hat{g_1}$ will most likely have a lower test RSS, although this is less certain than part **A**. It will most likely have a lower test RSS because we are constraining it more, which is to say there is less of a chance that it incorporates the error term ϵ into the model itself.
- **C**. If $\lambda = 0$, the two equations are the same so they will have the same training and test RSS (one that interpolates all data points).

Applied

6

- **A**. Using 10-Fold CV of wage predicted by age for polynomial fits ranging in degree from 1 to 10, the minimus MSE is at a degree of 10. However, the RMSE only improves marginally after a third degree polynomial. Therefore, since a more complex model is only justifiable when accompanied by a significant decrease in the error rate, I will move forward with the third degree polynomial (which coincides with the results obtained from ANOVA.

```
# imports
suppressPackageStartupMessages(library(ISLR))
suppressPackageStartupMessages(library(caret))
suppressPackageStartupMessages(library(boot))
suppressPackageStartupMessages(library(ggplot2))
attach(Wage)

set.seed(5)

# 10-Fold CV of Polynomial models with degree 1 - 10
degrees <- 1:10
cv.errors <- rep(0, 10)
for (i in degrees) {
    cv.fit <- glm(wage ~ poly(age, i), data = Wage)
    cv.errors[i] <- cv.glm(Wage, cv.fit, K = 10)$delta[1]
}

# Plot of CV errors
g <- ggplot(data.frame(x=1:10, y=sqrt(cv.errors)), aes(x, y)) +
    geom_point() +
    geom_point(aes(x=which.min(cv.errors),
                   y=sqrt(cv.errors[which.min(cv.errors)])),
               color = 'firebrick1',
               shape = "O",
               size = 6) +
     geom_point(aes(x=3,
               y=sqrt(cv.errors[3])),
           color = 'royalblue1',
           shape = "O",
           size = 6) +
    scale_x_continuous(breaks = 1:10,
                       labels = as.character(c(1:10))) +
    ggtitle("Average RMSE Over 10-Fold Cross Validation") +
    xlab("Degree") +
    ylab("Average RMSE")
g
```

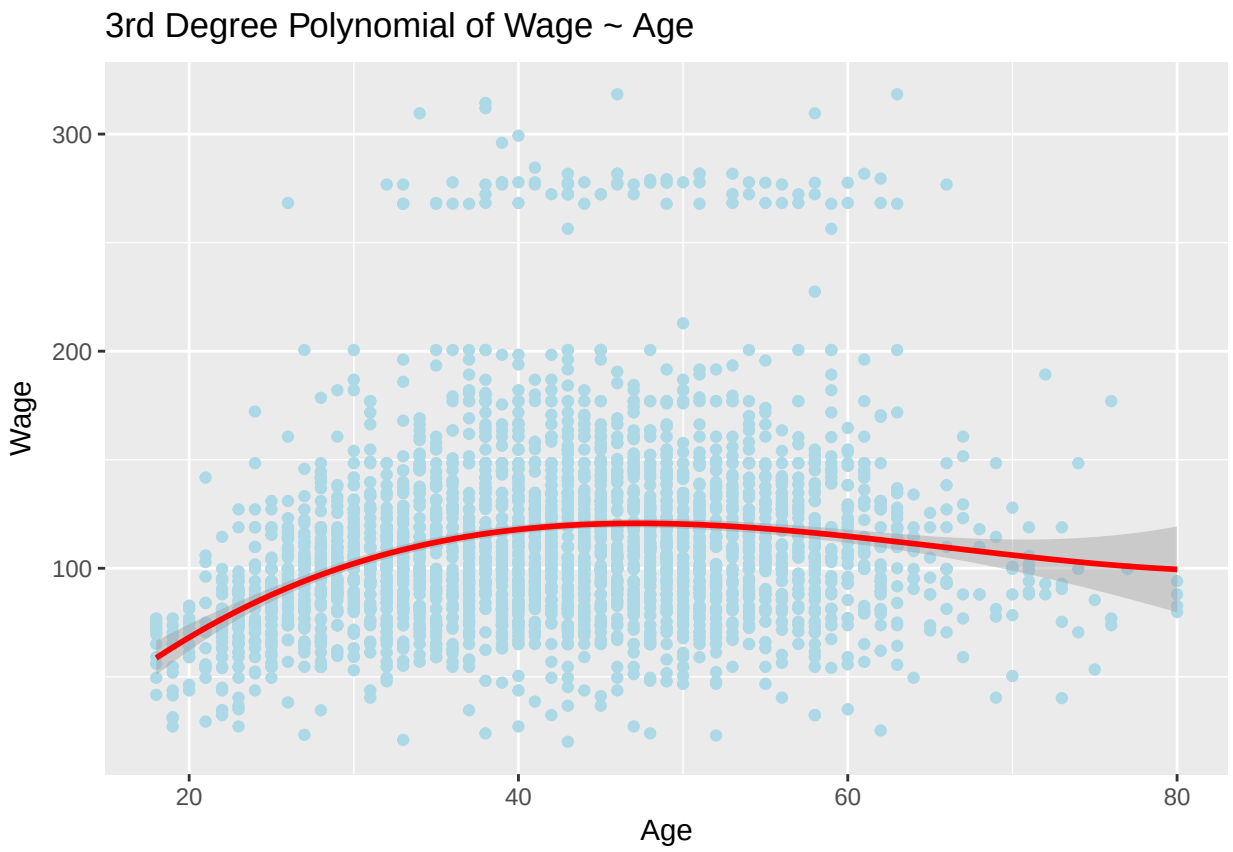



```
# ANOVA
fit.1 <- lm(wage ~ age, data = Wage)
fit.2 <- lm(wage ~ poly(age, 2), data = Wage)
fit.3 <- lm(wage ~ poly(age, 3), data = Wage)
fit.4 <- lm(wage ~ poly(age, 4), data = Wage)
fit.5 <- lm(wage ~ poly(age, 5), data = Wage)
fit.6 <- lm(wage ~ poly(age, 6), data = Wage)
fit.7 <- lm(wage ~ poly(age, 7), data = Wage)
fit.8 <- lm(wage ~ poly(age, 8), data = Wage)
fit.9 <- lm(wage ~ poly(age, 9), data = Wage)
fit.10 <- lm(wage ~ poly(age, 10), data = Wage)
anova(fit.1, fit.2, fit.3, fit.4, fit.5, fit.6, fit.7, fit.8, fit.9, fit.10)
```

```
## Analysis of Variance Table
##
## Model  1: wage ~ age
## Model  2: wage ~ poly(age, 2)
## Model  3: wage ~ poly(age, 3)
## Model  4: wage ~ poly(age, 4)
## Model  5: wage ~ poly(age, 5)
## Model  6: wage ~ poly(age, 6)
## Model  7: wage ~ poly(age, 7)
## Model  8: wage ~ poly(age, 8)
## Model  9: wage ~ poly(age, 9)
## Model 10: wage ~ poly(age, 10)
##    Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1    2998 5022216
```

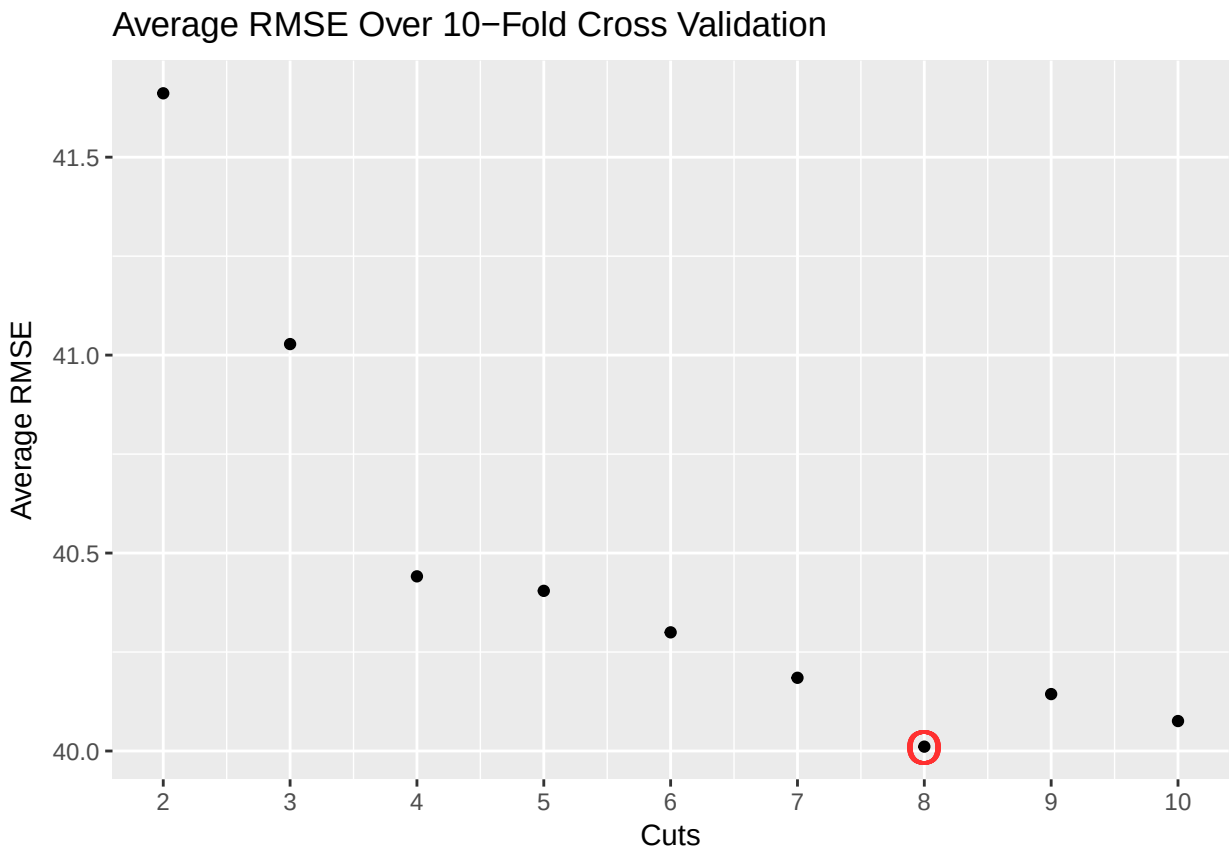
```
## 2    2997 4793430  1    228786 143.7638 < 2.2e-16 ***
## 3    2996 4777674  1     15756   9.9005  0.001669 **
## 4    2995 4771604  1      6070   3.8143  0.050909 .
## 5    2994 4770322  1      1283   0.8059  0.369398
## 6    2993 4766389  1      3932   2.4709  0.116074
## 7    2992 4763834  1      2555   1.6057  0.205199
## 8    2991 4763707  1       127   0.0796  0.777865
## 9    2990 4756703  1      7004   4.4014  0.035994 *
## 10   2989 4756701  1         3   0.0017  0.967529
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Plot 3rd degree polynomial
g <- ggplot(Wage,
            aes(x = age, y = wage)) +
     geom_point(color = 'lightblue') +
     stat_smooth(method = 'lm',
                 formula = y ~ poly(x, 3),
                 size = 1,
                 color = 'red') +
     ggtitle("3rd Degree Polynomial of Wage ~ Age") +
     xlab("Age") +
     ylab("Wage")
g
```

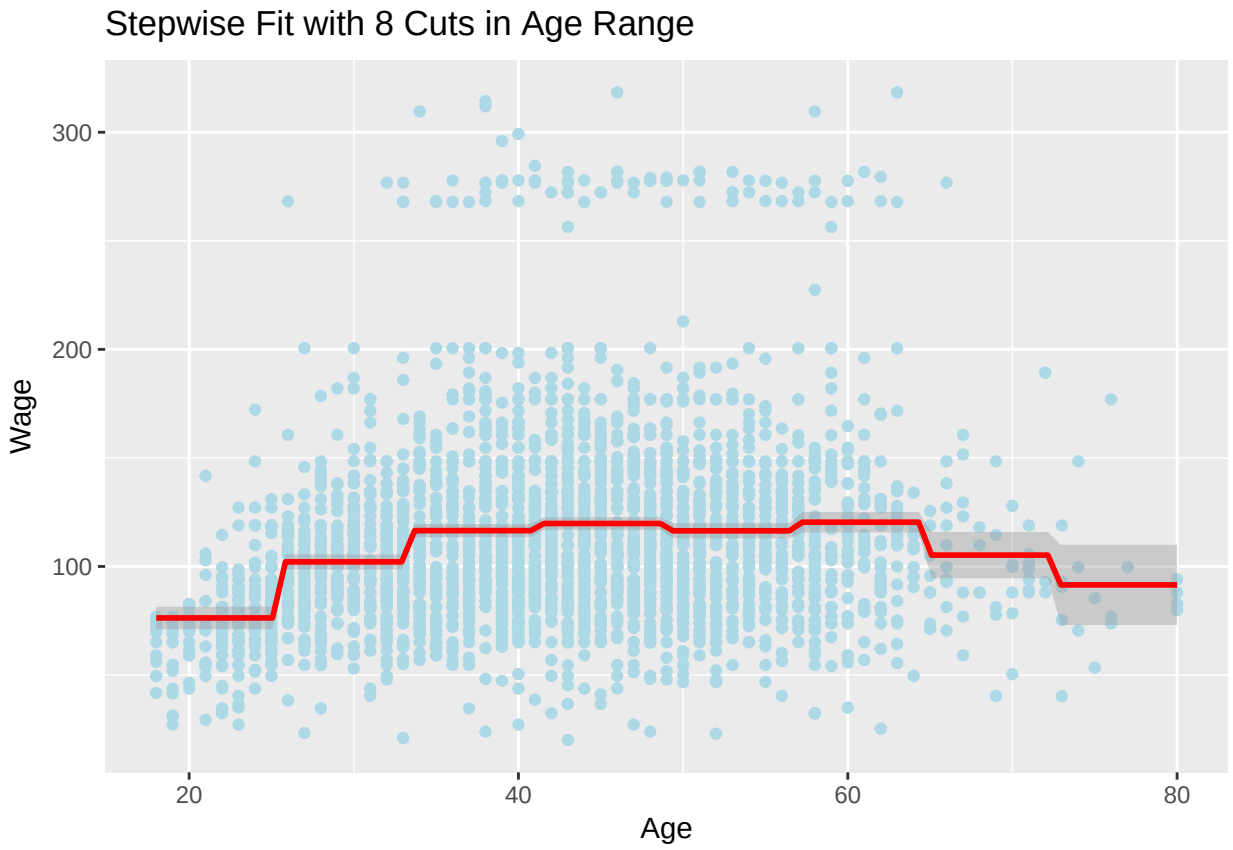


- **B**. Since the model will start to overfit as the number of cuts increases, I will limit the number of cuts to be a maximum of 10. As shown below, the minimum error is produced with 8 cuts.

```r
# 10-Fold CV of step functions up to 10 cuts
set.seed(5)

cuts <- 2:10
cv.errors <- rep(0, 9)
for (i in cuts) {
    Wage$age.cut <- cut(age, i)
    cv.fit <- glm(wage ~ age.cut, data = Wage)
    cv.errors[i-1] <- cv.glm(Wage, cv.fit, K = 10)$delta[1]
}

# Plot of CV error
g <- ggplot(data.frame(x=cuts, y=sqrt(cv.errors)), aes(x, y)) +
    geom_point() +
    geom_point(aes(x=which.min(cv.errors) + 1,
                   y=sqrt(cv.errors[which.min(cv.errors)])),
               color = 'firebrick1',
               shape = "O",
               size = 6) +
    scale_x_continuous(breaks = 1:10,
                       labels = as.character(c(1:10))) +
    ggtitle("Average RMSE Over 10-Fold Cross Validation") +
    xlab("Cuts") +
    ylab("Average RMSE")
g
```



```r
# Plot step function
g <- ggplot(Wage,
            aes(x = age, y = wage)) +
     geom_point(color = 'lightblue') +
     stat_smooth(method = 'lm',
                 formula = y ~ cut(x, 8),
                 size = 1,
                 color = 'red') +
     ggtitle("Stepwise Fit with 8 Cuts in Age Range") +
     xlab("Age") +
     ylab("Wage")
g
```

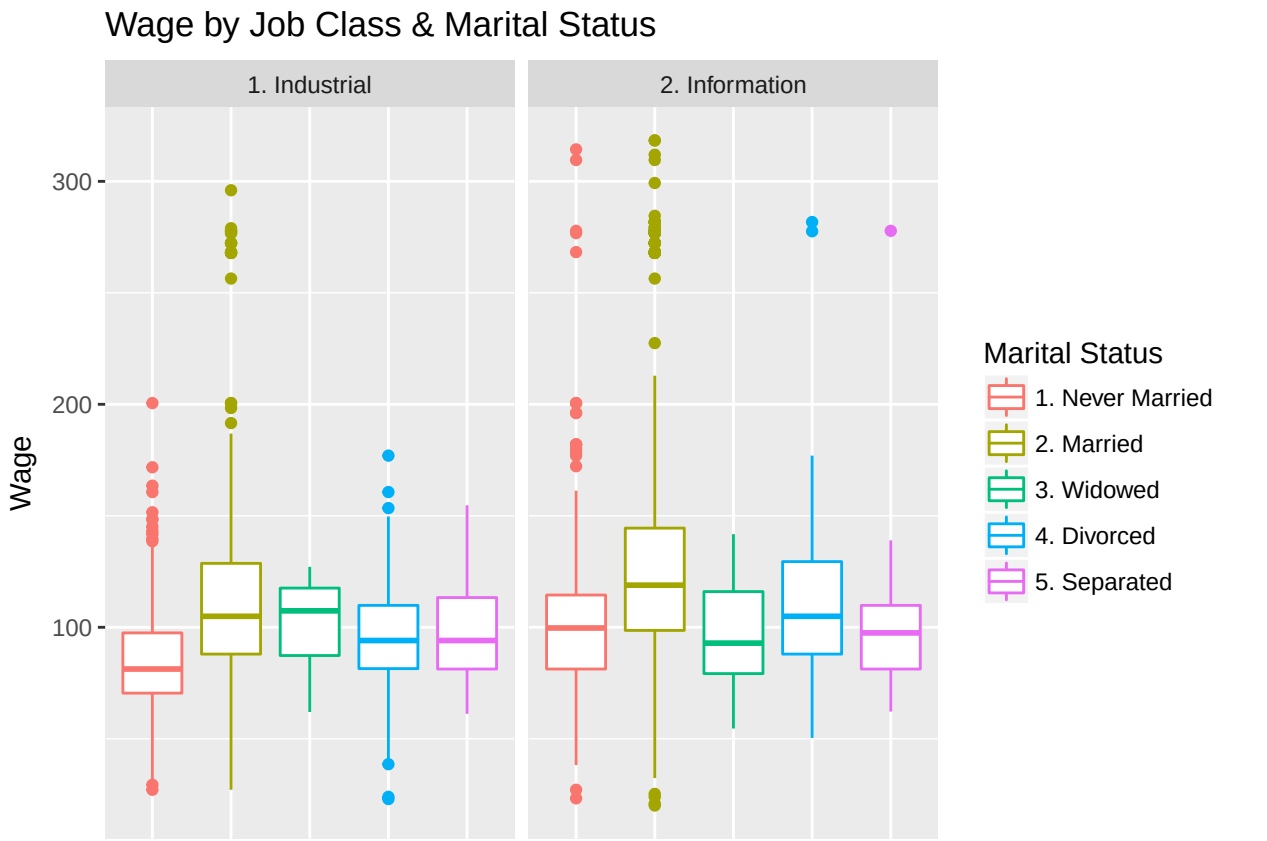


7

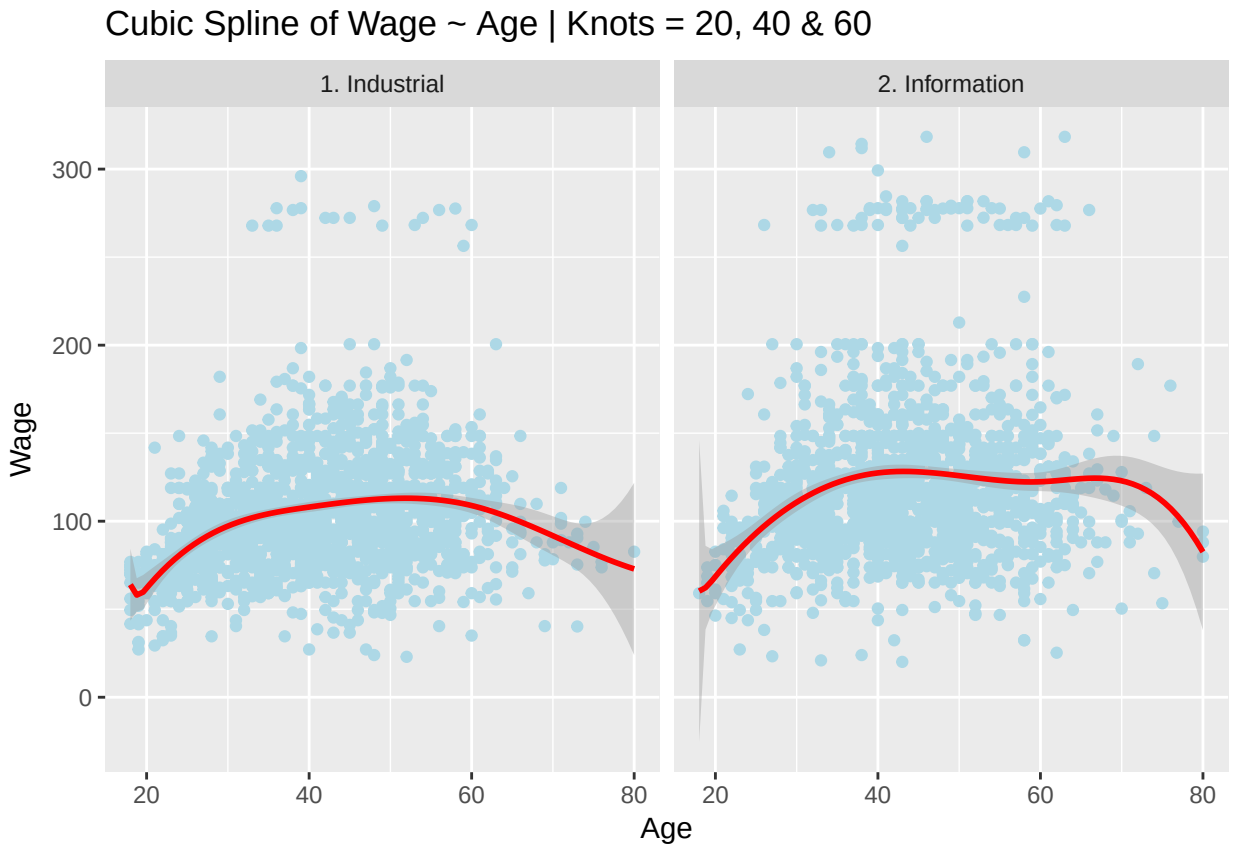
Since polynomial, stepwise functions or splines can't be fit with qualitative data, below is a plot showing the division of wage across job class (industrial and informational) and marital status. Interestingly, there is not a large difference between the average income for workers in the industrial sector and those in the informational sector. In fact, given that a worker is widowed, one would expect him/her to earn more in the industrial sector based on this data.

In addition, looking at cubic splines of age segmented by job class and marital status, it is clear that income in the information sector is higher, although it drops off more sharply in one's later years. One can also expect to earn more if married, as shown in the third set of plots (only marital status' with a significant number of observations were included).

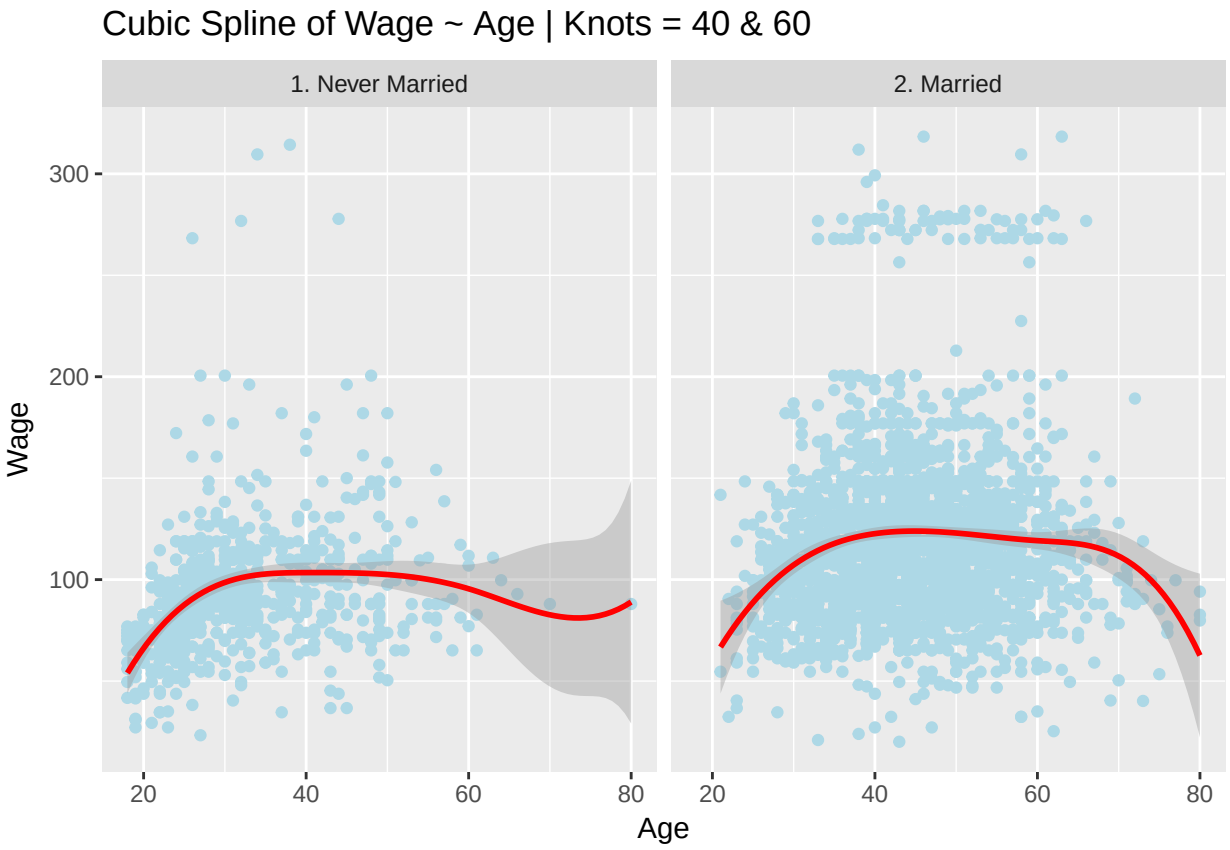
```
suppressPackageStartupMessages(library(splines))
set.seed(5)

# Plot of Wage across job class and marital status
g <- ggplot(Wage, aes(maritl, wage)) +
    geom_boxplot(aes(colour=maritl)) +
    facet_wrap(~jobclass) +
    theme(axis.ticks.x = element_blank(),
          axis.text.x = element_blank(),
          axis.title.x = element_blank()) +
    guides(color=guide_legend(title = "Marital Status")) +
    ggtitle("Wage by Job Class & Marital Status") +
    ylab("Wage")
g
```



```r
# Plot of Wage predicted by age partitioned by job class
g <- ggplot(Wage,
            aes(x = age, y = wage)) +
    geom_point(color = 'lightblue') +
    facet_wrap(~jobclass) +
    stat_smooth(method = 'lm',
                formula = y ~ bs(x, knots = c(20,40,60)),
                size = 1,
                color = 'red') +
    ggtitle("Cubic Spline of Wage ~ Age | Knots = 20, 40 & 60") +
    xlab("Age") +
    ylab("Wage")
g
```




```r
# Plot of Wage predicted by age partitioned by marital status
g <- ggplot(subset(Wage, maritl %in% c("1. Never Married", "2. Married")),
            aes(x = age, y = wage)) +
    geom_point(color = 'lightblue') +
    facet_wrap(~maritl) +
    stat_smooth(method = 'lm',
                formula = y ~ bs(x, knots = c(40,60)),
                size = 1,
                color = 'red') +
    ggtitle("Cubic Spline of Wage ~ Age | Knots = 40 & 60") +
    xlab("Age") +
    ylab("Wage")
g
```



8

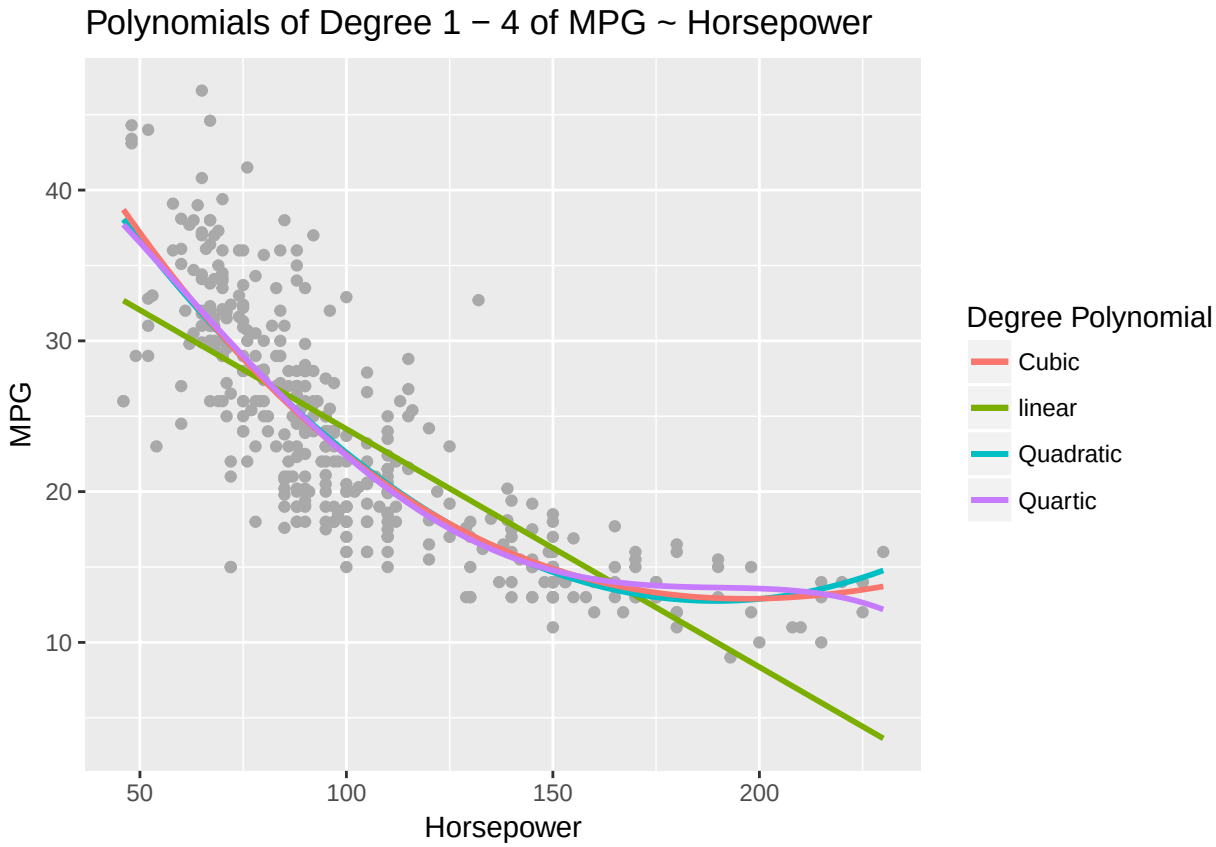
Plotting a linear model of miles per gallon predicted by horsepower, along with 3 polynomial models with degree 2 - 4, it seems clear that there is evidence for a quadratic model, althought the improvements after that become marginal. This is confirmed with ANOVA of all 4 models. The quadratic model returns a rather impressive RMSE of 4.38.

```
detach(Wage)
attach(Auto)
```

```
## The following object is masked from package:ggplot2:
##
##     mpg
```

```
set.seed(5)

# Plot of polynomials of MPG as predicted by horsepower
g <- ggplot(Auto,
            aes(x = horsepower, y = mpg)) +
    geom_point(color = 'darkgrey') +
    stat_smooth(method = 'lm',
                formula = y ~ x,
                se = F,
                size = 1,
                aes(color = 'linear')) +
    stat_smooth(method = 'lm',
                formula = y ~ poly(x, 2),
```

```
                se = F,
                size = 1,
                aes(color = 'Quadratic')) +
    stat_smooth(method = 'lm',
                formula = y ~ poly(x, 3),
                se = F,
                size = 1,
                aes(color = 'Cubic')) +
    stat_smooth(method = 'lm',
                formula = y ~ poly(x, 4),
                se = F,
                size = 1,
                aes(color = 'Quartic')) +
    guides(color=guide_legend(title = "Degree Polynomial")) +
    ggtitle("Polynomials of Degree 1 - 4 of MPG ~ Horsepower") +
    xlab("Horsepower") +
    ylab("MPG")
g
```




```
# ANOVA of polynomial degree models
fit.1 <- lm(mpg ~ horsepower, data = Auto)
fit.2 <- lm(mpg ~ poly(horsepower, 2), data = Auto)
fit.3 <- lm(mpg ~ poly(horsepower, 3), data = Auto)
fit.4 <- lm(mpg ~ poly(horsepower, 4), data = Auto)
anova(fit.1, fit.2, fit.3, fit.4)
```

```
## Analysis of Variance Table
```

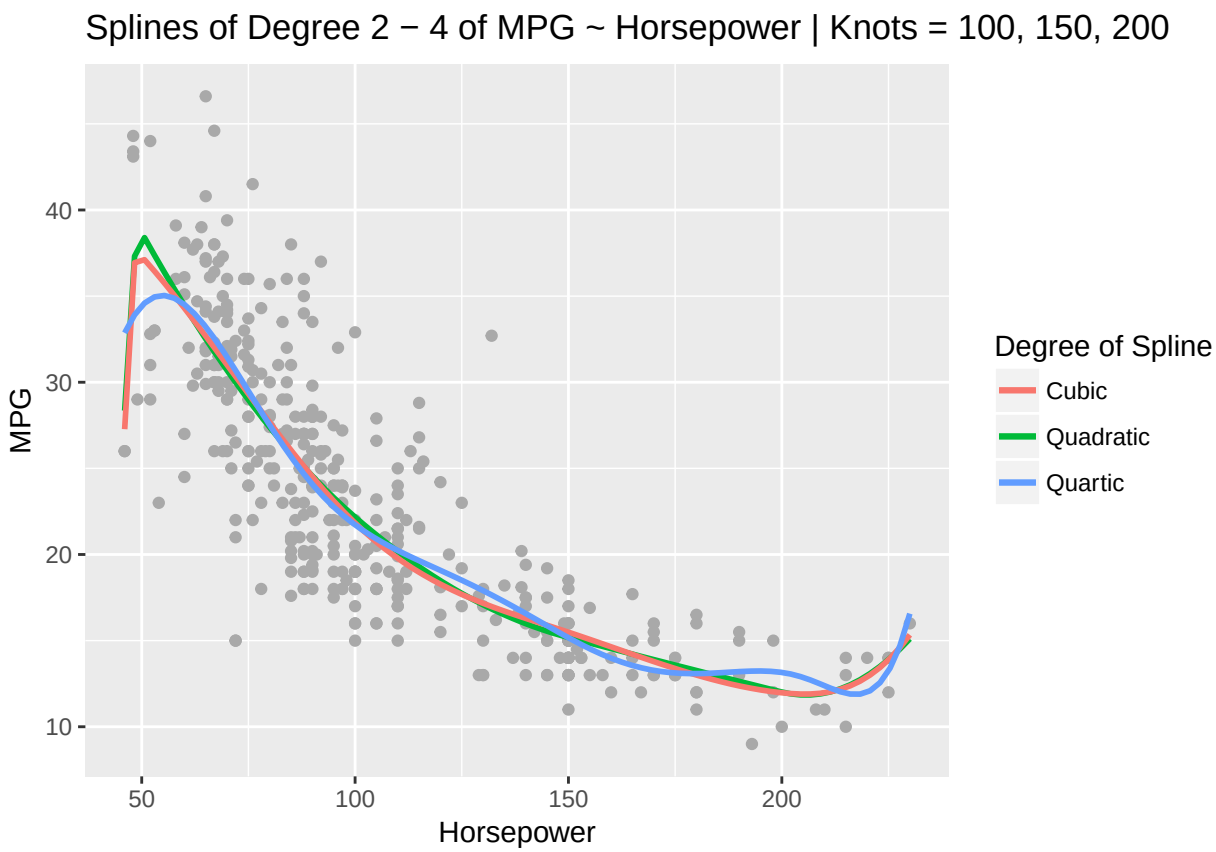
```
##
## Model 1: mpg ~ horsepower
## Model 2: mpg ~ poly(horsepower, 2)
## Model 3: mpg ~ poly(horsepower, 3)
## Model 4: mpg ~ poly(horsepower, 4)
##   Res.Df    RSS Df Sum of Sq        F Pr(>F)
## 1    390 9385.9
## 2    389 7442.0  1   1943.89 101.6666 <2e-16 ***
## 3    388 7426.4  1     15.59   0.8155 0.3670
## 4    387 7399.5  1     26.91   1.4076 0.2362
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# CV error of final polynomial model (degree = 2)
cv.fit <- glm(mpg ~ poly(horsepower, 2), data = Auto)
cv.error <- cv.glm(Auto, cv.fit, K = 10)$delta[1]
sqrt(cv.error)
```

```
## [1] 4.383345
```

Moving to splines, while there is statistically significant evidence that a quartic spline with knots at 100, 150 and 200 produces a better fit than a quadratic polynomial, the difference in the RMSE is marginal, therefore if a model had to be chosen for production, I would choose the quadratic polynomial.

```
# Plot of splines of MPG as predicted by horsepower with knots at 100, 150, 200
g <- ggplot(Auto,
            aes(x = horsepower, y = mpg)) +
    geom_point(color = 'darkgrey') +
    stat_smooth(method = 'lm',
                formula = y ~ bs(x,
                                 degree = 2,
                                 knots = seq(50, 200, 50)),
                se = F,
                size = 1,
                aes(color = 'Quadratic')) +
    stat_smooth(method = 'lm',
                formula = y ~ bs(x,
                                 degree = 3,
                                 knots = seq(50, 200, 50)),
                se = F,
                size = 1,
                aes(color = 'Cubic')) +
    stat_smooth(method = 'lm',
                formula = y ~ bs(x,
                                 degree = 4,
                                 knots = seq(100, 200, 50)),
                se = F,
                size = 1,
                aes(color = 'Quartic')) +
    guides(color=guide_legend(title = "Degree of Spline")) +
    ggtitle("Splines of Degree 2 - 4 of MPG ~ Horsepower | Knots = 100, 150, 200") +
    xlab("Horsepower") +
    ylab("MPG")
g
```




```
# ANOVA of splines models with knots at 100, 150, 200
fit.1 <- lm(mpg ~ horsepower, data = Auto)
fit.2 <- lm(mpg ~ bs(horsepower,
                     degree = 2,
                     knots = seq(100, 200, 50)),
            data = Auto)
fit.3 <- lm(mpg ~ bs(horsepower,
                     degree = 3,
                     knots = seq(100, 200, 50)),
            data = Auto)
fit.4 <- lm(mpg ~ bs(horsepower,
                     degree = 4,
                     knots = seq(100, 200, 50)),
            data = Auto)
anova(fit.1, fit.2, fit.3, fit.4)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ horsepower
## Model 2: mpg ~ bs(horsepower, degree = 2, knots = seq(100, 200, 50))
## Model 3: mpg ~ bs(horsepower, degree = 3, knots = seq(100, 200, 50))
## Model 4: mpg ~ bs(horsepower, degree = 4, knots = seq(100, 200, 50))
##   Res.Df    RSS Df Sum of Sq       F    Pr(>F)
## 1    390 9385.9
## 2    386 7390.5  4   1995.45 27.0332 < 2.2e-16 ***
## 3    385 7220.9  1    169.61  9.1912  0.002597 **
## 4    384 7086.2  1    134.64  7.2961  0.007217 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# CV error of final spline model (degree = 4, knots @ 100, 150, 200)

cv.fit <- glm(mpg ~ bs(horsepower,
                       degree = 4,
                       knots = seq(100, 200, 50)),
              data = Auto)
suppressWarnings(
    cv.error <- cv.glm(Auto, cv.fit, K = 10)$delta[1]
)
sqrt(cv.error)
```

```
## [1] 4.376662
```

```
# Final model test
fit.poly <- lm(mpg ~ poly(horsepower, 2), data = Auto)
fit.spline <- lm(mpg ~ bs(horsepower,
                          degree = 4,
                          knots = seq(100, 200, 50)),
                 data = Auto)
anova(fit.poly, fit.spline)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ poly(horsepower, 2)
## Model 2: mpg ~ bs(horsepower, degree = 4, knots = seq(100, 200, 50))
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1    389 7442.0
## 2    384 7086.2  5    355.82 3.8563 0.002018 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

9

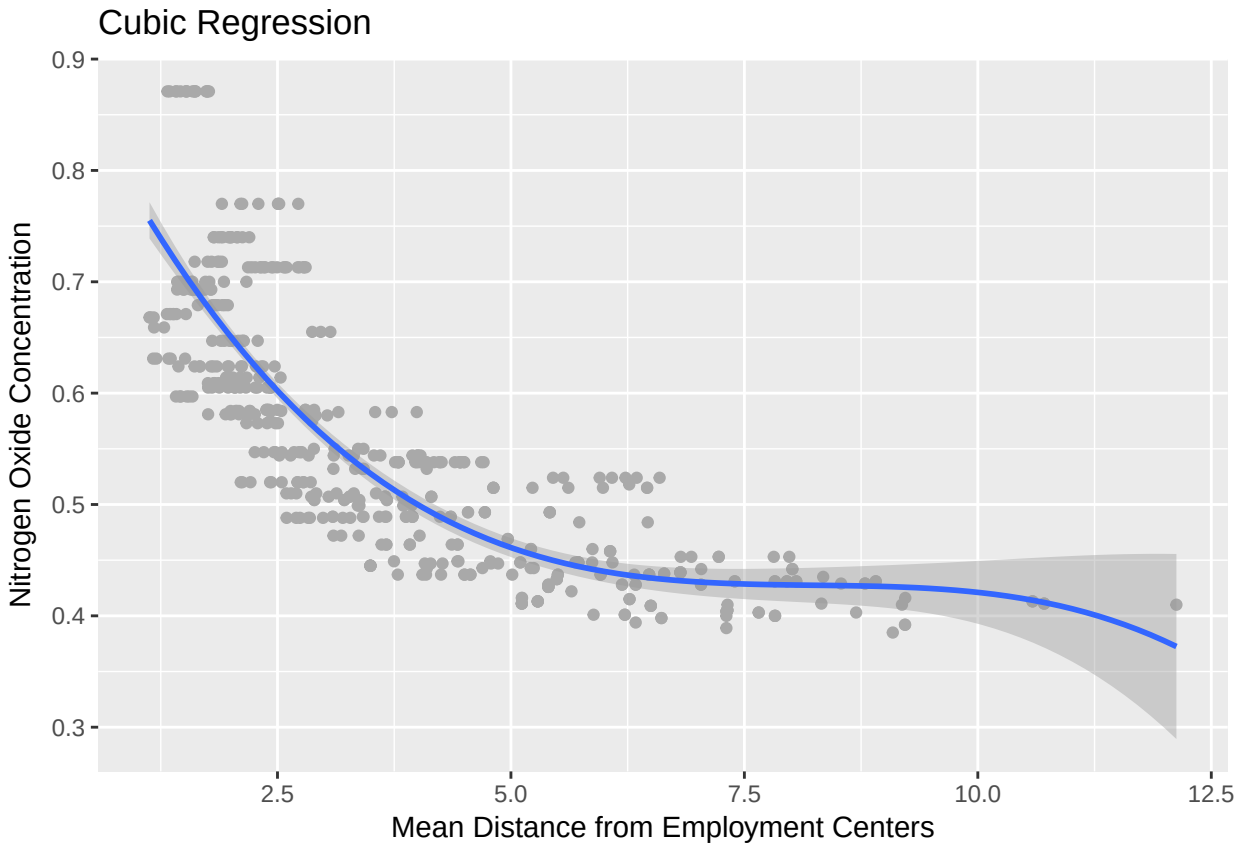
- **A**.

```
suppressPackageStartupMessages(library(MASS))
detach(Auto)
attach(Boston)

# Fit a cubic polynomial
fit.cubic <- lm(nox ~ poly(dis, 3), data = Boston)
summary(fit.cubic)
```

```
##
## 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:
```

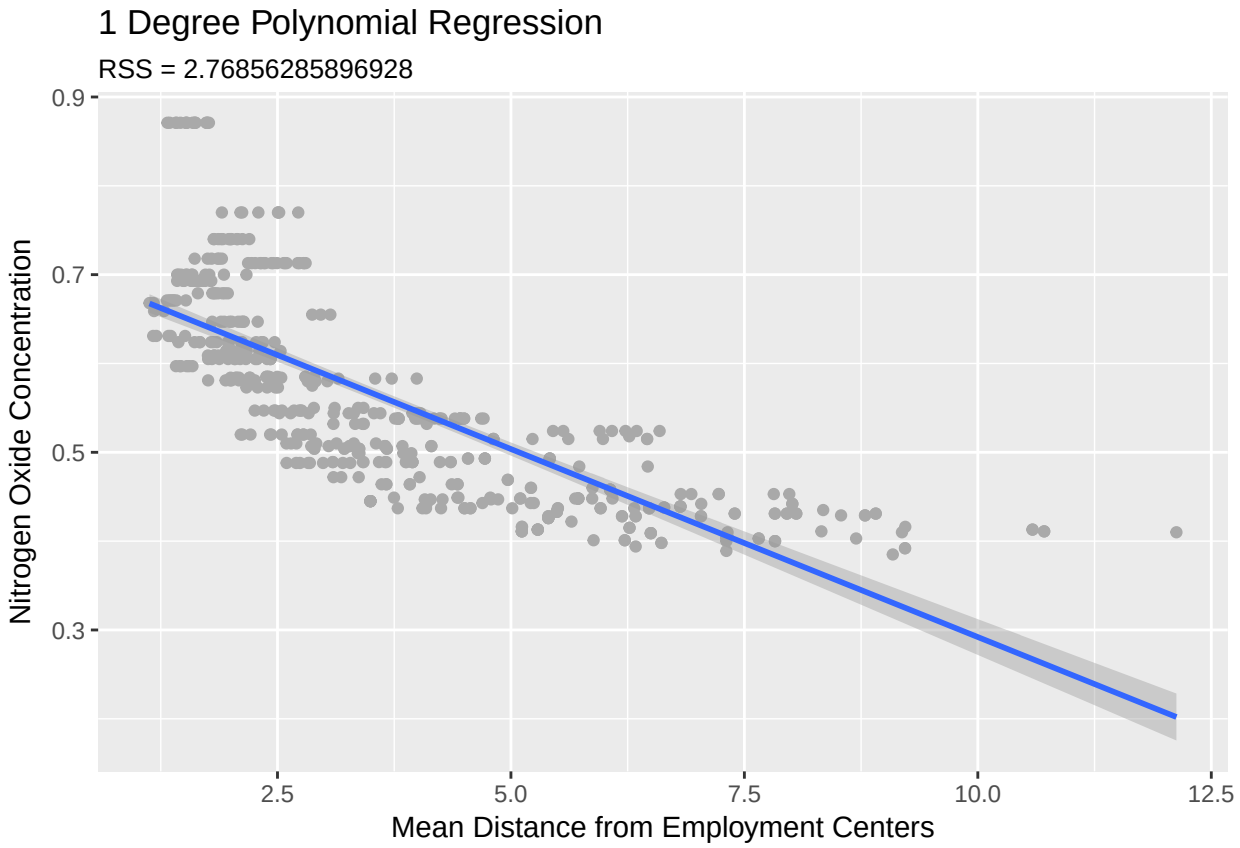
```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
## poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
## poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
## poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
## ---
## 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
```

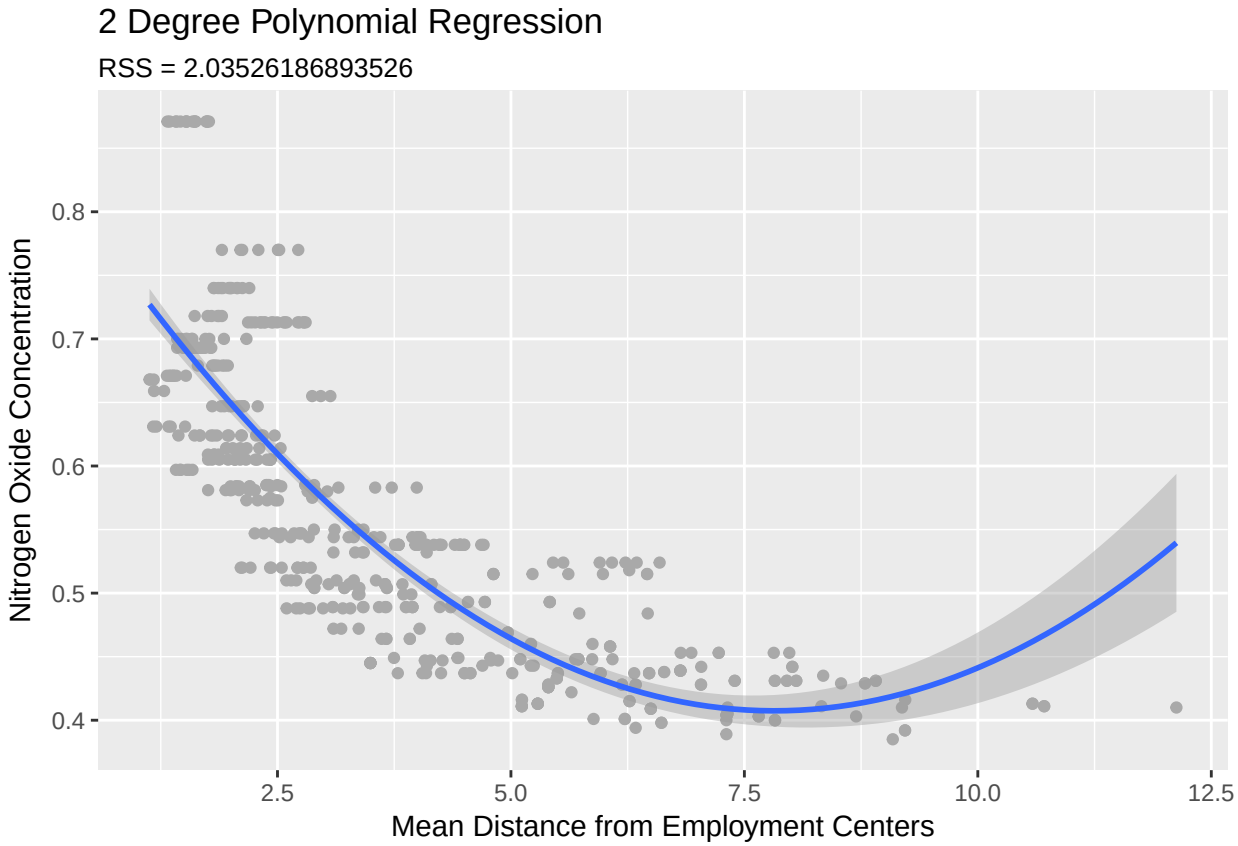
```r
# Plot a cubic polynomial
g <- ggplot(Boston, aes(x = dis, y = nox)) +
    geom_point(color = 'darkgrey') +
    stat_smooth(method = 'lm',
                formula = y ~ poly(x, 3)) +
    guides(color=guide_legend(title = "Degree of Polynomial")) +
    ggtitle("Cubic Regression") +
    xlab("Mean Distance from Employment Centers") +
    ylab("Nitrogen Oxide Concentration")
g
```

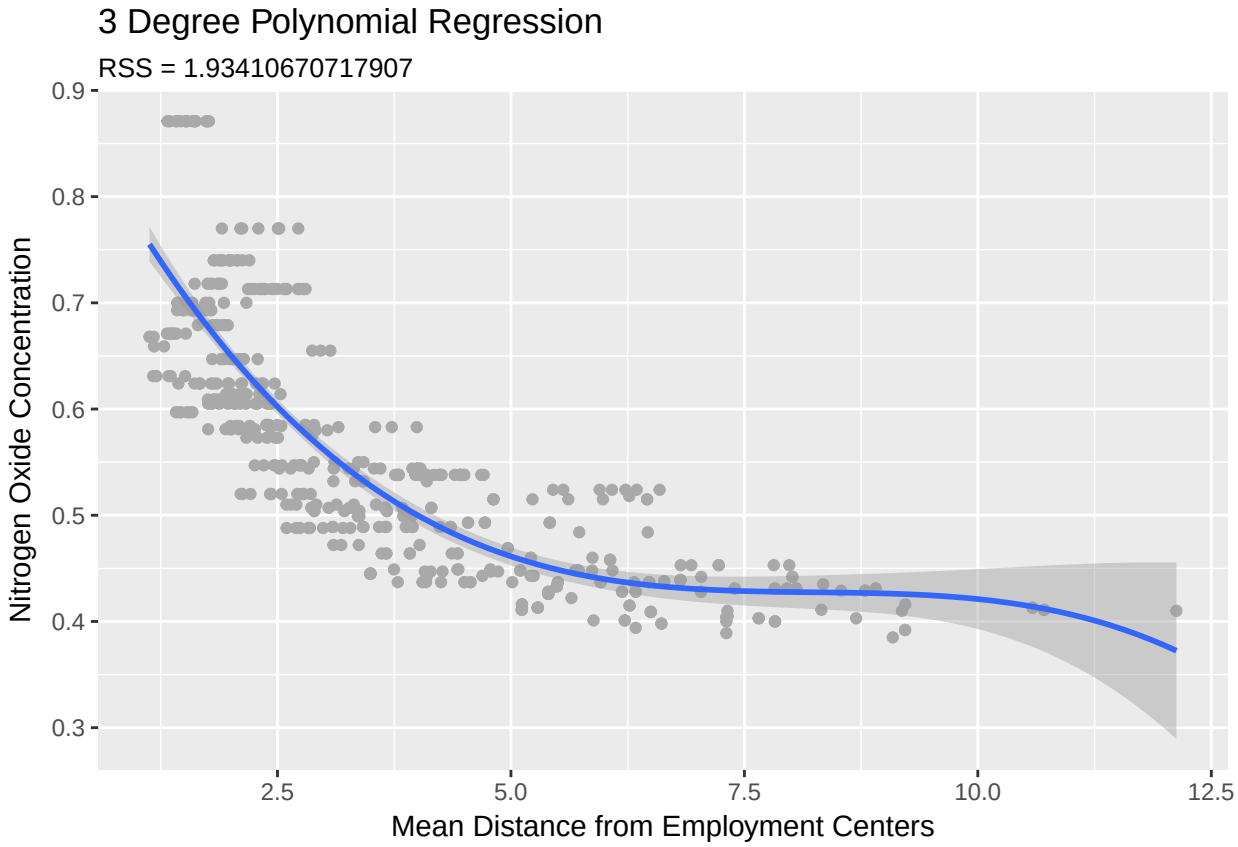


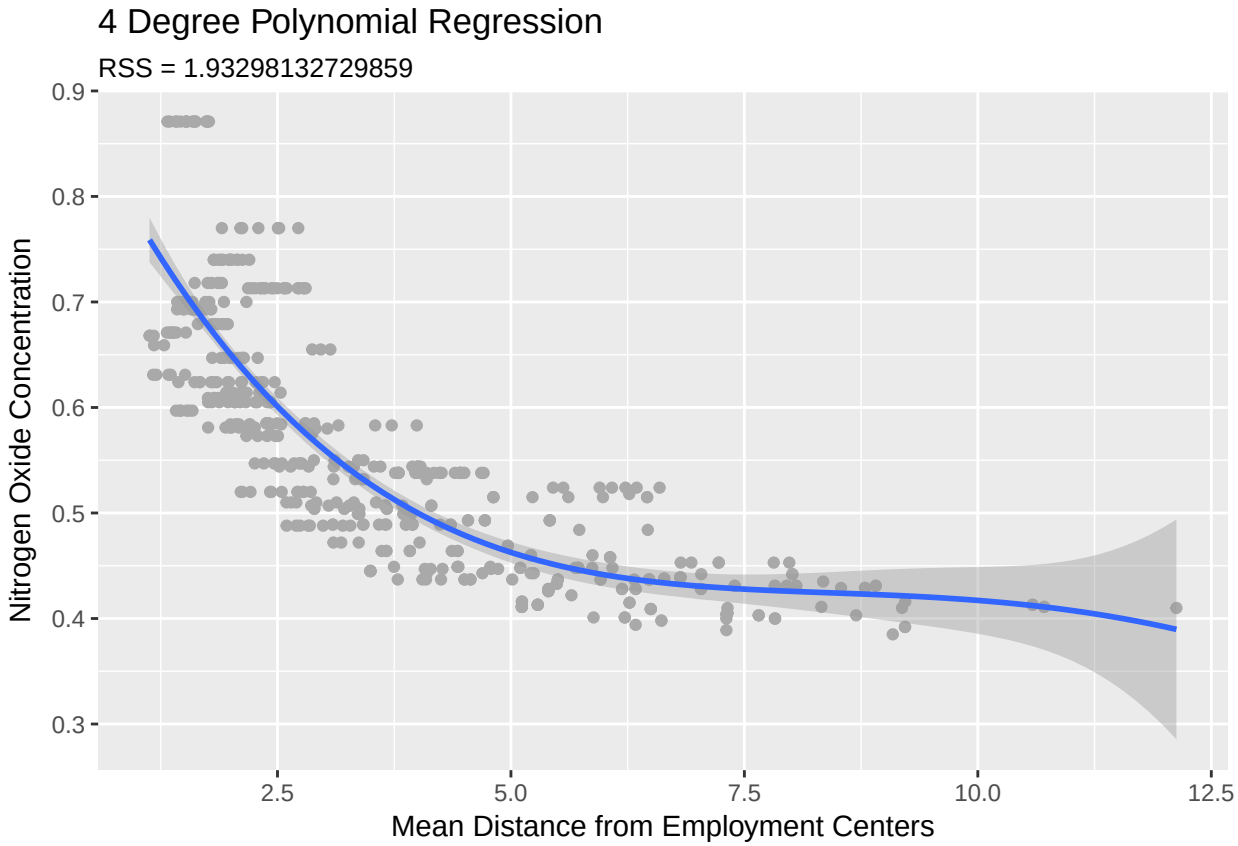
- **B**. The RSS is printed as a subtitle below each plot.

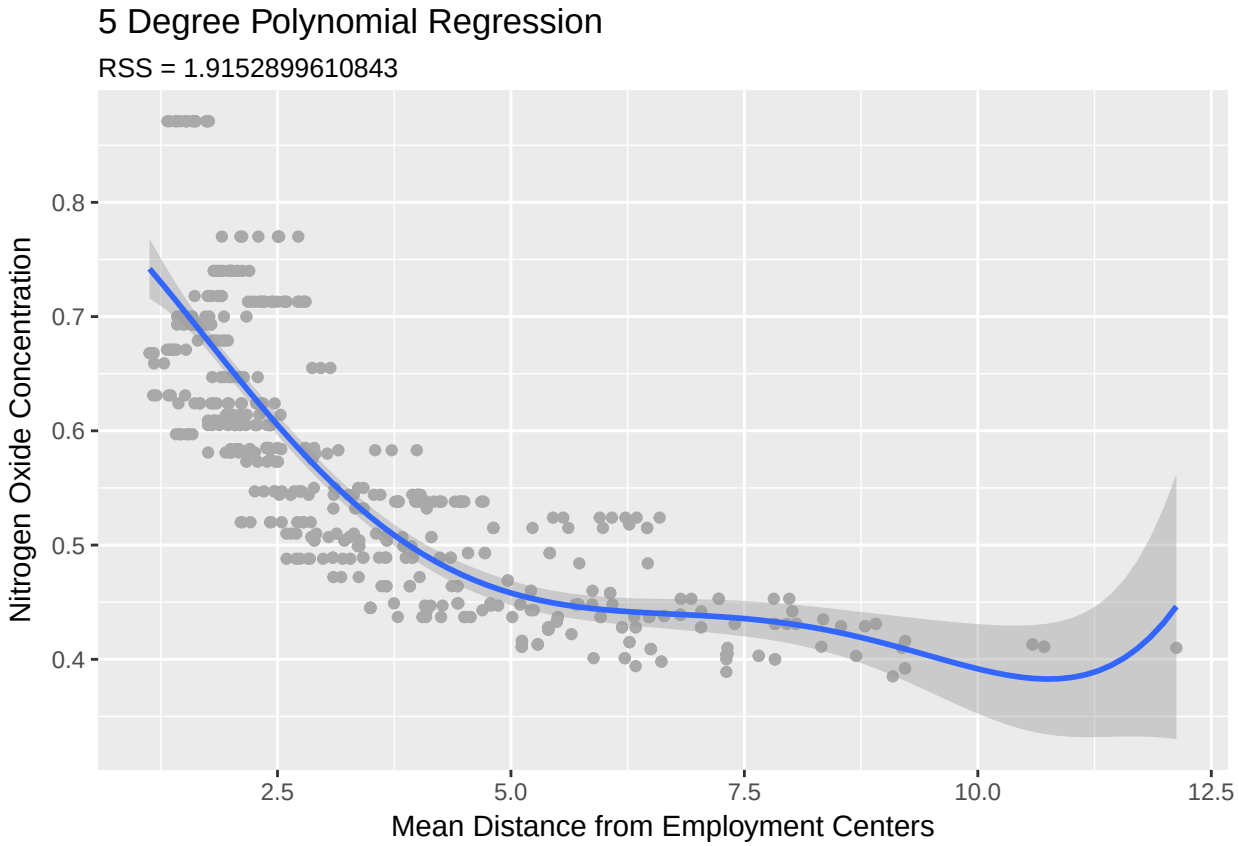
```r
# Plots of i-degree polynomial fits for i in 1:10
for (i in 1:10) {
    fit <- lm(nox ~ poly(dis, i), data = Boston)
    error <- sum(resid(fit)^2)
    g <- ggplot(Boston, aes(x = dis, y = nox)) +
        geom_point(color = 'darkgrey') +
        stat_smooth(method = 'lm',
                    formula = y ~ poly(x, i)) +
        guides(color=guide_legend(title = paste(i, "Degree of Polynomial"))) +
        ggtitle(paste(i, "Degree Polynomial Regression"),
                subtitle = paste("RSS =", error)) +
        xlab("Mean Distance from Employment Centers") +
        ylab("Nitrogen Oxide Concentration")
    print(g)
}
```

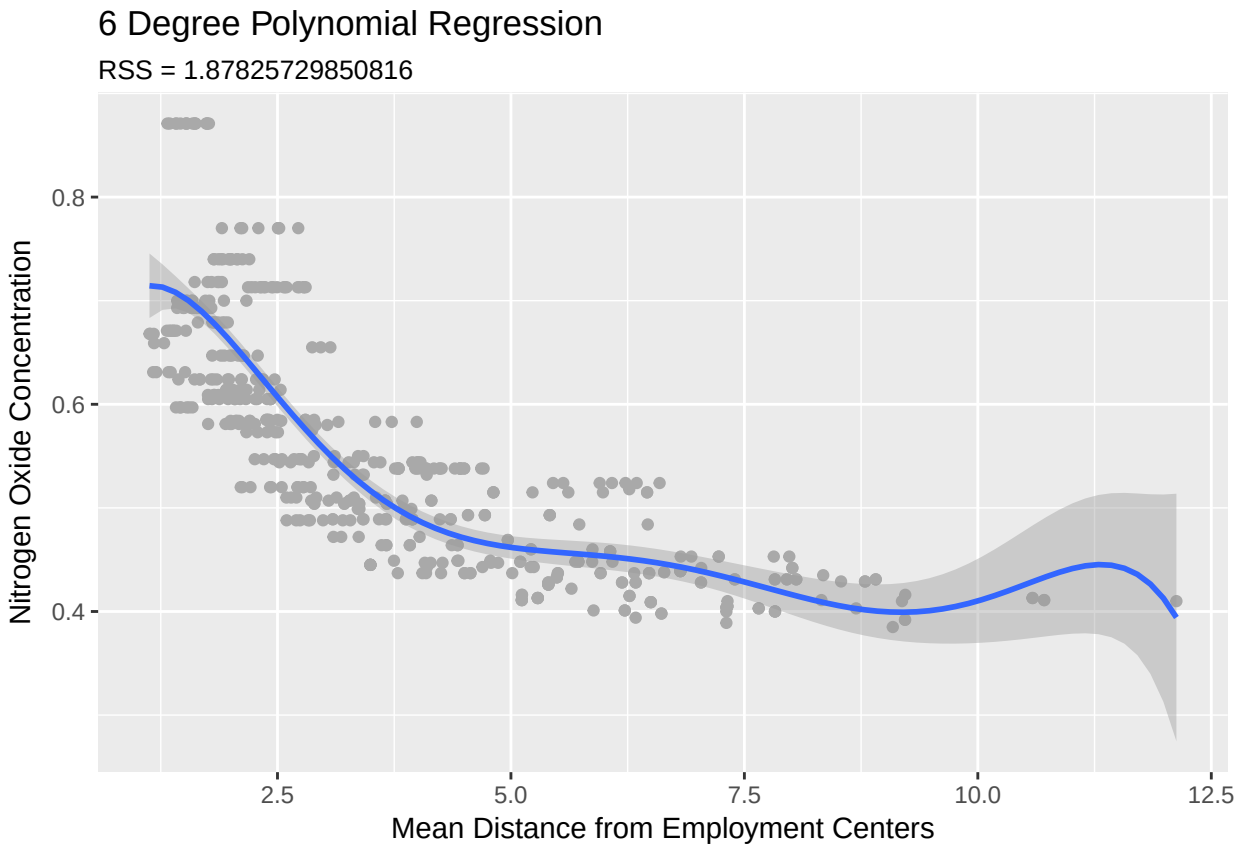


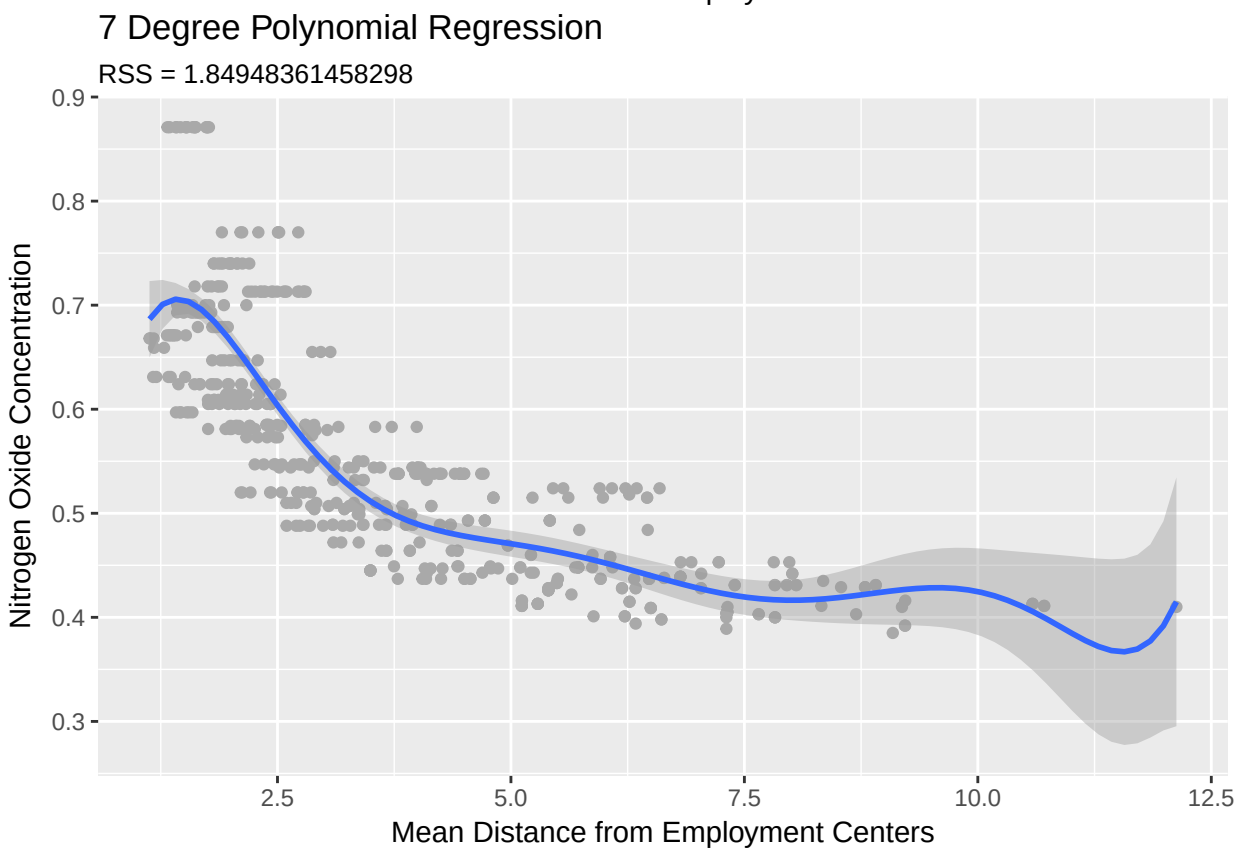

2 Degree Polynomial Regression
RSS = 2.03526186893526
Nitrogen Oxide Concentration
0.8
0.7
0.6
0.5
0.4
2.5
5.0
7.5
10.0
12.5
Mean Distance from Employment Centers

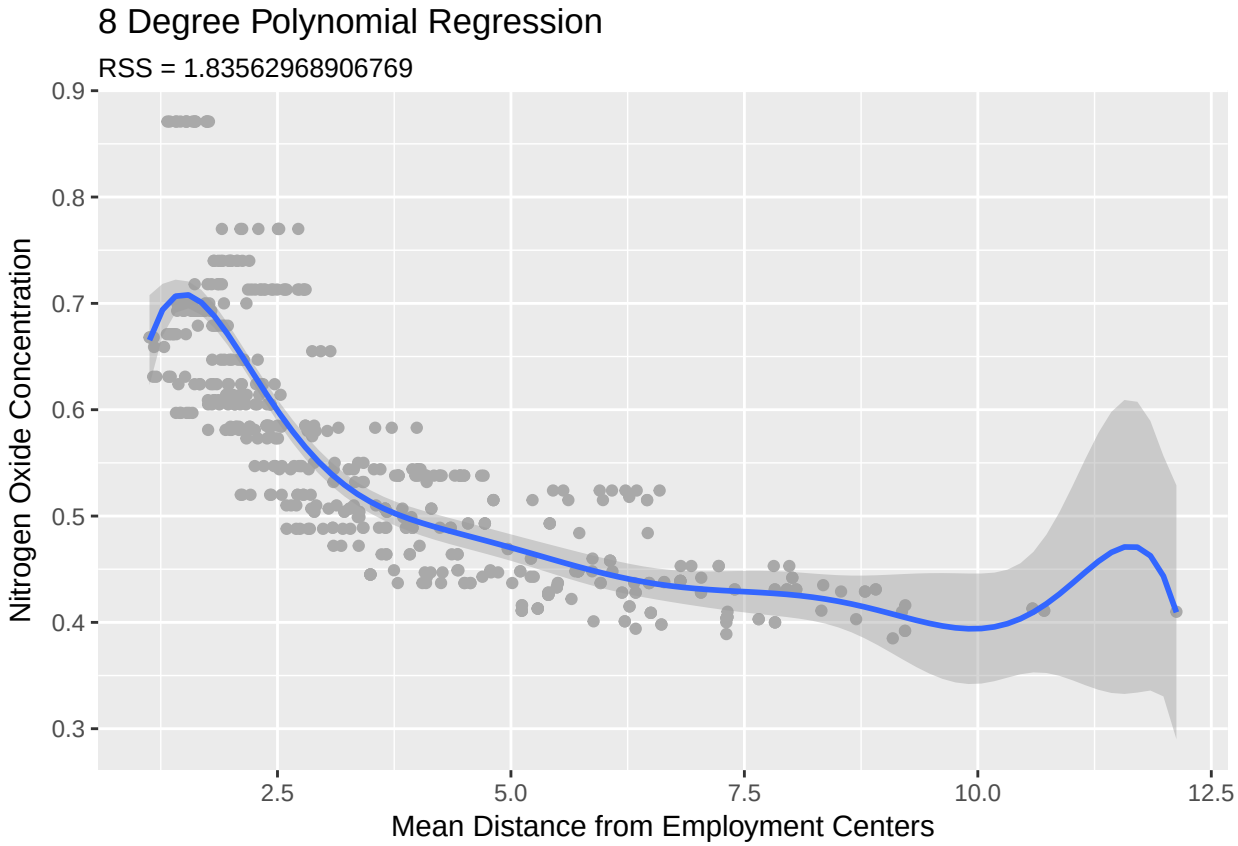

3 Degree Polynomial Regression
RSS = 1.93410670717907
Nitrogen Oxide Concentration
0.9
0.8
0.7
0.6
0.5
0.4
0.3
2.5
5.0
7.5
10.0
12.5
Mean Distance from Employment Centers

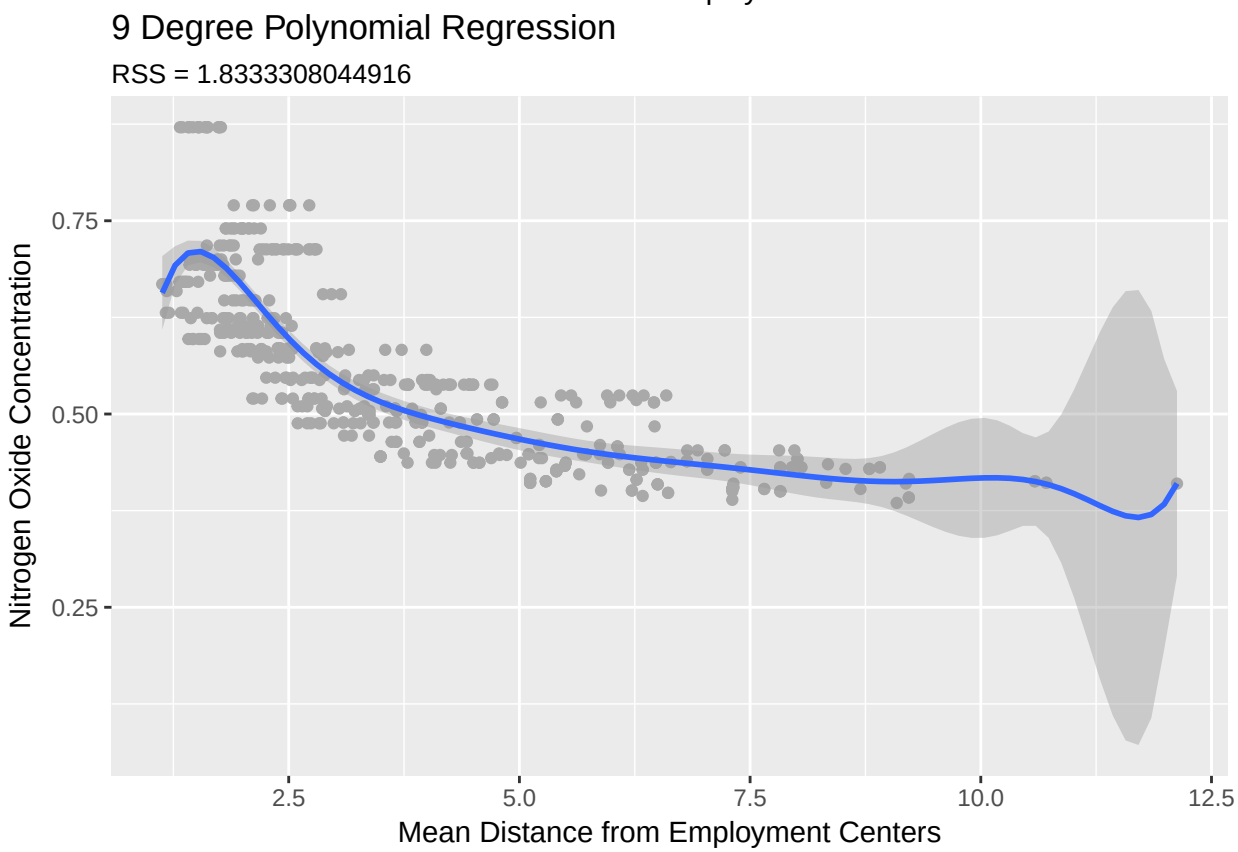

4 Degree Polynomial Regression
RSS = 1.93298132729859
Nitrogen Oxide Concentration
0.9
0.8
0.7
0.6
0.5
0.4
0.3
2.5
5.0
7.5
10.0
12.5
Mean Distance from Employment Centers

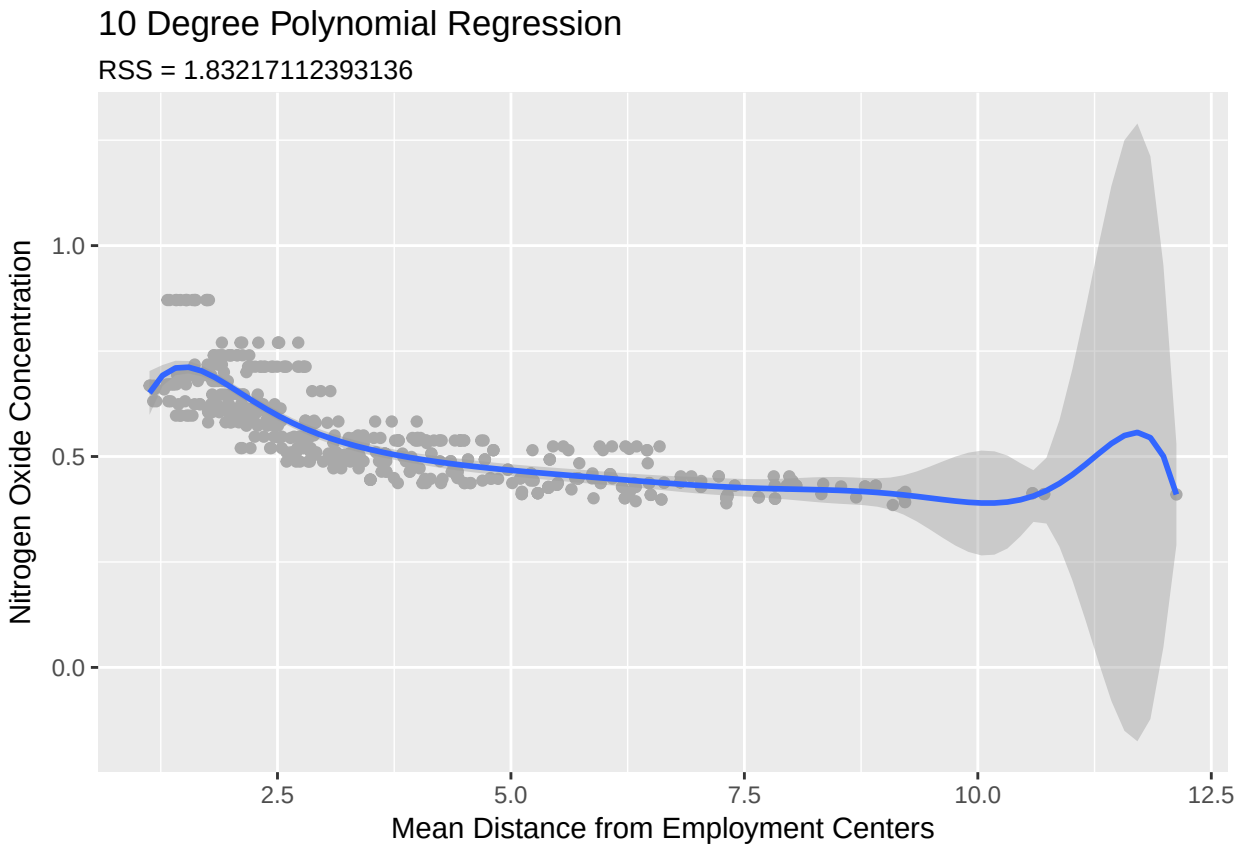

5 Degree Polynomial Regression
RSS = 1.9152899610843
Nitrogen Oxide Concentration
0.8
0.7
0.6
0.5
0.4
2.5
5.0
7.5
10.0
12.5
Mean Distance from Employment Centers


6 Degree Polynomial Regression
RSS = 1.87825729850816
Nitrogen Oxide Concentration
0.8
0.6
0.4
2.5
5.0
7.5
10.0
12.5
Mean Distance from Employment Centers


7 Degree Polynomial Regression
RSS = 1.84948361458298
Nitrogen Oxide Concentration
0.9
0.8
0.7
0.6
0.5
0.4
0.3
2.5
5.0
7.5
10.0
12.5
Mean Distance from Employment Centers


8 Degree Polynomial Regression
RSS = 1.83562968906769
Nitrogen Oxide Concentration
0.9
0.8
0.7
0.6
0.5
0.4
0.3
2.5
5.0
7.5
10.0
12.5
Mean Distance from Employment Centers

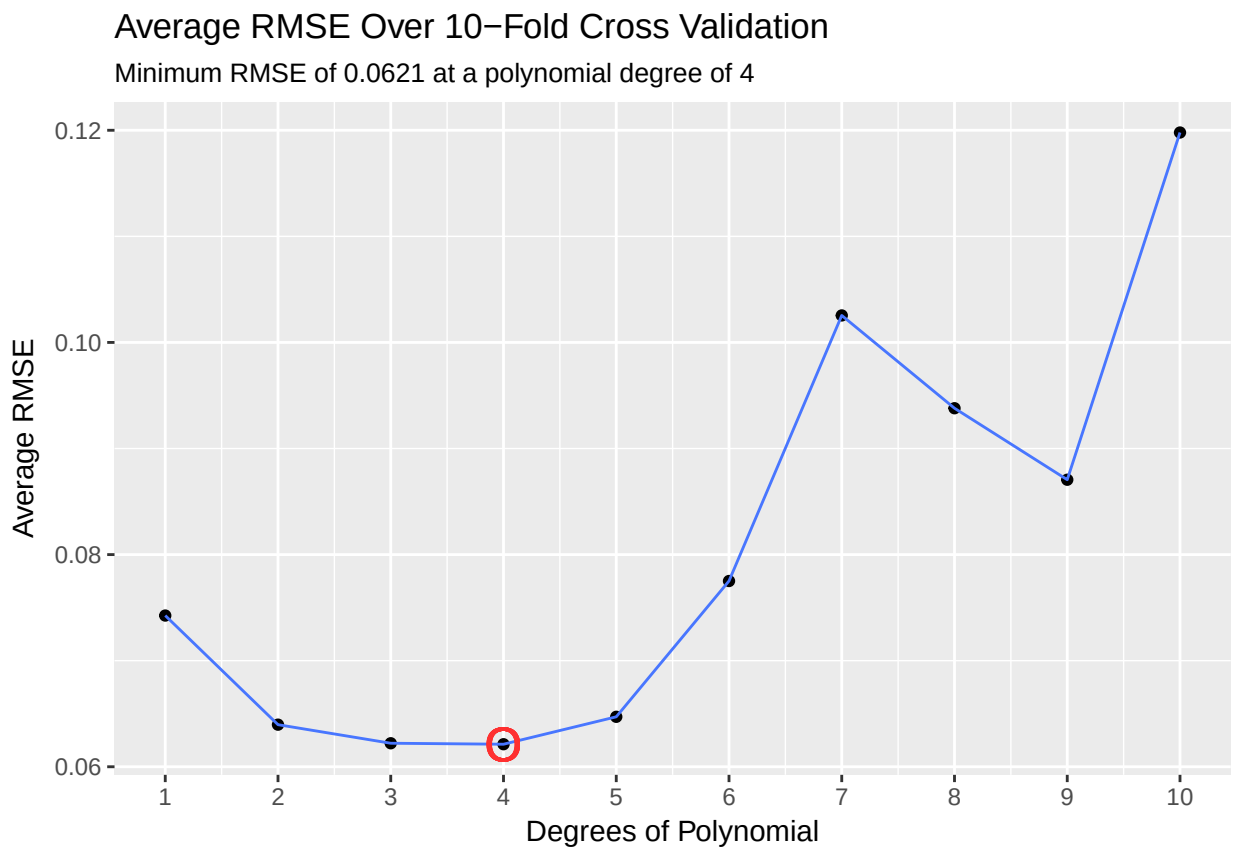

9 Degree Polynomial Regression
RSS = 1.8333308044916
Nitrogen Oxide Concentration
0.75
0.50
0.25
2.5
5.0
7.5
10.0
12.5
Mean Distance from Employment Centers



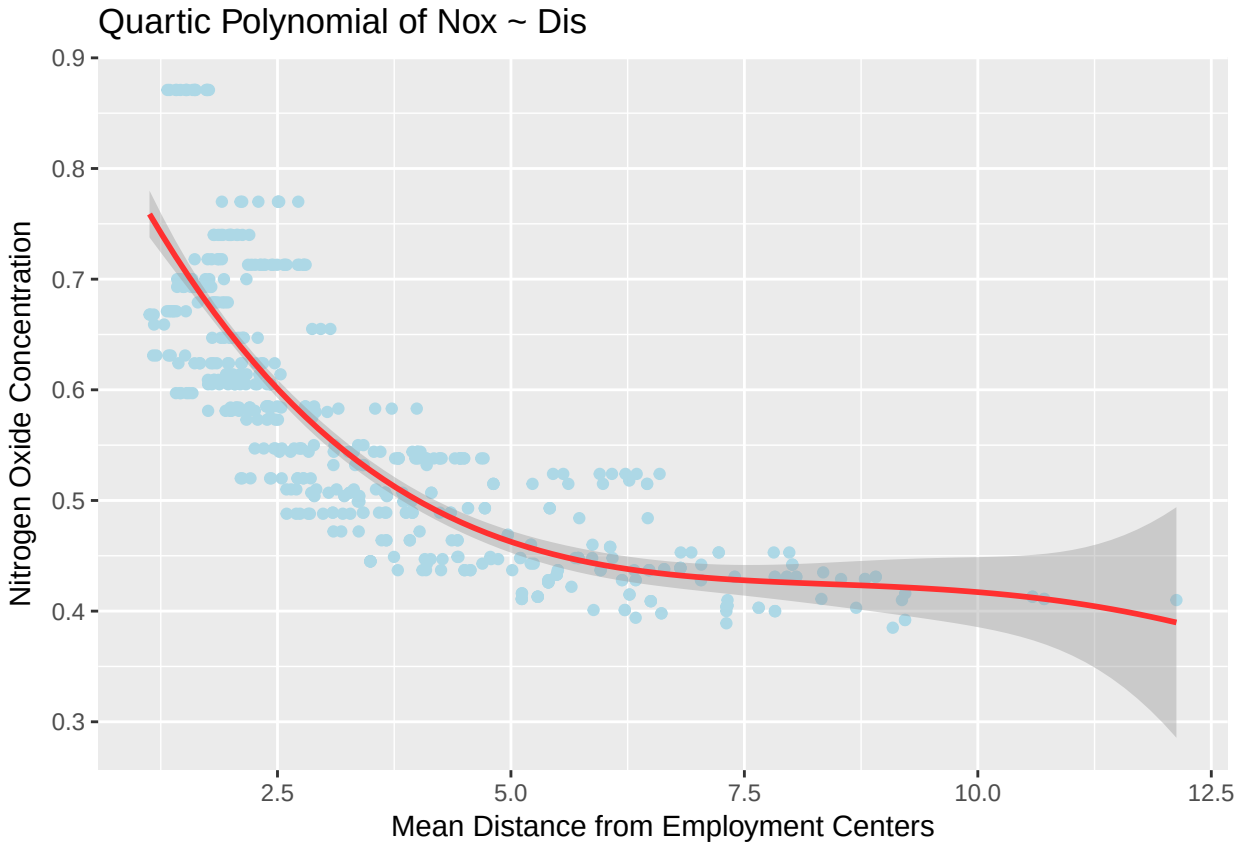
- **C**. 10-fold CV illustrates that the quartic polynomial has the lowest RMSE, shown by the typical U-shape of the CV error plot. The quartic polynomial is fit to the data in the second plot.

```
# 10-fold CV for each i degree polynomial
set.seed(5)

degrees <- 1:10
cv.errors <- rep(0, 10)
for (i in degrees) {
    fit <- glm(nox ~ poly(dis, i), data = Boston)
    cv.errors[i] <- cv.glm(Boston, fit, K = 10)$delta[1]
}

# Plot of CV errors
g <- ggplot(data.frame(x = 1:10, y = cv.errors), aes(x, sqrt(y))) +
    geom_point() +
    geom_line(color = 'royalblue1') +
    geom_point(x = which.min(cv.errors),
               y = sqrt(cv.errors[which.min(cv.errors)]),
               shape = "O",
               size = 6,
               color = 'firebrick1') +
    scale_x_continuous(breaks = 1:10,
                       labels = as.character(c(1:10))) +
```

```
    ggtitle("Average RMSE Over 10-Fold Cross Validation",
            subtitle = paste("Minimum RMSE of",
                             round(sqrt(cv.errors[which.min(cv.errors)]), 4),
                             "at a polynomial degree of",
                             which.min(cv.errors))) +
    xlab("Degrees of Polynomial") +
    ylab("Average RMSE")
g
```




```
# Plot of quartic polynomial of nox ~ dix
g <- ggplot(Boston, aes(x = dis, y = nox)) +
    geom_point(color = 'lightblue') +
    stat_smooth(method = 'lm',
                formula = y ~ poly(x, 4),
                color = 'firebrick1') +
    ggtitle("Quartic Polynomial of Nox ~ Dis") +
    xlab("Mean Distance from Employment Centers") +
    ylab("Nitrogen Oxide Concentration")
g
```



- **D**. When fitting a regression spline, one can specify the degrees of freedom instead of the knots. When this is the case, the `bs()` function selects `(degrees of freedom) - (degree of polynomial)` knots (in this case 4 - 3 = 1).

```
# Regression spline specifying 4 degrees of freedom
fit.spline <- glm(nox ~ bs(dis, df = 4), data = Boston)
summary(fit.spline)
```

```
##
## Call:
## glm(formula = nox ~ bs(dis, df = 4), data = Boston)
##
## Deviance 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
```

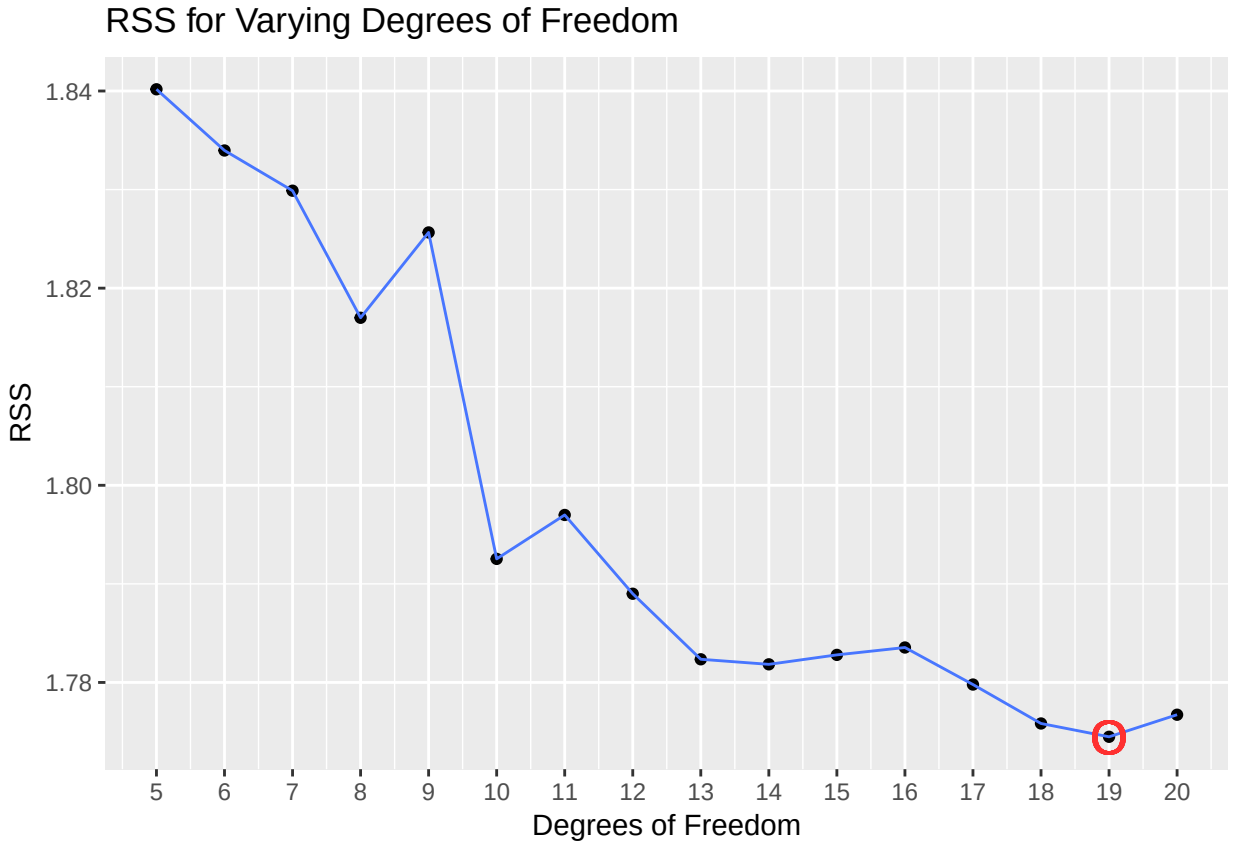
```
##
## (Dispersion parameter for gaussian family taken to be 0.003837874)
##
##     Null deviance: 6.7810  on 505  degrees of freedom
## Residual deviance: 1.9228  on 501  degrees of freedom
## AIC: -1371.9
##
## Number of Fisher Scoring iterations: 2
```

```r
# Plot of cubic spline with 4 DOF
g <- ggplot(Boston, aes(x = dis, y = nox)) +
    geom_point(color = 'lightblue') +
    stat_smooth(method = 'lm',
                formula = y ~ bs(x, df = 4),
                color = 'firebrick1') +
    ggtitle("Cubic Spline with 1 Knot") +
    xlab("Mean Distance From Employment Centers") +
    ylab("Nitrogen Oxide Concentration")
g
```

- **E**. As shown in the plot of the RSS for varying degrees of freedom below, the RSS has a minimum at 19, and it monotonically decreases as the degrees of freedom increases.

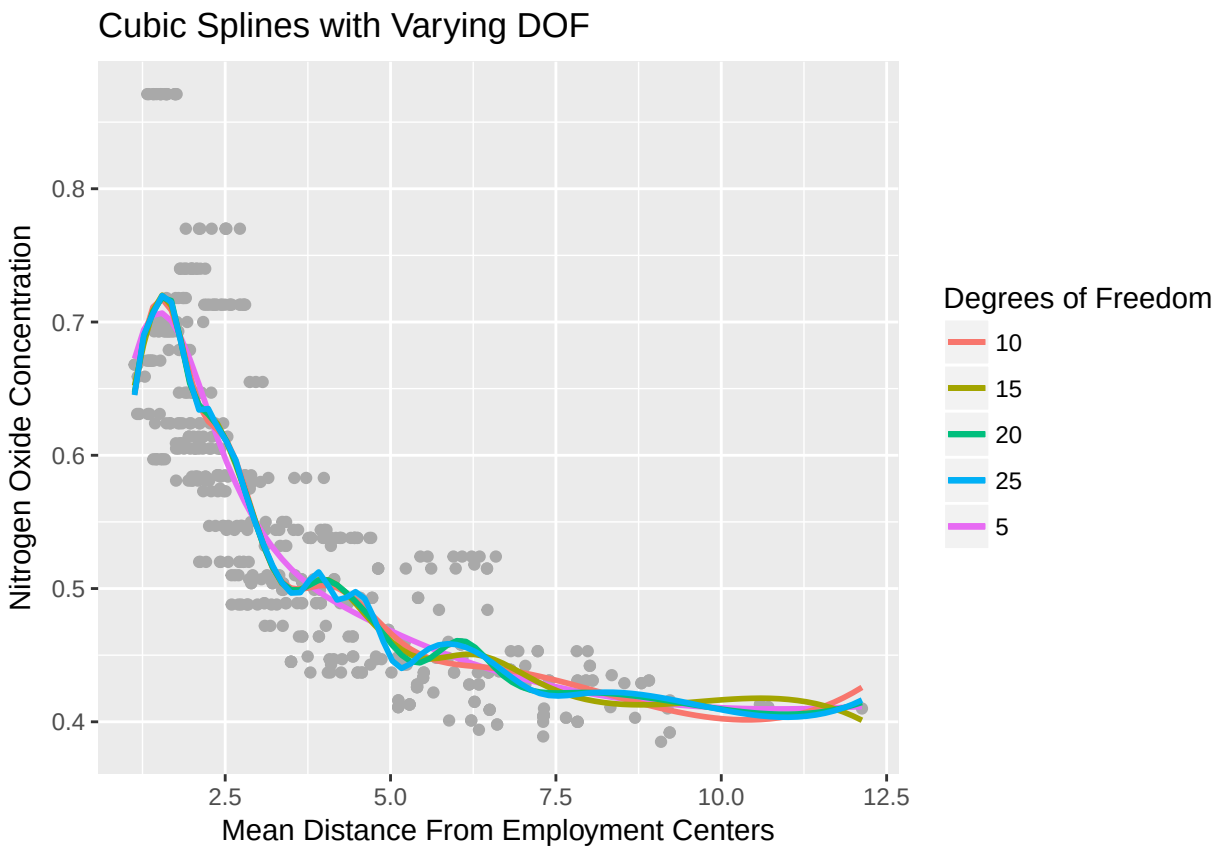
```r
# RSS for range of degrees of freedom
set.seed(5)
```

```r
dof <- seq(5, 20, 1)
rss <- c()
for (i in dof) {
    fit <- glm(nox ~ bs(dis, df = i), data = Boston)
    rss <- c(rss, sum(resid(fit)^2))
}

# Plot of RSS with varying DOF
g <- ggplot(data.frame(x = dof, y = rss), aes(x, y)) +
    geom_point() +
    geom_line(color = 'royalblue1') +
    geom_point(x = dof[which.min(rss)],
               y = rss[which.min(rss)],
               shape = "O",
               size = 6,
               color = 'firebrick1') +
    scale_x_continuous(breaks = dof,
                       labels = dof) +
    ggtitle("RSS for Varying Degrees of Freedom") +
    xlab("Degrees of Freedom") +
    ylab("RSS")
g
```



Cubic Splines with 5, 10, 15, 20 and 25 degrees of freedom are plotted below. As one would expect, as the degrees of freedom increases the fit becomes more flexible.

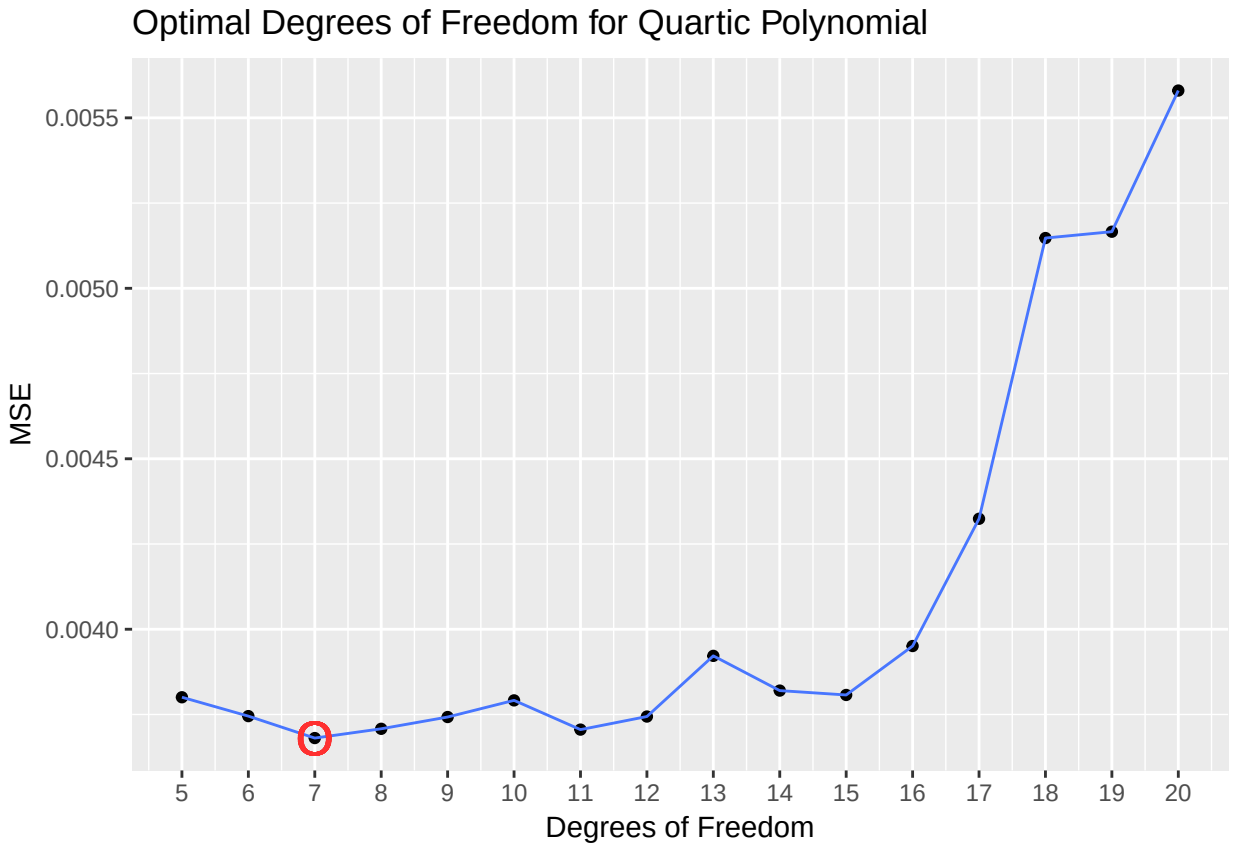
```r
# Plots of cubic splines with varying DOF
g <- ggplot(Boston, aes(x = dis, y = nox)) +
    geom_point(color = 'darkgrey') +
    stat_smooth(method = 'lm',
                se = F,
                formula = y ~ bs(x, df = 5),
                aes(color = '5')) +
    stat_smooth(method = 'lm',
                se = F,
                formula = y ~ bs(x, df = 10),
                aes(color = '10')) +
    stat_smooth(method = 'lm',
                se = F,
                formula = y ~ bs(x, df = 15),
                aes(color = '15')) +
    stat_smooth(method = 'lm',
                se = F,
                formula = y ~ bs(x, df = 20),
                aes(color = '20')) +
    stat_smooth(method = 'lm',
                se = F,
                formula = y ~ bs(x, df = 25),
                aes(color = '25')) +
    guides(color=guide_legend(title = "Degrees of Freedom")) +
    ggtitle("Cubic Splines with Varying DOF") +
    xlab("Mean Distance From Employment Centers") +
    ylab("Nitrogen Oxide Concentration")
g
```



- **F**. As shown in the plot below, the optimal degrees of freedom for a quartic polynomial is 7.

```
# 10-fold CV to select optimal DF of quartic spline
set.seed(5)
dof <- seq(5, 20, 1)
cv.errors <- c()
suppressWarnings(
    for (i in dof) {
        cv.fit <- glm(nox ~ bs(dis, df = i, degree = 4), data = Boston)
        cv.errors <- c(cv.errors, cv.glm(Boston, cv.fit, K = 10)$delta[1])
    }
)

g <- ggplot(data.frame(x = dof, y = cv.errors), aes(x, y)) +
    geom_point() +
    geom_line(color = 'royalblue1') +
    geom_point(x = dof[which.min(cv.errors)],
               y = cv.errors[which.min(cv.errors)],
               shape = "O",
               size = 6,
               color = 'firebrick1') +
    scale_x_continuous(breaks = dof,
                       labels = dof) +
    ggtitle("Optimal Degrees of Freedom for Quartic Polynomial") +
```

```
    xlab("Degrees of Freedom") +
    ylab("MSE")
g
```



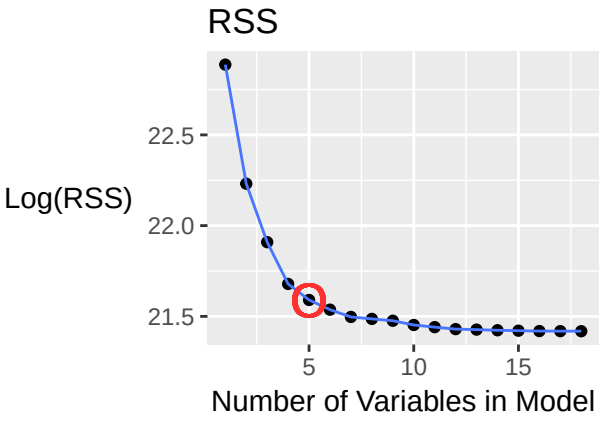
10

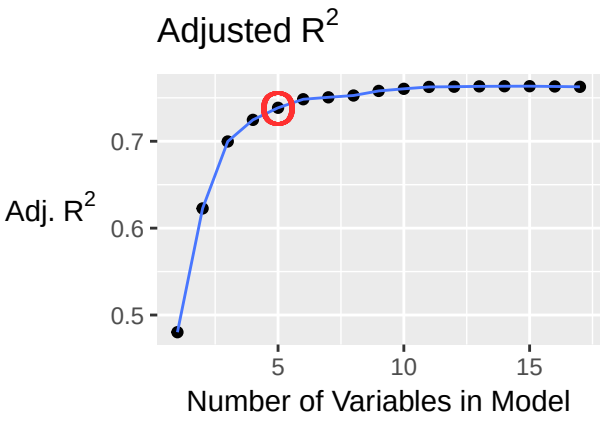
- **A**. In the plot below, the log of the RSS, *Adj.* R^2, Cp and BIC are plotted in the same image. All images show a leveling off around the 5 variable model. One could make the case for the 6 variable model, however I believe the (albiet) slight increase in complexity is not worth the marginal increase in predictive power. The final model coefficients are printed below.

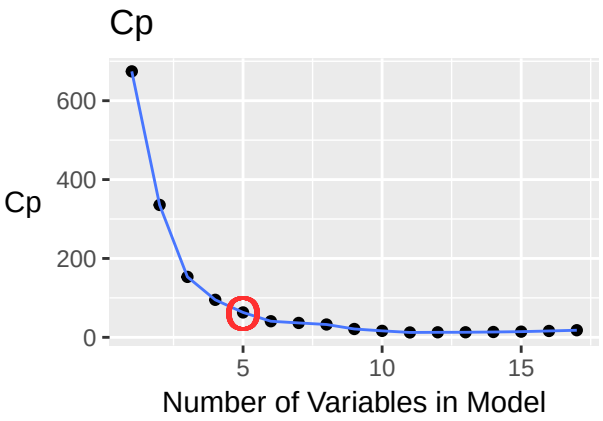
```
detach(Boston)
attach(College)

# Train/test split
set.seed(5)
train <- sample(c(TRUE, FALSE),
                size = nrow(College),
                replace = TRUE,
                prob = c(0.75, 0.25))
train_set <- College[train,]
test_set <- College[!train,]

# Forward Stepwise selection
```

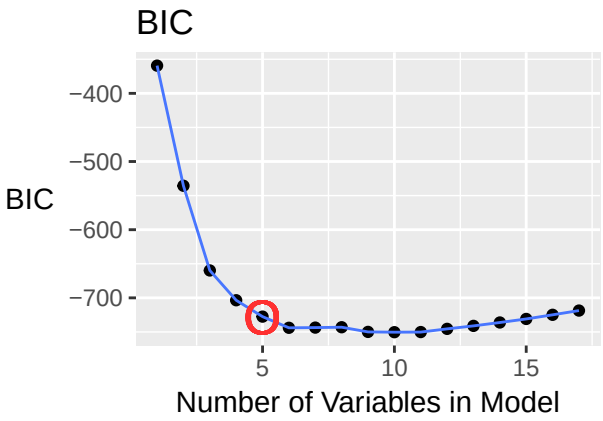
```
suppressPackageStartupMessages(library(leaps))

regfit.forward <- regsubsets(Outstate ~ .,
                             data = train_set,
                             nvmax = ncol(train_set) - 1,
                             method = "forward")
subset.summary <- summary(regfit.forward)

# Plots
suppressPackageStartupMessages(library(gridExtra))

# Plot of RSS
g1 <- ggplot(data.frame(x = 1:length(regfit.forward$rss),
                        y = log(regfit.forward$rss)),
             aes(x, y)) +
    geom_point() +
    geom_line(color = 'royalblue1') +
    geom_point(x = 5,
               y = log(regfit.forward$rss[5]),
               shape = "O",
               size = 6,
               color = 'firebrick1') +
    ggtitle("RSS") +
    ylab("Log(RSS)") +
    theme(axis.title.y = element_text(angle = 360,
                                      vjust = 0.5)) +
    xlab("Number of Variables in Model")

# Plot of Adjusted R^2
g2 <- ggplot(data.frame(x = 1:length(subset.summary$adjr2),
                        y = subset.summary$adjr2),
             aes(x, y)) +
    geom_point() +
    geom_line(color = 'royalblue1') +
    geom_point(x = 5,
               y = subset.summary$adjr2[5],
               shape = "O",
               size = 6,
               color = 'firebrick1') +
    ggtitle(expression(Adjusted~R^{2})) +
    ylab(expression(Adj.~R^{2})) +
    theme(axis.title.y = element_text(angle = 360,
                                      vjust = 0.5)) +
    xlab("Number of Variables in Model")

# Plot of Cp
g3 <- ggplot(data.frame(x = 1:length(subset.summary$cp),
                        y = subset.summary$cp),
             aes(x, y)) +
    geom_point() +
    geom_line(color = 'royalblue1') +
    geom_point(x = 5,
               y = subset.summary$cp[5],
```

```
                shape = "O",
                size = 6,
                color = 'firebrick1') +
    ggtitle("Cp") +
    ylab("Cp") +
    theme(axis.title.y = element_text(angle = 360,
                                      vjust = 0.5)) +
    xlab("Number of Variables in Model")

# Plot of BIC
g4 <- ggplot(data.frame(x = 1:length(subset.summary$bic),
                        y = subset.summary$bic),
             aes(x, y)) +
    geom_point() +
    geom_line(color = 'royalblue1') +
    geom_point(x = 5,
                y = subset.summary$bic[5],
                shape = "O",
                size = 6,
                color = 'firebrick1') +
    ggtitle("BIC") +
    ylab("BIC") +
    theme(axis.title.y = element_text(angle = 360,
                                      vjust = 0.5)) +
    xlab("Number of Variables in Model")

grid.arrange(g1, g2, g3, g4, ncol = 2, nrow = 2)
```









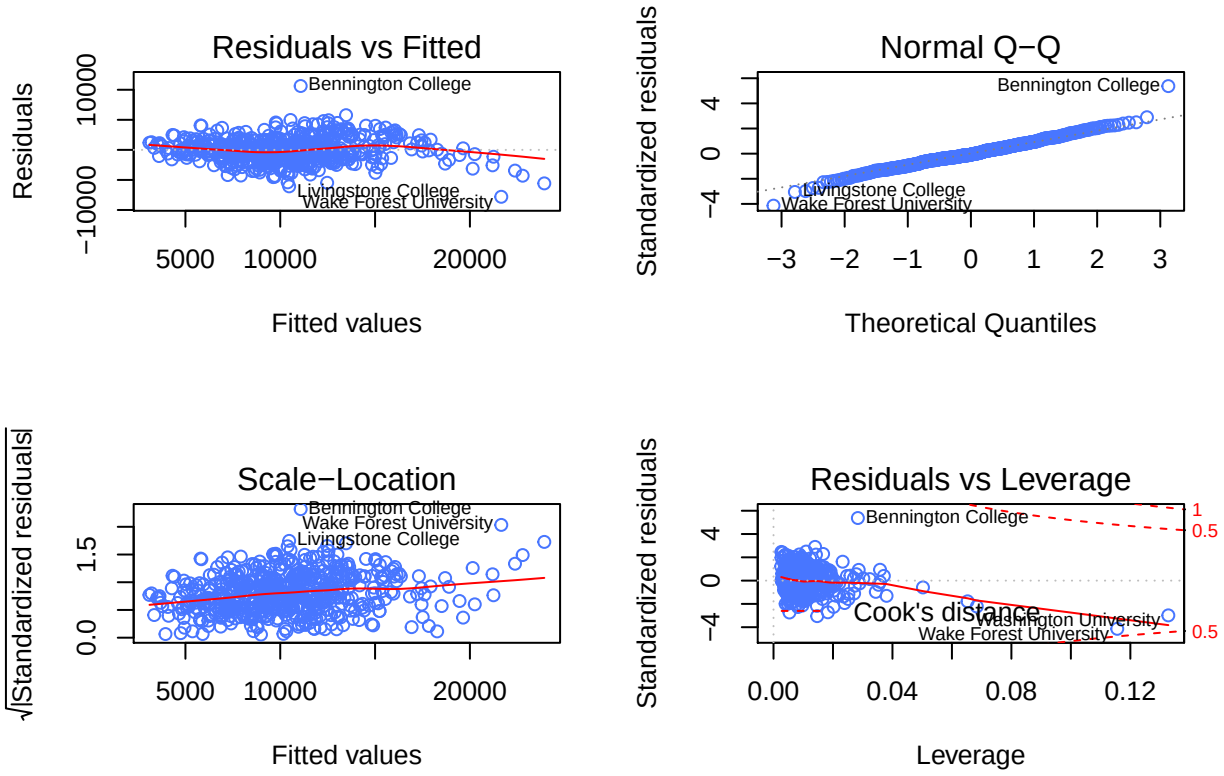

```
# final model coefficients
final.model <- coef(regfit.forward, id = 5)
print(final.model)
```

```
##   (Intercept)    PrivateYes    Room.Board           PhD        Expend
## -3488.4139409  3076.2903223     0.9282057    37.3334553     0.2717432
##     Grad.Rate
##    35.1002841
```

- **B**. Using the variables selected in the part **A** (*Private*, *Room.Board*, *PhD*, *Expend* and *Grad.Rate*), it is clear that the model fits the data relatively well. There is a slight dip in the residuals at higher predicted values, although it is only minor (shown in the upper left hand plot displayed below).

  Note that I am starting with a GLM (not specifically a GAM with non-linear terms) to ensure that a more complex model is warranted.

```
# General Linear Model
glm.fit <- lm(Outstate ~ Private + Room.Board + PhD + Expend + Grad.Rate,
              data = train_set)

par(mfrow = c(2,2))
plot(glm.fit, col = 'royalblue1')
```



- **C**. Using the GLM from above, the testing RMSE is 9273.823, which can be interpretted as the average amount (in dollars) which the predictions are off of the true values.

```
set.seed(5)

# Evaluating model on test set
preds <- predict(glm.fit, newdata = test_set)
rmse <- sqrt(mean(sum(test_set$Outstate - preds)^2))
rmse
```

```
## [1] 9273.823
```

- **D**. In order to determine if a more complex model is appropriate, I create 3 additional GAM's:

1. Quadratic Polynomial - Each term (excluding *Private*, which is qualitative) is given a quadratic term.
2. Quadratic Spline - Each term (excluding *Private*, which is qualitative) is specified to have 4 degrees of freedom and a quadratic term. The results in R choosing 2 suitable knots (dof - degree $\rightarrow$ 4 - 2 = 2) in the data.
3. Cubic Spline - Each term (excluding *Private*, which is qualitative) is specified to have 6 degrees of freedom and a cubic term. The results in R choosing 3 suitable knots (dof - degree $\rightarrow$ 6 - 3 = 3) in the data.

   Looking at the results of ANOVA, there is statistically significant evidence for the quadratic spline with 4 degrees of freedom. There is not sufficient evidence for the Cubic spline model.

   Diving a little deeper into the quadratic spline model, the P-Values for all of the PhD terms are relatively high, indicating there is little evidence that that term has substantial predictive power (if one returns to the output of Forward Stepwise Selection, the 4 variable model excludes *PhD*, as expected)

```
# Build multiple GAM's
glm.poly <- lm(Outstate ~ Private +
                   poly(Room.Board, 2) +
                   poly(PhD, 2) +
                   poly(Expend, 2) +
                   poly(Grad.Rate, 2),
               data = train_set)
glm.spline <- lm(Outstate ~ Private +
                   bs(Room.Board, df = 4, degree = 2) +
                   bs(PhD, df = 4, degree = 2) +
                   bs(Expend, df = 4, degree = 2) +
                   bs(Grad.Rate, df = 4, degree = 2),
               data = train_set)
glm.cubic.spline <- lm(Outstate ~ Private +
                   bs(Room.Board, df = 6, degree = 3) +
                   bs(PhD, df = 6, degree = 3) +
                   bs(Expend, df = 6, degree = 3) +
                   bs(Grad.Rate, df = 6, degree = 3),
               data = train_set)

# ANOVA of all models
anova(glm.fit, glm.poly, glm.spline, glm.cubic.spline)
```

```
## Analysis of Variance Table
##
## Model 1: Outstate ~ Private + Room.Board + PhD + Expend + Grad.Rate
## Model 2: Outstate ~ Private + poly(Room.Board, 2) + poly(PhD, 2) + poly(Expend,
##     2) + poly(Grad.Rate, 2)
## Model 3: Outstate ~ Private + bs(Room.Board, df = 4, degree = 2) + bs(PhD,
##     df = 4, degree = 2) + bs(Expend, df = 4, degree = 2) + bs(Grad.Rate,
##     df = 4, degree = 2)
## Model 4: Outstate ~ Private + bs(Room.Board, df = 6, degree = 3) + bs(PhD,
##     df = 6, degree = 3) + bs(Expend, df = 6, degree = 3) + bs(Grad.Rate,
##     df = 6, degree = 3)
##   Res.Df        RSS Df Sum of Sq       F    Pr(>F)
## 1    561 2256103346
```

```
## 2    557 1953605479  4 302497867 22.4486 < 2.2e-16 ***
## 3    549 1864068579  8  89536900  3.3223  0.001014 **
## 4    541 1822510713  8  41557866  1.5420  0.139765
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(glm.spline)
```

```
##
## Call:
## lm(formula = Outstate ~ Private + bs(Room.Board, df = 4, degree = 2) +
##     bs(PhD, df = 4, degree = 2) + bs(Expend, df = 4, degree = 2) +
##     bs(Grad.Rate, df = 4, degree = 2), data = train_set)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -6199  -1141    -75   1304   8050
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                         3777.70    1611.92   2.344  0.01945
## PrivateYes                          2728.16     225.79  12.083  < 2e-16
## bs(Room.Board, df = 4, degree = 2)1  -900.05    1099.82  -0.818  0.41351
## bs(Room.Board, df = 4, degree = 2)2  1091.19     821.02   1.329  0.18438
## bs(Room.Board, df = 4, degree = 2)3  2400.00    1067.32   2.249  0.02493
## bs(Room.Board, df = 4, degree = 2)4  2706.38    1328.64   2.037  0.04213
## bs(PhD, df = 4, degree = 2)1        -1803.62    1407.40  -1.282  0.20055
## bs(PhD, df = 4, degree = 2)2         -347.53    1014.97  -0.342  0.73218
## bs(PhD, df = 4, degree = 2)3         -194.67    1167.67  -0.167  0.86765
## bs(PhD, df = 4, degree = 2)4          -13.84    1174.37  -0.012  0.99060
## bs(Expend, df = 4, degree = 2)1       212.35     883.79   0.240  0.81021
## bs(Expend, df = 4, degree = 2)2      1719.62     664.48   2.588  0.00991
## bs(Expend, df = 4, degree = 2)3     13483.34    1225.15  11.005  < 2e-16
## bs(Expend, df = 4, degree = 2)4      4036.09    1391.37   2.901  0.00387
## bs(Grad.Rate, df = 4, degree = 2)1   1162.21    1047.35   1.110  0.26763
## bs(Grad.Rate, df = 4, degree = 2)2   1571.57     724.36   2.170  0.03047
## bs(Grad.Rate, df = 4, degree = 2)3   3810.55     885.38   4.304 1.99e-05
## bs(Grad.Rate, df = 4, degree = 2)4   2148.19     835.52   2.571  0.01040
##
## (Intercept)                         *
## PrivateYes                          ***
## bs(Room.Board, df = 4, degree = 2)1
## bs(Room.Board, df = 4, degree = 2)2
## bs(Room.Board, df = 4, degree = 2)3 *
## bs(Room.Board, df = 4, degree = 2)4 *
## bs(PhD, df = 4, degree = 2)1
## bs(PhD, df = 4, degree = 2)2
## bs(PhD, df = 4, degree = 2)3
## bs(PhD, df = 4, degree = 2)4
## bs(Expend, df = 4, degree = 2)1
## bs(Expend, df = 4, degree = 2)2     **
## bs(Expend, df = 4, degree = 2)3     ***
## bs(Expend, df = 4, degree = 2)4     **
## bs(Grad.Rate, df = 4, degree = 2)1
## bs(Grad.Rate, df = 4, degree = 2)2  *
```

```
## bs(Grad.Rate, df = 4, degree = 2)3  ***
## bs(Grad.Rate, df = 4, degree = 2)4  *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1843 on 549 degrees of freedom
## Multiple R-squared:  0.7859, Adjusted R-squared:  0.7792
## F-statistic: 118.5 on 17 and 549 DF,  p-value: < 2.2e-16
```

11

- **A**.

```
y <- rnorm(100, mean = 10, sd = 50)
x1 <- rnorm(100)
x2 <- rnorm(100)
```

- **B**.

```
beta_1 <- 5
```

- **C**.

```
a <- y - beta_1*x1
beta_2 <- lm(a ~ x2)$coef[2]
```

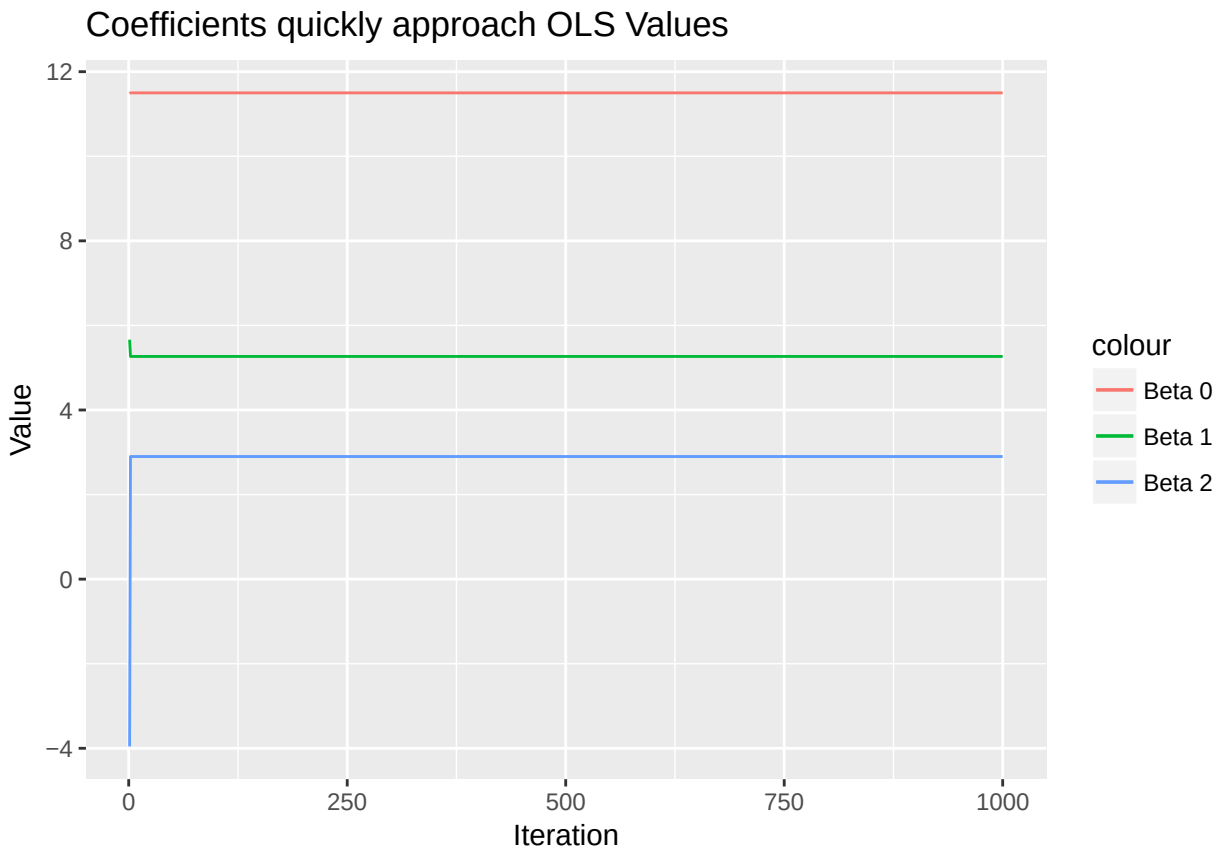
- **D**.

```
a <- y - beta_2*x2
beta_1 <- lm(a ~ x1)$coef[2]
```

- **E**.

```
# Iteratively hone in on coefficient values
beta_0 <- rep(0, 1000)
beta_1 <- rep(0, 1000)
beta_2 <- rep(0, 1000)
beta_1[1] <- 100
for (i in 1:1000) {
    a <- y - beta_1[i]*x1
    beta_2[i] <- lm(a ~ x2)$coef[2]

    a <- y - beta_2[i]*x2
```

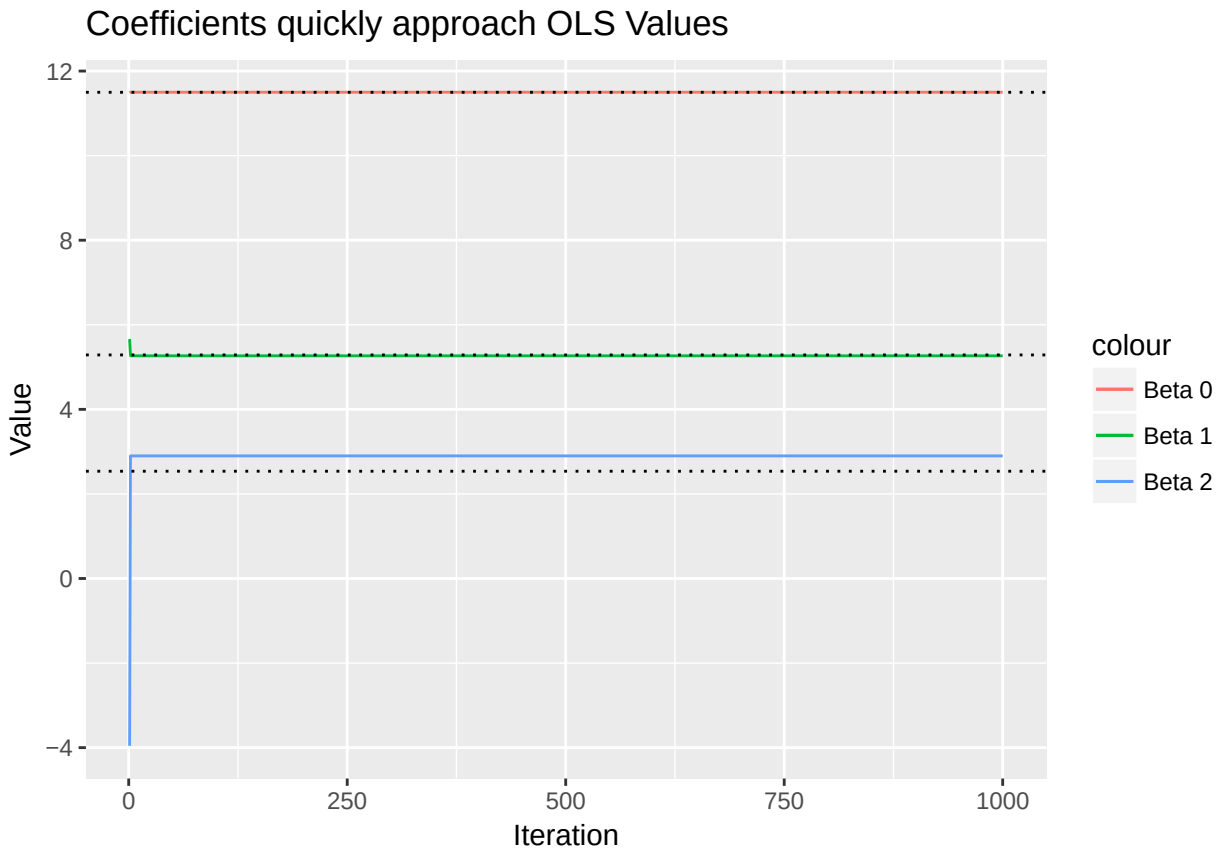
```
    beta_1[i] <- lm(a ~ x1)$coef[2]

    beta_0[i] <- lm(a ~ x1)$coef[1]
}

# Plot beta_0, beta_1 and beta_2 values
g <- ggplot(data.frame(x = 1:1000,
                       b1 = beta_1,
                       b2 = beta_2,
                       b0 = beta_0),
            aes(x = x, y = b2)) +
    geom_line(aes(y = b1, color = "Beta 1")) +
    geom_line(aes(y = b2, color = "Beta 2")) +
    geom_line(aes(y = b0, color = "Beta 0")) +
    ggtitle("Coefficients quickly approach OLS Values") +
    ylab("Value") +
    xlab("Iteration")
g
```



- **F**.

```
lm.fit <- lm(y ~ x1 + x2)

# Reproduce plot from above, with dotted lines representing OLS estimates
```

```r
g <- ggplot(data.frame(x = 1:1000,
                       b1 = beta_1,
                       b2 = beta_2,
                       b0 = beta_0),
            aes(x = x, y = b2)) +
    geom_line(aes(y = b1, color = "Beta 1")) +
    geom_line(aes(y = b2, color = "Beta 2")) +
    geom_line(aes(y = b0, color = "Beta 0")) +
    geom_hline(yintercept = coef(lm.fit)[1], lty = 3) +
    geom_hline(yintercept = coef(lm.fit)[2], lty = 3) +
    geom_hline(yintercept = coef(lm.fit)[3], lty = 3) +
    ggtitle("Coefficients quickly approach OLS Values") +
    ylab("Value") +
    xlab("Iteration")
g
```



- **G**. Looking at the first few iterations of the beta vectors below, it is shown that the coefficients converged after 2 iterations.

```r
beta_0[1:5]
```

```
## [1] 11.48265 11.49892 11.49892 11.49892 11.49892
```

```r
beta_1[1:5]
```

```
## [1] 5.662012 5.266228 5.266228 5.266228 5.266228
```

```
beta_2[1:5]
```

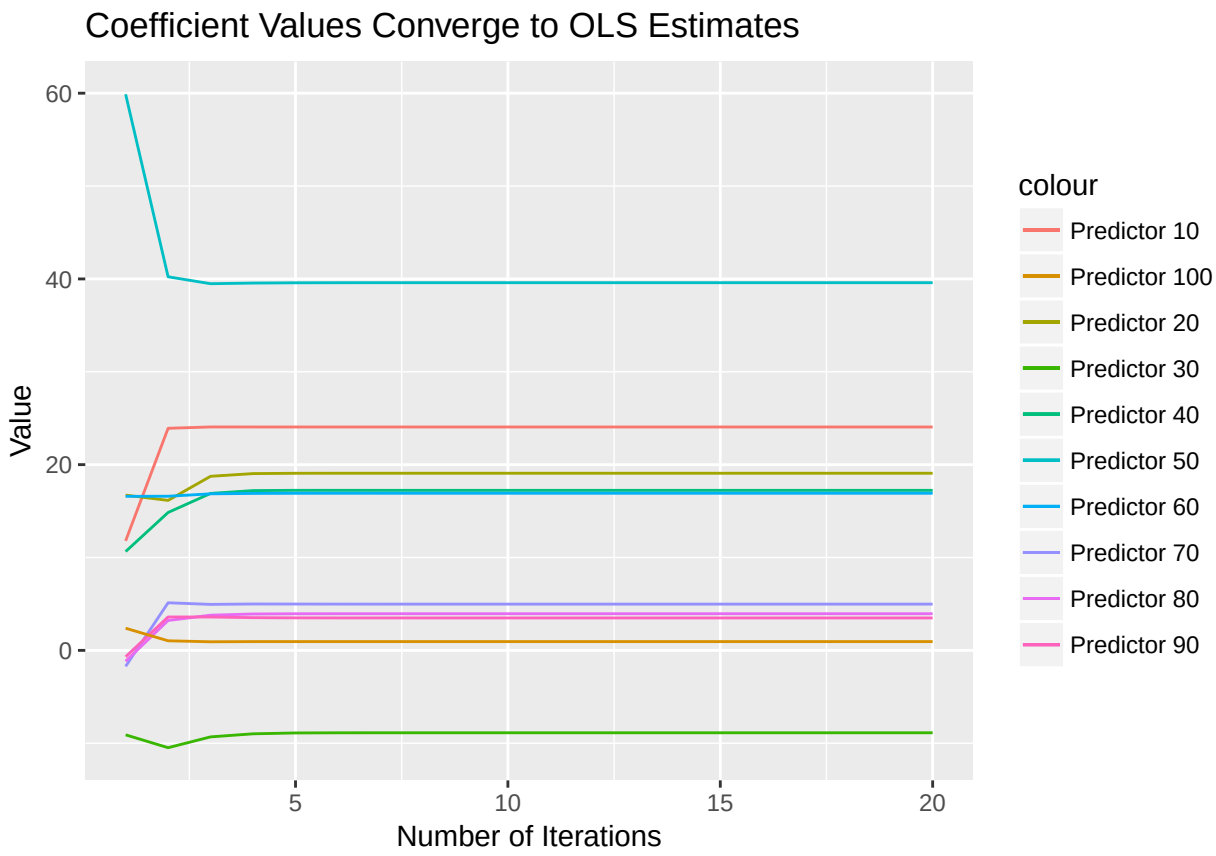
```
## [1] -3.957181  2.899620  2.899620  2.899620  2.899620
```

12

Looking at the two plots below, it is clear that the coefficients converge after roughly 5 iterations.

```
# Create toy matrix
p = 100
n = 1000
X <- matrix(nrow = n, ncol = p, data = 0)

# Create true values
true.coefs <- rnorm(100, mean = 10, sd = 25)

# Populate toy matrix with random numbers
for (col in 1:p) {
    X[, col] <- rnorm(n)
}

# Create true response values
y <- X %*% true.coefs + rnorm(n)

# Backfitting method
num.iter <- 20
beta.vector <- rep(0, p)
coef.matrix <- matrix(nrow = num.iter, ncol = p, data = 0)
error.matrix <- matrix(nrow = num.iter, ncol = p, data = 0)
# For i in 1000 iterations...
for (i in 1:num.iter) {
    # for each coefficient in the vector of 100 coefficients...
    for (coef in 1:p) {
        # hold all coefficients except the current coefficient (coef) constant
        # and fit the model specified in question 11-G
        a <- y - X[,-coef] %*% beta.vector[-coef]
        # set the current coefficient (coef) equal to the simple linear
        # regression coefficient of a predicted by coef
        beta.vector[coef] <- lm(a ~ X[,coef])$coef[2]
    }
    # set the ith row of coef.matrix equal to the beta.vector
    # (gives us estimates of all coefficients at the current iteration)
    coef.matrix[i,] <- beta.vector
    # set the ith row of error.matrix to be the absolute difference between
    # the current beta vector and the true values
    error.matrix[i,] <- abs(true.coefs - beta.vector)
}

# Add iteration number as column to both data frames
coef.df <- data.frame(coef.matrix)
coef.df$iter <- 1:nrow(coef.df)
```

```r
error.df <- data.frame(error.matrix)
error.df$iter <- 1:nrow(error.df)

# Plot example coefficients converging
g <- ggplot(coef.df, aes(x = iter,
                         y = X10)) +
    geom_line(aes(y = X10, color = "Predictor 10")) +
    geom_line(aes(y = X20, color = "Predictor 20")) +
    geom_line(aes(y = X30, color = "Predictor 30")) +
    geom_line(aes(y = X40, color = "Predictor 40")) +
    geom_line(aes(y = X50, color = "Predictor 50")) +
    geom_line(aes(y = X60, color = "Predictor 60")) +
    geom_line(aes(y = X70, color = "Predictor 70")) +
    geom_line(aes(y = X80, color = "Predictor 80")) +
    geom_line(aes(y = X90, color = "Predictor 90")) +
    geom_line(aes(y = X100, color = "Predictor 100")) +
    ggtitle("Coefficient Values Converge to OLS Estimates") +
    ylab("Value") +
    xlab("Number of Iterations")
g
```




```r
# Plot difference between true coefficients and estimates
g <- ggplot(error.df, aes(x = iter,
                         y = X10)) +
    geom_line(aes(y = X10, color = "Predictor 10")) +
    geom_line(aes(y = X20, color = "Predictor 20")) +
```

```
    geom_line(aes(y = X30, color = "Predictor 30")) +
    geom_line(aes(y = X40, color = "Predictor 40")) +
    geom_line(aes(y = X50, color = "Predictor 50")) +
    geom_line(aes(y = X60, color = "Predictor 60")) +
    geom_line(aes(y = X70, color = "Predictor 70")) +
    geom_line(aes(y = X80, color = "Predictor 80")) +
    geom_line(aes(y = X90, color = "Predictor 90")) +
    geom_line(aes(y = X100, color = "Predictor 100")) +
    ggtitle("Difference Between True Coefficient Values and Estimate Approaches Zero") +
    ylab("abs(True Value - Estimate)") +
    xlab("Number of Iterations")
g
```