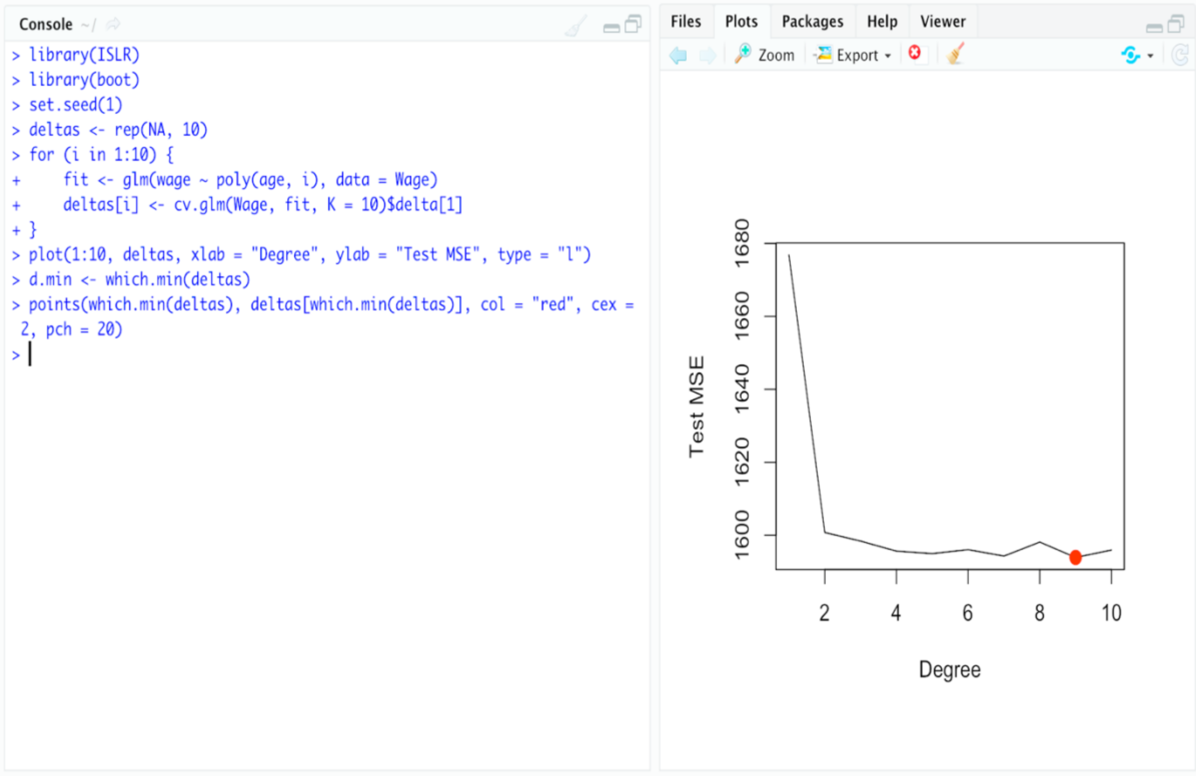
ANUSHA MUPPALLA
16286311

Lab Assignment6

1.(20 points total) In this exercise, you will further analyze the Wage data set considered throughout this chapter.

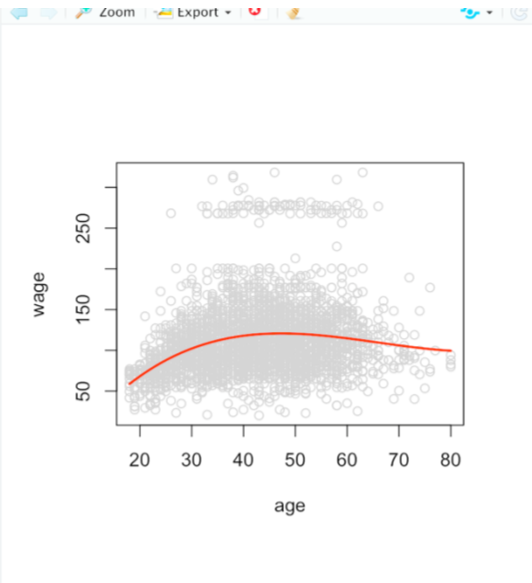
(a) (10 points) Perform polynomial regression to predict wage using age. Use cross- validation to select the optimal degree d for the polynomial. What degree was chosen, and how does this compare to the results of hypothesis testing using ANOVA? Make a plot of the resulting polynomial fit to the data.



```
> fit1 <- lm(wage ~ age, data = Wage)
> fit2 <- lm(wage ~ poly(age, 2), data = Wage)
> fit3 <- lm(wage ~ poly(age, 3), data = Wage)
> fit4 <- lm(wage ~ poly(age, 4), data = Wage)
> fit5 <- lm(wage ~ poly(age, 5), data = Wage)
> anova(fit1, fit2, fit3, fit4, fit5)
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)
  Res.Df     RSS Df Sum of Sq        F    Pr(>F)
1   2998 5022216
2   2997 4793430  1    228786 143.5931 < 2.2e-16 ***
3   2996 4777674  1     15756   9.8888  0.001679 **
4   2995 4771604  1      6070   3.8098  0.051046 .
5   2994 4770322  1      1283   0.8050  0.369682
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> |
```

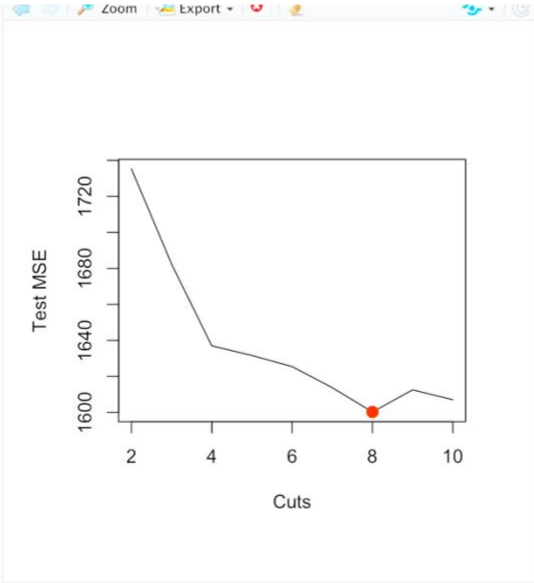

```
> fit1 <- lm(wage ~ age, data = Wage)
> fit2 <- lm(wage ~ poly(age, 2), data = Wage)
> fit3 <- lm(wage ~ poly(age, 3), data = Wage)
> fit4 <- lm(wage ~ poly(age, 4), data = Wage)
> fit5 <- lm(wage ~ poly(age, 5), data = Wage)
> anova(fit1, fit2, fit3, fit4, fit5)
Analysis of Variance Table

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)
  Res.Df     RSS Df Sum of Sq        F    Pr(>F)
1   2998 5022216
2   2997 4793430  1    228786 143.5931 < 2.2e-16 ***
3   2996 4777674  1     15756   9.8888  0.001679 **
4   2995 4771604  1      6070   3.8098  0.051046 .
5   2994 4770322  1      1283   0.8050  0.369682
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```


```
> plot(wage ~ age, data = Wage, col = "darkgrey")
> agelims <- range(Wage$age)
> age.grid <- seq(from = agelims[1], to = agelims[2])
> fit <- lm(wage ~ poly(age, 3), data = Wage)
> preds <- predict(fit, newdata = list(age = age.grid))
> lines(age.grid, preds, col = "red", lwd = 2)
> |
```

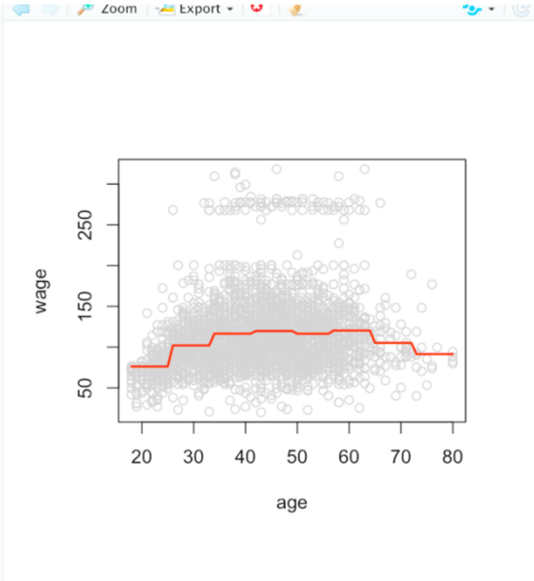


(b) (10 points) Fit a step function to predict wage using age, and perform crossvali- dation to choose the optimal number of cuts. Make a plot of the fit obtained.

```
Model 3: wage ~ poly(age, 3)
Model 4: wage ~ poly(age, 4)
Model 5: wage ~ poly(age, 5)
  Res.Df     RSS Df Sum of Sq        F    Pr(>F)
1   2998 5022216
2   2997 4793430  1    228786 143.5931 < 2.2e-16 ***
3   2996 4777674  1     15756   9.8888  0.001679 **
4   2995 4771604  1      6070   3.8098  0.051046 .
5   2994 4770322  1      1283   0.8050  0.369682
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> plot(wage ~ age, data = Wage, col = "darkgrey")
> agelims <- range(Wage$age)
> age.grid <- seq(from = agelims[1], to = agelims[2])
> fit <- lm(wage ~ poly(age, 3), data = Wage)
> preds <- predict(fit, newdata = list(age = age.grid))
> lines(age.grid, preds, col = "red", lwd = 2)
> cvs <- rep(NA, 10)
> for (i in 2:10) {
+     Wage$age.cut <- cut(Wage$age, i)
+     fit <- glm(wage ~ age.cut, data = Wage)
+     cvs[i] <- cv.glm(Wage, fit, K = 10)$delta[1]
+ }
> plot(2:10, cvs[-1], xlab = "Cuts", ylab = "Test MSE", type = "l")
> d.min <- which.min(cvs)
> points(which.min(cvs), cvs[which.min(cvs)], col = "red", cex = 2, pch =
 20)
>
```



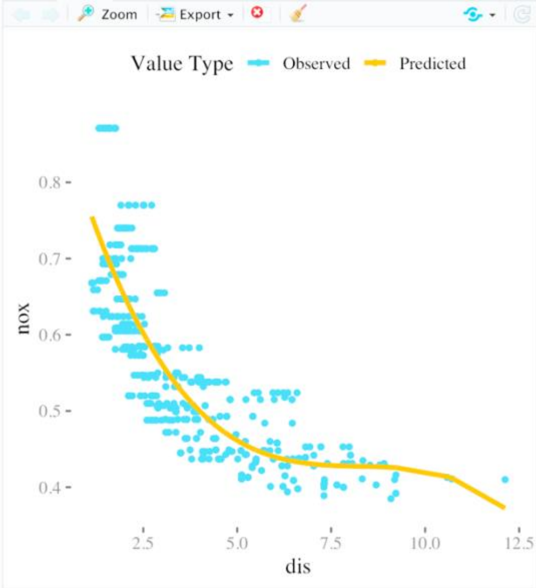

```
3   2996 4777674  1     15756   9.8888  0.001679 **
4   2995 4771604  1      6070   3.8098  0.051046 .
5   2994 4770322  1      1283   0.8050  0.369682
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> plot(wage ~ age, data = Wage, col = "darkgrey")
> agelims <- range(Wage$age)
> age.grid <- seq(from = agelims[1], to = agelims[2])
> fit <- lm(wage ~ poly(age, 3), data = Wage)
> preds <- predict(fit, newdata = list(age = age.grid))
> lines(age.grid, preds, col = "red", lwd = 2)
> cvs <- rep(NA, 10)
> for (i in 2:10) {
+     Wage$age.cut <- cut(Wage$age, i)
+     fit <- glm(wage ~ age.cut, data = Wage)
+     cvs[i] <- cv.glm(Wage, fit, K = 10)$delta[1]
+ }
> plot(2:10, cvs[-1], xlab = "Cuts", ylab = "Test MSE", type = "l")
> d.min <- which.min(cvs)
> points(which.min(cvs), cvs[which.min(cvs)], col = "red", cex = 2, pch =
 20)
> plot(wage ~ age, data = Wage, col = "darkgrey")
> agelims <- range(Wage$age)
> age.grid <- seq(from = agelims[1], to = agelims[2])
> fit <- glm(wage ~ cut(age, 8), data = Wage)
> preds <- predict(fit, data.frame(age = age.grid))
> lines(age.grid, preds, col = "red", lwd = 2)
> |
```



2.(30 points total) 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.

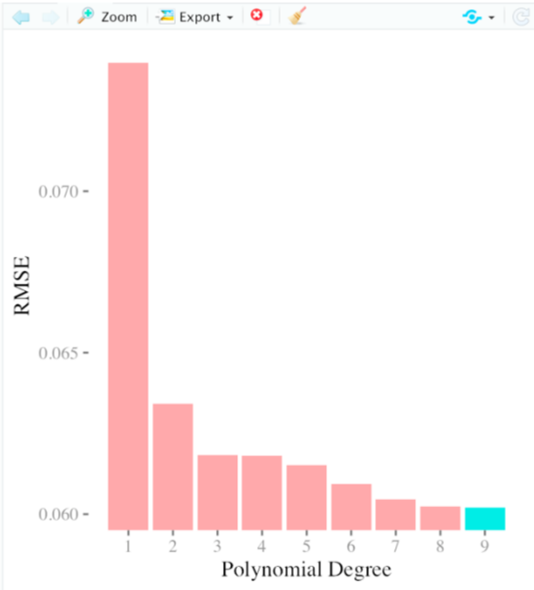
(a) (5 points) Use the poly() function to fit a cubic polynomial regression to predict nox using dis. Report the regression output, and plot the resulting data and polynomial fits.

```
> set.seed(1)
> require(MASS); require(tidyverse); require(ggplot2); require(ggthemes)
> require(broom); require(knitr); require(caret)
> theme_set(theme_tufte(base_size = 14) + theme(legend.position = 'top'))
> data('Boston')
>
> model <- lm(nox ~ poly(dis, 3), data = Boston)
> tidy(model) %>%
+     kable(digits = 3)


|term          | estimate| std.error| statistic| p.value|
|:-------------|--------:|---------:|---------:|-------:|
|(Intercept)   |    0.555|     0.003|   201.021|       0|
|poly(dis, 3)1 |   -2.003|     0.062|   -32.271|       0|
|poly(dis, 3)2 |    0.856|     0.062|    13.796|       0|
|poly(dis, 3)3 |   -0.318|     0.062|    -5.124|       0|
> Boston %>%
+     mutate(pred = predict(model, Boston)) %>%
+     ggplot() +
+     geom_point(aes(dis, nox, col = '1')) +
+     geom_line(aes(dis, pred, col = '2'), size = 1.5) +
+     scale_color_manual(name = 'Value Type',
+                        labels = c('Observed', 'Predicted'),
+                        values = c('#56B4E9', '#E69F00'))
>
>
```

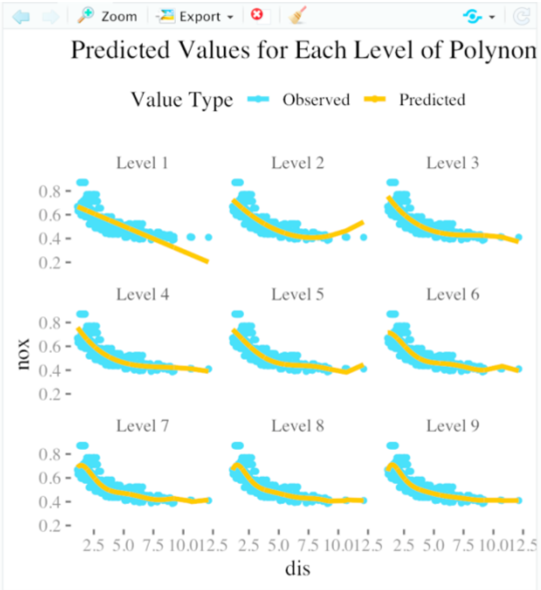


(b) (5 points) Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

```
+     scale_color_manual(name = 'Value Type',
+                        labels = c('Observed', 'Predicted'),
+                        values = c('#56B4E9', '#E69F00'))
>
> errors <- list()
> models <- list()
> pred_df <- data_frame(V1 = 1:506)
Warning message:
`data_frame()` is deprecated, use `tibble()`.
This warning is displayed once per session.
> for (i in 1:9) {
+     models[[i]] <- lm(nox ~ poly(dis, i), data = Boston)
+     preds <- predict(models[[i]])
+     pred_df[[i]] <- preds
+     errors[[i]] <- sqrt(mean((Boston$nox - preds)^2))
+ }
>
> errors <- unlist(errors)
>
> names(pred_df) <- paste('Level', 1:9)
> data_frame(RMSE = errors) %>%
+     mutate(Poly = row_number()) %>%
+     ggplot(aes(Poly, RMSE, fill = Poly == which.min(errors))) +
+     geom_col() +
+     guides(fill = FALSE) +
+     scale_x_continuous(breaks = 1:9) +
+     coord_cartesian(ylim = c(min(errors), max(errors))) +
+     labs(x = 'Polynomial Degree')
> |
```

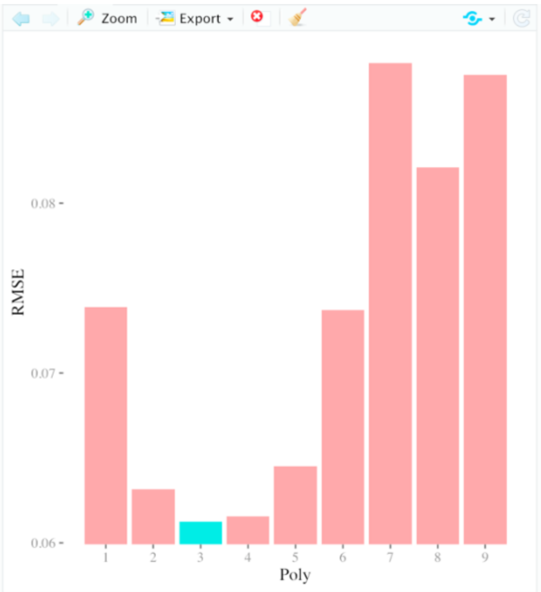


```
+     errors[[i]] <- sqrt(mean((Boston$nox - preds)^2))
+ }
>
> errors <- unlist(errors)
>
> names(pred_df) <- paste('Level', 1:9)
> data_frame(RMSE = errors) %>%
+     mutate(Poly = row_number()) %>%
+     ggplot(aes(Poly, RMSE, fill = Poly == which.min(errors))) +
+     geom_col() +
+     guides(fill = FALSE) +
+     scale_x_continuous(breaks = 1:9) +
+     coord_cartesian(ylim = c(min(errors), max(errors))) +
+     labs(x = 'Polynomial Degree')
> Boston %>%
+     cbind(pred_df) %>%
+     gather(Polynomial, prediction, -(1:14)) %>%
+     mutate(Polynomial = factor(Polynomial,
+                                levels = unique(as.character(Polynomia
l)))) %>%
+     ggplot() +
+     ggtitle('Predicted Values for Each Level of Polynomial') +
+     geom_point(aes(dis, nox, col = '1')) +
+     geom_line(aes(dis, prediction, col = '2'), size = 1.5) +
+     scale_color_manual(name = 'Value Type',
+                        labels = c('Observed', 'Predicted'),
+                        values = c('#56B4E9', '#E69F00')) +
+     facet_wrap(~ Polynomial, nrow = 3)
>
```



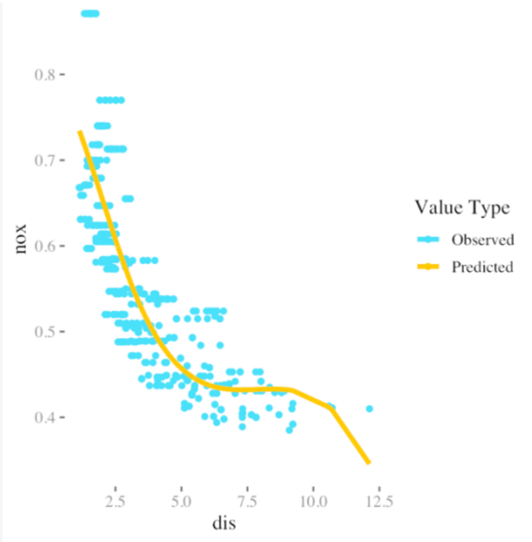
(c) (5 points) Perform cross-validation or another approach to select the optimal degree for the polynomial, and explain your results.

```
+     ggplot() +
+     ggtitle('Predicted Values for Each Level of Polynomial') +
+     geom_point(aes(dis, nox, col = '1')) +
+     geom_line(aes(dis, prediction, col = '2'), size = 1.5) +
+     scale_color_manual(name = 'Value Type',
+                        labels = c('Observed', 'Predicted'),
+                        values = c('#56B4E9', '#E69F00')) +
+     facet_wrap(~ Polynomial, nrow = 3)
> errors <- list()
>
> folds <- sample(1:10, 506, replace = TRUE)
> errors <- matrix(NA, 10, 9)
> for (k in 1:10) {
+     for (i in 1:9) {
+         model <- lm(nox ~ poly(dis, i), data = Boston[folds != k,])
+         pred <- predict(model, Boston[folds == k,])
+         errors[k, i] <- sqrt(mean((Boston$nox[folds == k] - pred)^2))
+     }
+ }
>
> errors <- apply(errors, 2, mean)
>
> data_frame(RMSE = errors) %>%
+     mutate(Poly = row_number()) %>%
+     ggplot(aes(Poly, RMSE, fill = Poly == which.min(errors))) +
+     geom_col() + theme_tufte() + guides(fill = FALSE) +
+     scale_x_continuous(breaks = 1:9) +
+     coord_cartesian(ylim = range(errors))
>
```



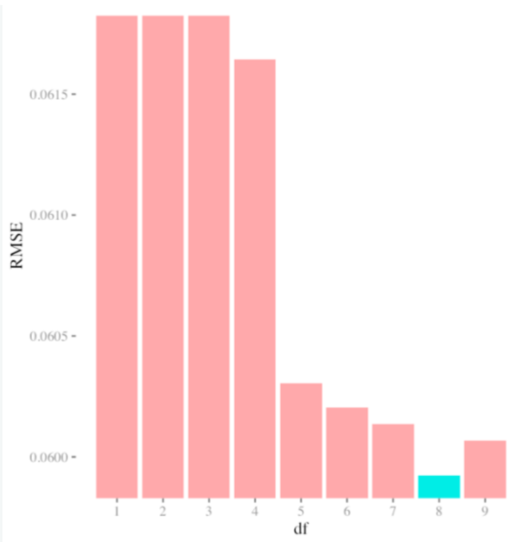
(d) (5 points) Use the bs() function to fit a regression spline to predict nox using dis. Report the output for the fit using four degrees of freedom. How did you choose the knots? Plot the resulting fit.

```
|poly(dis, i)8 |  -0.105|     0.060|    -1.762|   0.079|
|poly(dis, i)9 |   0.056|     0.060|     0.929|   0.353|
> require(splines)
Loading required package: splines
> model <- lm(nox ~ bs(dis, df = 4), data = Boston)
>
> kable(tidy(model), digits = 3)


|term            | estimate| std.error| statistic| p.value|
|:---------------|--------:|---------:|---------:|-------:|
|(Intercept)     |    0.734|     0.015|    50.306|   0.000|
|bs(dis, df = 4)1 |  -0.058|     0.022|    -2.658|   0.008|
|bs(dis, df = 4)2 |  -0.464|     0.024|   -19.596|   0.000|
|bs(dis, df = 4)3 |  -0.200|     0.043|    -4.634|   0.000|
|bs(dis, df = 4)4 |  -0.389|     0.046|    -8.544|   0.000|
> Boston %>%
+     mutate(pred = predict(model)) %>%
+     ggplot() +
+     geom_point(aes(dis, nox, col = '1')) +
+     geom_line(aes(dis, pred, col = '2'), size = 1.5) +
+     scale_color_manual(name = 'Value Type',
+                        labels = c('Observed', 'Predicted'),
+                        values = c('#56B4E9', '#E69F00')) +
+     theme_tufte(base_size = 13)
> |
```

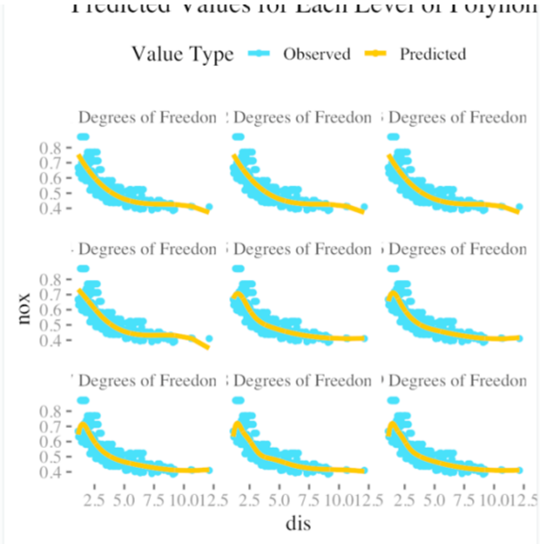


(e) (5 points) Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.

```
+     geom_line(aes(dis, pred, col = '2'), size = 1.5) +
+     scale_color_manual(name = 'Value Type',
+                        labels = c('Observed', 'Predicted'),
+                        values = c('#56B4E9', '#E69F00')) +
+     theme_tufte(base_size = 13)
> errors <- list()
> models <- list()
> pred_df <- data_frame(V1 = 1:506)
> for (i in 1:9) {
+     models[[i]] <- lm(nox ~ bs(dis, df = i), data = Boston)
+     preds <- predict(models[[i]])
+     pred_df[[i]] <- preds
+     errors[[i]] <- sqrt(mean((Boston$nox - preds)^2))
+ }
Warning messages:
1: In bs(dis, df = i) : 'df' was too small; have used 3
2: In bs(dis, df = i) : 'df' was too small; have used 3
>
> names(pred_df) <- paste(1:9, 'Degrees of Freedom')
> data_frame(RMSE = unlist(errors)) %>%
+     mutate(df = row_number()) %>%
+     ggplot(aes(df, RMSE, fill = df == which.min(errors))) +
+     geom_col() + guides(fill = FALSE) + theme_tufte() +
+     scale_x_continuous(breaks = 1:9) +
+     coord_cartesian(ylim = range(errors))
>
```



```
+     pred_df[[i]] <- preds
+     errors[[i]] <- sqrt(mean((Boston$nox - preds)^2))
+ }
Warning messages:
1: In bs(dis, df = i) : 'df' was too small; have used 3
2: In bs(dis, df = i) : 'df' was too small; have used 3
>
> names(pred_df) <- paste(1:9, 'Degrees of Freedom')
> data_frame(RMSE = unlist(errors)) %>%
+     mutate(df = row_number()) %>%
+     ggplot(aes(df, RMSE, fill = df == which.min(errors))) +
+     geom_col() + guides(fill = FALSE) + theme_tufte() +
+     scale_x_continuous(breaks = 1:9) +
+     coord_cartesian(ylim = range(errors))
> Boston %>%
+     cbind(pred_df) %>%
+     gather(df, prediction, -(1:14)) %>%
+     mutate(df = factor(df, levels = unique(as.character(df)))) %>%
+     ggplot() + ggtitle('Predicted Values for Each Level of Polynomial')
 +
+     geom_point(aes(dis, nox, col = '1')) +
+     geom_line(aes(dis, prediction, col = '2'), size = 1.5) +
+     scale_color_manual(name = 'Value Type',
+                        labels = c('Observed', 'Predicted'),
+                        values = c('#56B4E9', '#E69F00')) +
+     facet_wrap(~ df, nrow = 3)
>
```



(f) (5 points) 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.

```
+     geom_line(aes(dis, prediction, col = '2'), size = 1.5) +
+     scale_color_manual(name = 'Value Type',
+                        labels = c('Observed', 'Predicted'),
+                        values = c('#56B4E9', '#E69F00')) +
+     facet_wrap(~ df, nrow = 3)
> folds <- sample(1:10, size = 506, replace = TRUE)
> errors <- matrix(NA, 10, 9)
> models <- list()
> for (k in 1:10) {
+     for (i in 1:9) {
+         models[[i]] <- lm(nox ~ bs(nox, df = i), data = Boston[folds !=
 k,])
+         pred <- predict(models[[i]], Boston[folds == k,])
+         errors[k, i] <- sqrt(mean((Boston$nox[folds == k] - pred)^2))
+     }
+ }
There were 29 warnings (use warnings() to see them)
>
> errors <- apply(errors, 2, mean)
>
> data_frame(RMSE = errors) %>%
+     mutate(df = row_number()) %>%
+     ggplot(aes(df, RMSE, fill = df == which.min(errors))) +
+     geom_col() + theme_tufte() + guides(fill = FALSE) +
+     scale_x_continuous(breaks = 1:9) +
+     coord_cartesian(ylim = range(errors))
>
```