BIOSTAT 274 Spring 2021 Homework 2

Due by 11:59 PM, 05/14/2021

Zian ZHUANG

Remark. For **Computational Part**, please complete your answer in the **RMarkdown** file and summit both the generated PDF and RMD files.

Computational Part

Q1.

([ISL] 4.11, *25 pt*) In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the `Auto` data set. Write a data analysis report addressing the following problems.

```
data(Auto)
#help("Auto")
```

(a)

Create a binary variable, `mpg01`, that contains a 1 if `mpg` contains a value above its median, and a 0 if `mpg` contains a value below its median.

Answer:

```
df <- Auto %>%
  mutate(mpg01=ifelse(mpg > median(mpg), 1, 0) %>% as.factor)
```

(b)

Explore the data graphically in order to investigate the association between `mgp01` and the other features. Which of the other features seem most likely to be useful in predicting `mpg01`? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

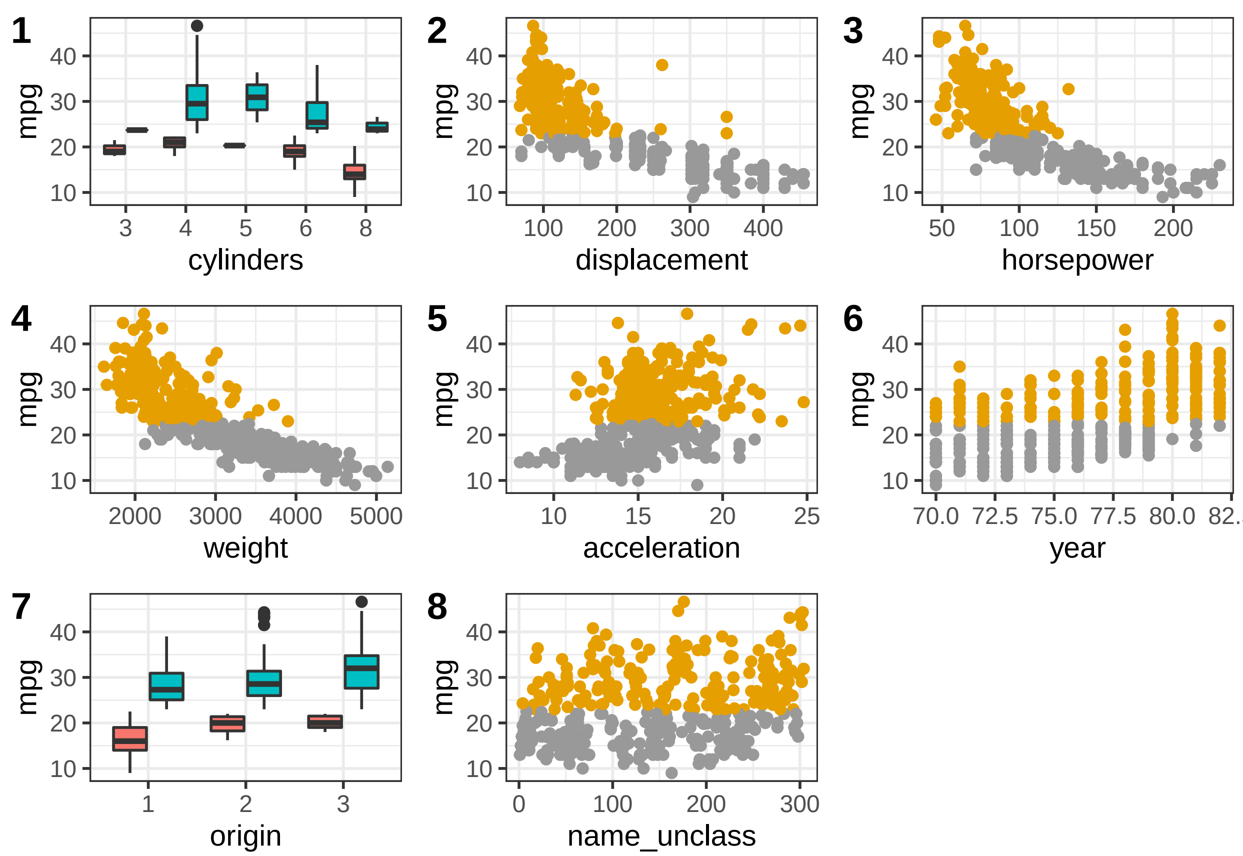
Answer:

```
df %>%
  mutate(name_unclass = unclass(name)) %>%
  mutate_at(vars(names(df)[c(2,8)]),as.factor)-> df_p
vars_need <- dput(names(df_p))[-c(1,9,10)]
```

```
## c("mpg", "cylinders", "displacement", "horsepower", "weight",
## "acceleration", "year", "origin", "name", "mpg01", "name_unclass"
## )
```

```
for(i in 1:8){
  if(i %in% c(1,7)){
    ggplot(df_p, aes_string(x=vars_need[i], y="mpg", fill="mpg01")) +
      geom_boxplot() +
      theme_bw() +
      theme(legend.position = "none") -> temp
  }else{
    ggplot(df_p, aes_string(vars_need[i], "mpg", group="mpg01", color="mpg01")) +
      geom_point() +
      xlab(vars_need[i]) +
      scale_color_manual(values = c('#999999','#E69F00')) +
      #geom_smooth(method=lm, se=T, fullrange=T)+
      theme_bw()+
      theme(legend.position = "none")-> temp
  }
  temp_name <- paste0("fig_",i)
  assign(temp_name, temp)
}

ggarrange(plotlist=mget(paste0("fig_",c(1:8))),
          nrow = 3, ncol = 3, labels = 1:8)
```



According to the scatter plot and boxplot, we found that variable `displacement`, `horsepower`, `weight`, `acceleration` and `year` are most likely to be useful in predicting `mpg01`.

(c)

Split the data into a training set and a test set with ratio 2:1.

Answer:

```
set.seed(1996)
trainid <- sample(1:nrow(df), nrow(df)*2/3 %>% round, replace=F)
train <- df[trainid,]
test <- df[-trainid,]
```

(d)

Perform LDA on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in (b). What is the test error of the model obtained?

Answer:

```
ldafit <- lda(mpg01 ~ displacement+horsepower+weight+acceleration+year,
               data=train)
ldafit_pred <- predict(ldafit, test)$class
table(ldafit_pred, test$mpg01)
```

```
##
## ldafit_pred  0  1
##           0 58  6
##           1  7 60
```

```
# test error
mean(ldafit_pred != test$mpg01)
```

```
## [1] 0.09923664
```

test error rate: 9.92%

(e)

Perform QDA on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in (b). What is the test error of the model obtained?

Answer:

```
qdafit <- qda(mpg01 ~ displacement+horsepower+weight+acceleration+year,
               data=train)
qdafit_pred <- predict(qdafit, test)$class
table(qdafit_pred, test$mpg01)
```

```
##
## qdafit_pred  0  1
##           0 58  7
##           1  7 59
```

```
# test error
mean(qdafit_pred != test$mpg01)
```

```
## [1] 0.1068702
```

test error rate: 10.69%

(f)

Perform logistic regression on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in (b). What is the test error of the model obtained?

Answer:

```
logitfit <- glm(mpg01 ~ displacement+horsepower+weight+acceleration+year,
                data=train, family=binomial)
logitfit_prob <- predict(logitfit, test, type="response")
logitfit_pred <- ifelse(logitfit_prob > 0.5, 1, 0)
table(logitfit_pred, test$mpg01)
```

```
##
## logitfit_pred  0  1
##             0 58  7
##             1  7 59
```

```
# error rate
mean(logitfit_pred != test$mpg01)
```

```
## [1] 0.1068702
```

test error rate is 10.69%, which turned out to be the same as that of QDA.

Q2.

The `Boston` dataset contains variables `dis` (the weighted mean of distances to five Boston employment centers) and `nox` (nitrogen oxides concentration in parts per 10 million).

```
data("Boston")
#help(Boston)
```

(a)

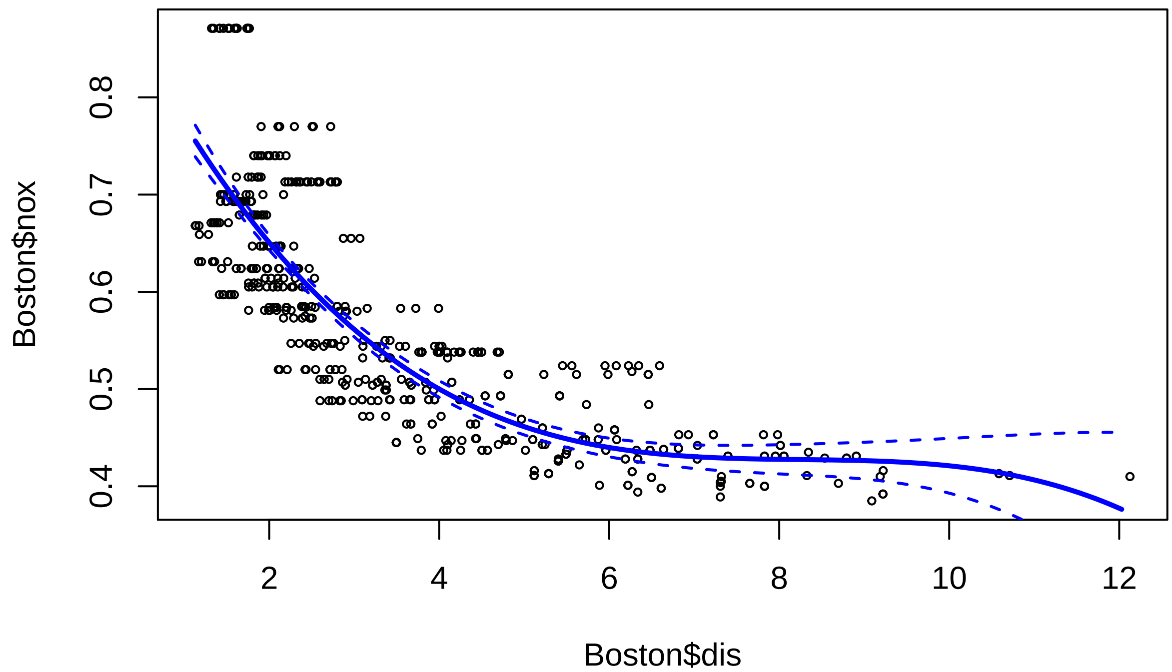
Use the `poly()` function to fit a cubic polynomial regression to predict `nox` using `dis`. Report the regression output, and plot the data and resulting polynomial fits.

Answer:

```
lmfit = lm(nox ~ poly(dis, 3), data = Boston)
summary(lmfit)
```

```
##
## 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
```

```
dislims <- range(Boston$dis)
dis.grid <- seq(dislims[1], dislims[2], 0.1)
preds <- predict(lmfit, newdata=list(dis=dis.grid), se=TRUE)
se95 <- preds$fit + cbind(1.96*preds$se.fit, -1.96*preds$se.fit)
plot(Boston$dis, Boston$nox, xlim=dislims, cex=0.5)
lines(dis.grid, preds$fit, lwd=2.5, col="blue")
matlines(dis.grid, se95, lwd=1.5, col="blue", lty=2)
```



(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.

Answer:

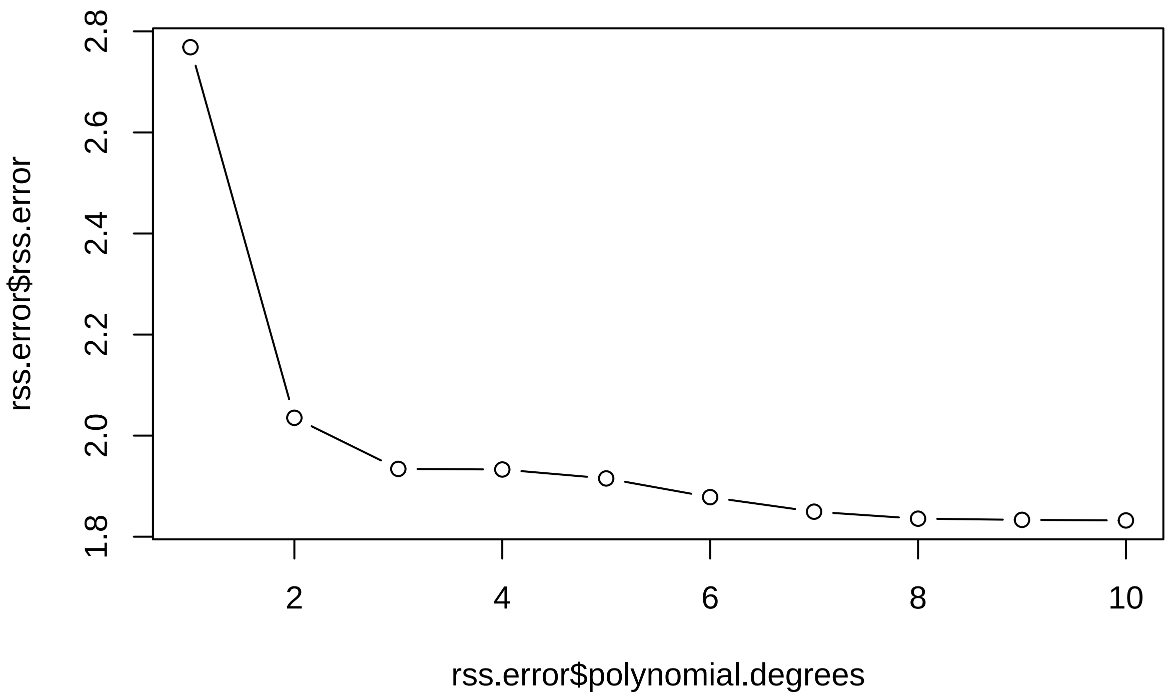
```
rss.error <- NULL
for (i in 1:10) {
  lmfit <- lm(nox ~ poly(dis,i), data=Boston)
  rss.error <- rbind(rss.error,
                     data.frame(rss.error=sum(lmfit$residuals^2),
                                polynomial.degrees=i))
}

# report the associated residual sum of squares
rss.error
```

```
##    rss.error polynomial.degrees
## 1   2.768563                  1
## 2   2.035262                  2
## 3   1.934107                  3
## 4   1.932981                  4
## 5   1.915290                  5
## 6   1.878257                  6
```

```
## 7   1.849484                 7
## 8   1.835630                 8
## 9   1.833331                 9
## 10  1.832171                10
```

```
plot(rss.error$polynomial.degrees, rss.error$rss.error, type="b")
```



We can tell from the plot that rss decreases monotonically when polynomial degrees increase.

(c)

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

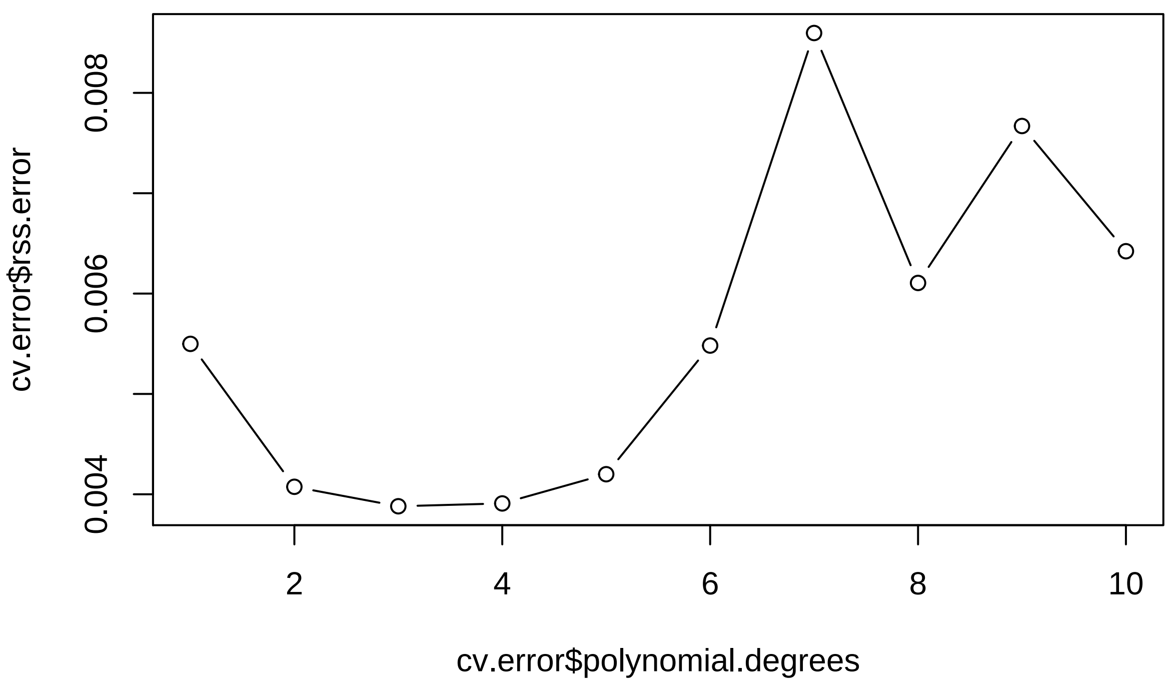
Answer:

```
cv.error <- NULL
for (i in 1:10) {
  set.seed(1996)
  glm.fit <- glm(nox~poly(dis,i), family = gaussian, data=Boston)
  temp <- cv.glm(Boston, glm.fit, K=10)$delta[2]
  cv.error <- rbind(cv.error,
                    data.frame(rss.error=temp, polynomial.degrees=i))
}
cv.error
```

```
##      rss.error polynomial.degrees
```

```
## 1  0.005498802                 1
## 2  0.004074714                 2
## 3  0.003880725                 3
## 4  0.003908910                 4
## 5  0.004200126                 5
## 6  0.005481924                 6
## 7  0.008596404                 7
## 8  0.006105778                 8
## 9  0.007669734                 9
## 10 0.006422154                10
```

```
plot(cv.error$polynomial.degrees, cv.error$rss.error, type="b")
```



According to the plot, we see that the CV error reduces when polynomial degrees increase from 1 to 3 and does not show clear improvement after degree 3 polynomial. Thus, we pick 3 as the best polynomial degree.

(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.

Answer:

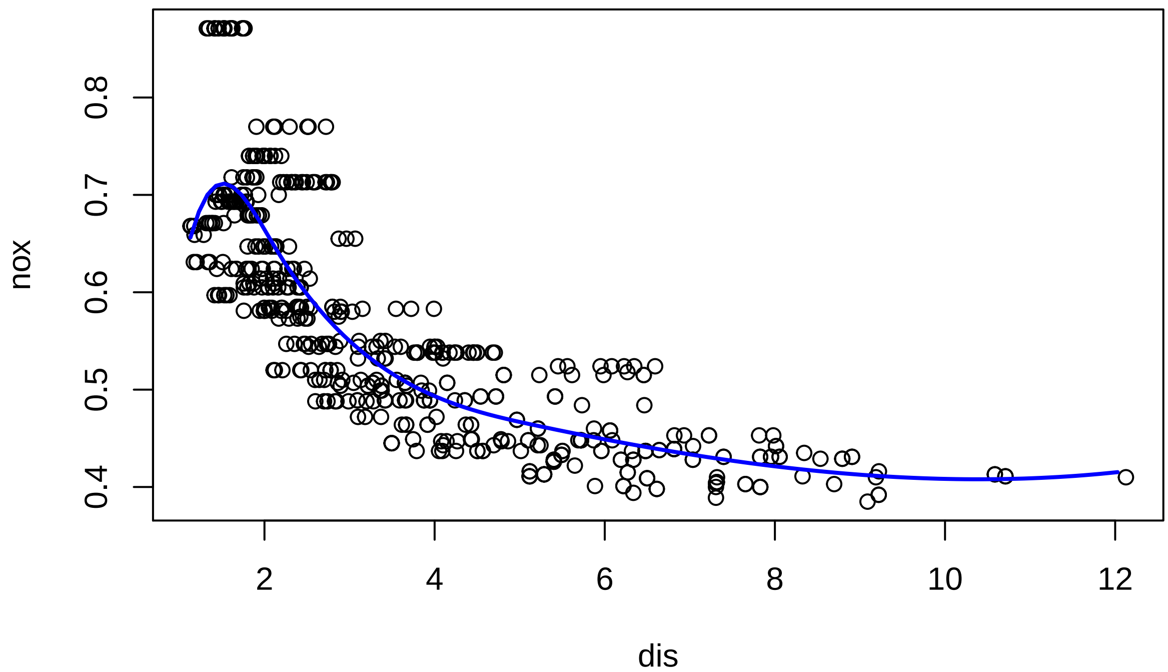
We choose the knots as the 25%, 50% and 75% quantile of the `dis` data.

```r
knots_set <- summary(Boston$dis) %>% as.numeric %>% .[c(2,3,5)]
spfit <- lm(nox ~ bs(dis, df = 4, knots = knots_set), data = Boston)

#Report the model summary
summary(spfit)
```

```
##
## Call:
## lm(formula = nox ~ bs(dis, df = 4, knots = knots_set), data = Boston)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.128538 -0.037813 -0.009987  0.022644  0.195494
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                           0.65622    0.02370  27.689  < 2e-16 ***
## bs(dis, df = 4, knots = knots_set)1  0.10222    0.03516   2.907  0.00381 **
## bs(dis, df = 4, knots = knots_set)2 -0.02963    0.02338  -1.267  0.20571
## bs(dis, df = 4, knots = knots_set)3 -0.15959    0.02791  -5.718 1.86e-08 ***
## bs(dis, df = 4, knots = knots_set)4 -0.22815    0.03324  -6.864 1.99e-11 ***
## bs(dis, df = 4, knots = knots_set)5 -0.26272    0.04930  -5.329 1.50e-07 ***
## bs(dis, df = 4, knots = knots_set)6 -0.24002    0.05434  -4.417 1.23e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06062 on 499 degrees of freedom
## Multiple R-squared:  0.7295, Adjusted R-squared:  0.7263
## F-statistic: 224.3 on 6 and 499 DF,  p-value: < 2.2e-16
```

```r
#Resulting fit
sppred <- predict(spfit, list(dis = dis.grid))
plot(nox ~ dis, data = Boston)
lines(dis.grid, sppred, col = "blue", lwd = 2)
```



The prediction line seems to fit the data well.

(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.

Answer:

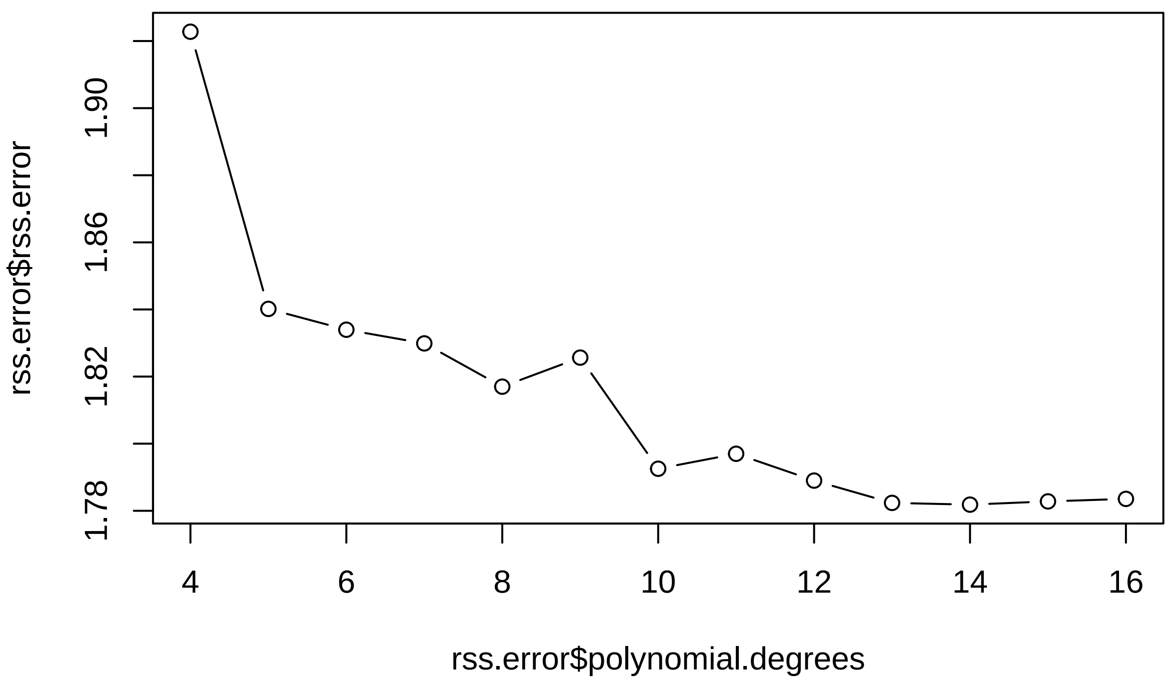
```
rss.error <- NULL
for (i in 4:16) {
  spfit <- lm(nox~bs(dis, df=i), data=Boston)
  rss.error <- rbind(rss.error,
                     data.frame(rss.error=sum(spfit$residuals^2),
                                polynomial.degrees=i))
}

# report the associated residual sum of squares
rss.error
```

```
##    rss.error polynomial.degrees
## 1   1.922775                  4
## 2   1.840173                  5
## 3   1.833966                  6
## 4   1.829884                  7
## 5   1.816995                  8
```

```
## 6   1.825653                 9
## 7   1.792535                10
## 8   1.796992                11
## 9   1.788999                12
## 10  1.782350                13
## 11  1.781838                14
## 12  1.782798                15
## 13  1.783546                16
```

```
plot(rss.error$polynomial.degrees, rss.error$rss.error, type="b")
```



We can tell from the plots that rss error reduces when degrees of freedom increase from 4 to 14 and does not show clear improvement after 14 degrees of freedom.

(f)

Perform cross-validation to select the best degrees of freedom for a regression spline on this data. Describe your results.

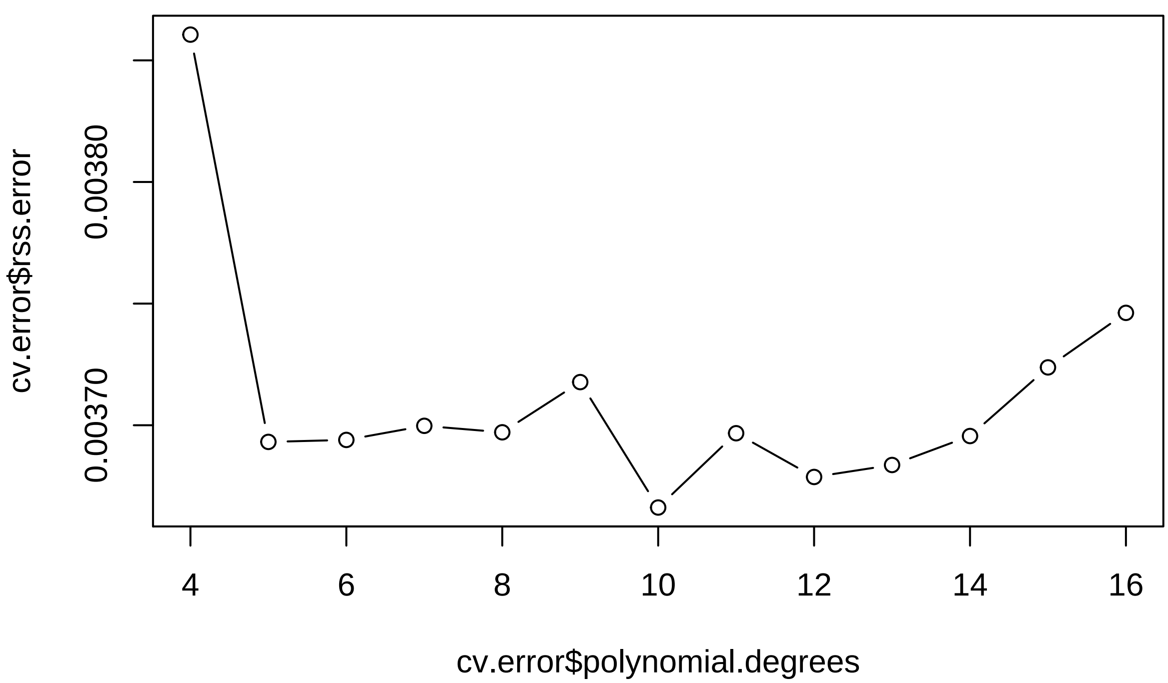
Answer:

```
cv.error <- NULL
for (i in 4:16) {
  set.seed(19969)
  glm.fit <- glm(nox ~ bs(dis, df = i), family = gaussian, data=Boston)
```

```
  temp <- cv.glm(Boston, glm.fit, K=10)$delta[2]
  cv.error <- rbind(cv.error,
                    data.frame(rss.error=temp, polynomial.degrees=i))
}
cv.error
```

```
##      rss.error polynomial.degrees
## 1  0.003860574                  4
## 2  0.003693149                  5
## 3  0.003693960                  6
## 4  0.003699757                  7
## 5  0.003697093                  8
## 6  0.003717735                  9
## 7  0.003666172                 10
## 8  0.003696715                 11
## 9  0.003678730                 12
## 10 0.003683654                 13
## 11 0.003695563                 14
## 12 0.003723776                 15
## 13 0.003746196                 16
```

```
plot(cv.error$polynomial.degrees, cv.error$rss.error, type="b")
```



After 10-folds cross-validation, we can tell from the plots that cv error reach the minimum value at 10 degrees of freedom and does not show clear improvement after 10 degrees of freedom. Thus, we choose 10 as the best degrees of freedom.