HW8

Jingshi

4/2/2017

7.9 #2

a)

When λ (the smoothing factor) is extremely large and m = 0, $g^{(0)}$ (which equals to g) is forced to converge to 0. Therefore, $\hat{g} = 0$.

b)

When λ (the smoothing factor) is extremely large and m = 1, $g^{(1)}$ (which is the first derivative of g) is forced to converge to 0. Therefore, $\hat{g} = c$ where c stands for a constant. For example, $\hat{g} = 9$.

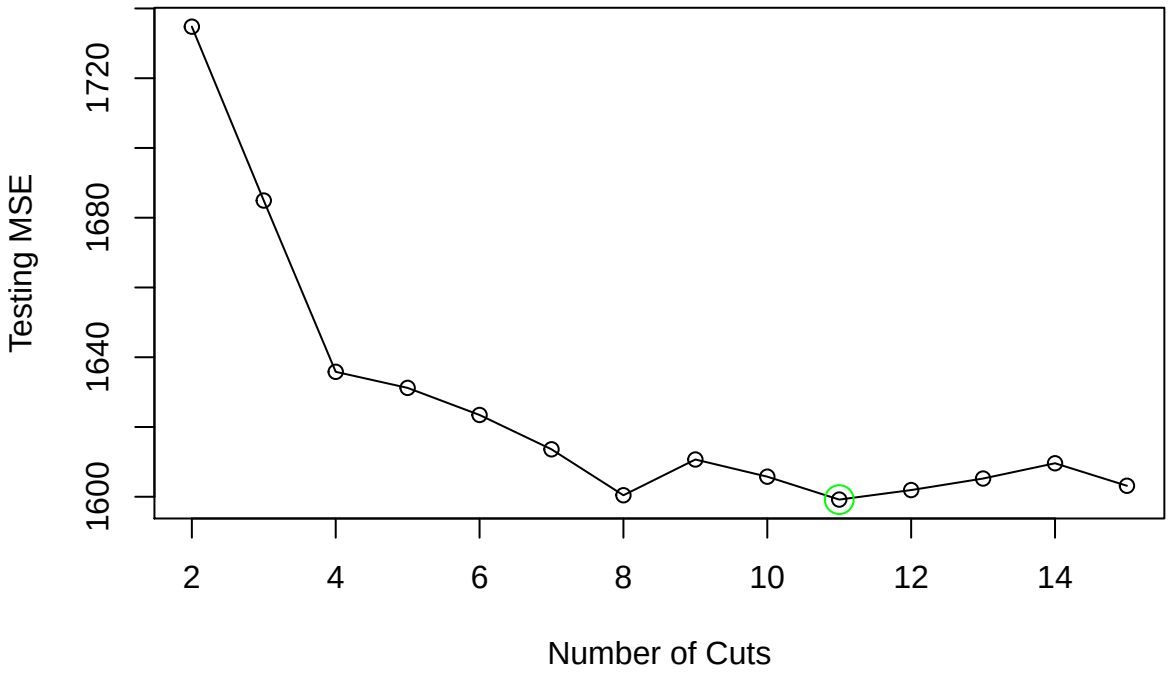
c)

When λ (the smoothing factor) is extremely large and m = 2, $g^{(2)}$ (which is the second derivative of g) is forced to converge to 0. Therefore, $\hat{g} = a \times x + b$ where a and b are both stands for constants. For example, $\hat{g} = 9 \times x + 3$.

7.9 #6

b)

```
library(boot)
library(ISLR)
#head(Wage)

# perform cross-validation to choose the optimal number of cuts
set.seed(6)
# Divide into 2 to 15 cuts
cross.validation <- rep(NA, 15)
for (i in 2:15) {
    Wage$cut <- cut(Wage$age, i)
    fit <- glm(wage ~ cut, data = Wage)
    # 12 fold cross validation
    cross.validation[i] <- cv.glm(Wage, fit, K = 12)$delta[1]
}
plot(2:15, cross.validation[-1], xlab = "Number of Cuts", ylab = "Testing MSE", type = "o")
points(which.min(cross.validation), cross.validation[which.min(cross.validation)], col = "green",
       pch = 1, cex = 2)
```



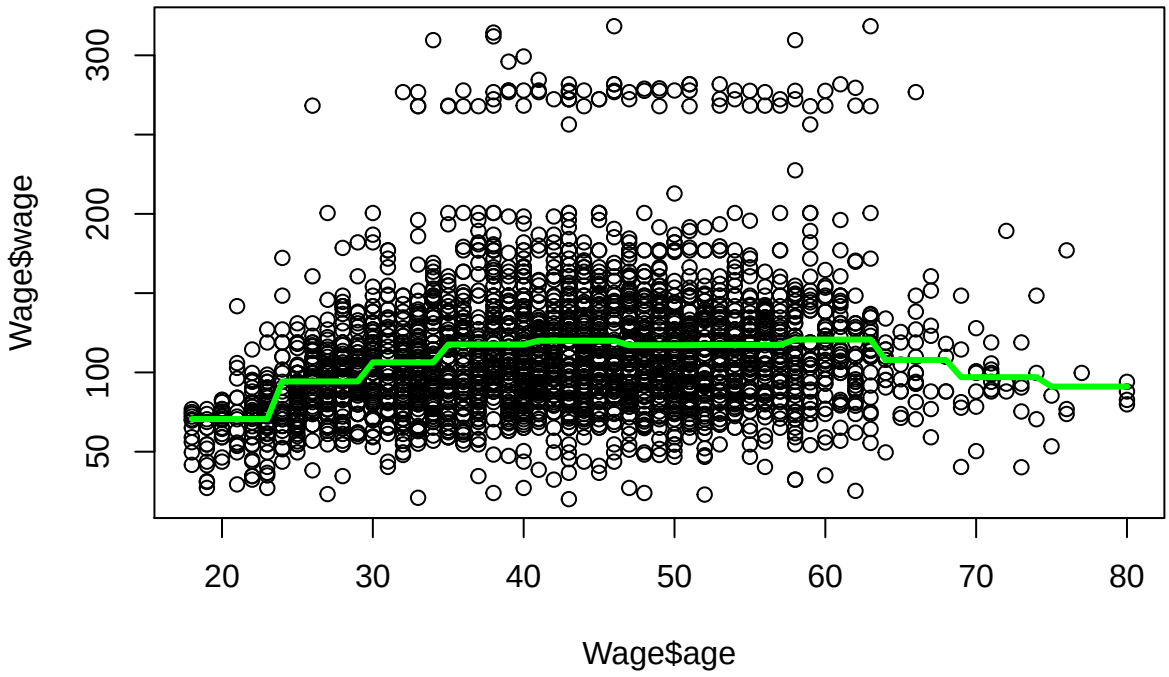
As the above plot shows, when number of cuts is 11, the testing mean squared error is the lowerst among the other cuts. Thus, the optimal number of cuts is 11.

```
# fit a step function
model.step <- glm(wage ~ cut(age, 11), data = Wage)
# take a look at the summary
summary(model.step)
```

```
##
## Call:
## glm(formula = wage ~ cut(age, 11), data = Wage)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -99.89  -24.77   -5.43   15.46  203.22
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                70.580      3.340  21.133  < 2e-16 ***
## cut(age, 11)(23.6,29.3]    23.719      4.048   5.860 5.14e-09 ***
## cut(age, 11)(29.3,34.9]    35.768      3.921   9.122  < 2e-16 ***
## cut(age, 11)(34.9,40.5]    47.001      3.785  12.418  < 2e-16 ***
## cut(age, 11)(40.5,46.2]    49.399      3.752  13.167  < 2e-16 ***
## cut(age, 11)(46.2,51.8]    46.571      3.833  12.150  < 2e-16 ***
## cut(age, 11)(51.8,57.5]    46.781      3.936  11.887  < 2e-16 ***
## cut(age, 11)(57.5,63.1]    50.142      4.279  11.719  < 2e-16 ***
## cut(age, 11)(63.1,68.7]    37.296      6.946   5.369 8.50e-08 ***
## cut(age, 11)(68.7,74.4]    26.414      8.020   3.294    0.001 **
## cut(age, 11)(74.4,80.1]    20.491     13.064   1.569    0.117
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1595.04)
```

```
##
##     Null deviance: 5222086  on 2999  degrees of freedom
## Residual deviance: 4767575  on 2989  degrees of freedom
## AIC: 30651
##
## Number of Fisher Scoring iterations: 2
```

```r
# preperaton for the plot
seq.age <- seq(from = range(Wage$age)[1], to = range(Wage$age)[2])
prediction <- predict(model.step, list(age = seq.age))

# make a plot of the fit obtained.
plot(Wage$wage ~ Wage$age)
lines(seq.age, prediction, col = "green", lwd = 3)
```



7.9 #9

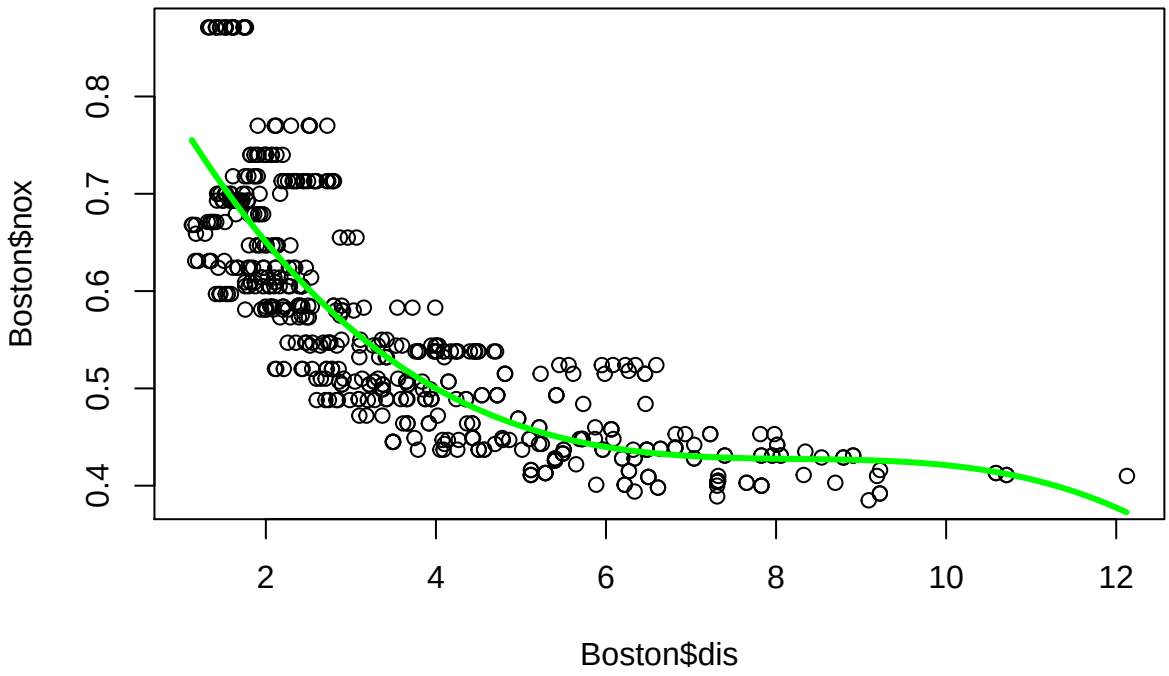
a)

```r
library(MASS)
# use the poly() function to fit a cubic polynomial regression to predict nox using dis.
model.poly <- lm(nox ~ poly(dis, 3), data = Boston)
# report the regression output.
summary(model.poly)
```

```
##
## 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
```

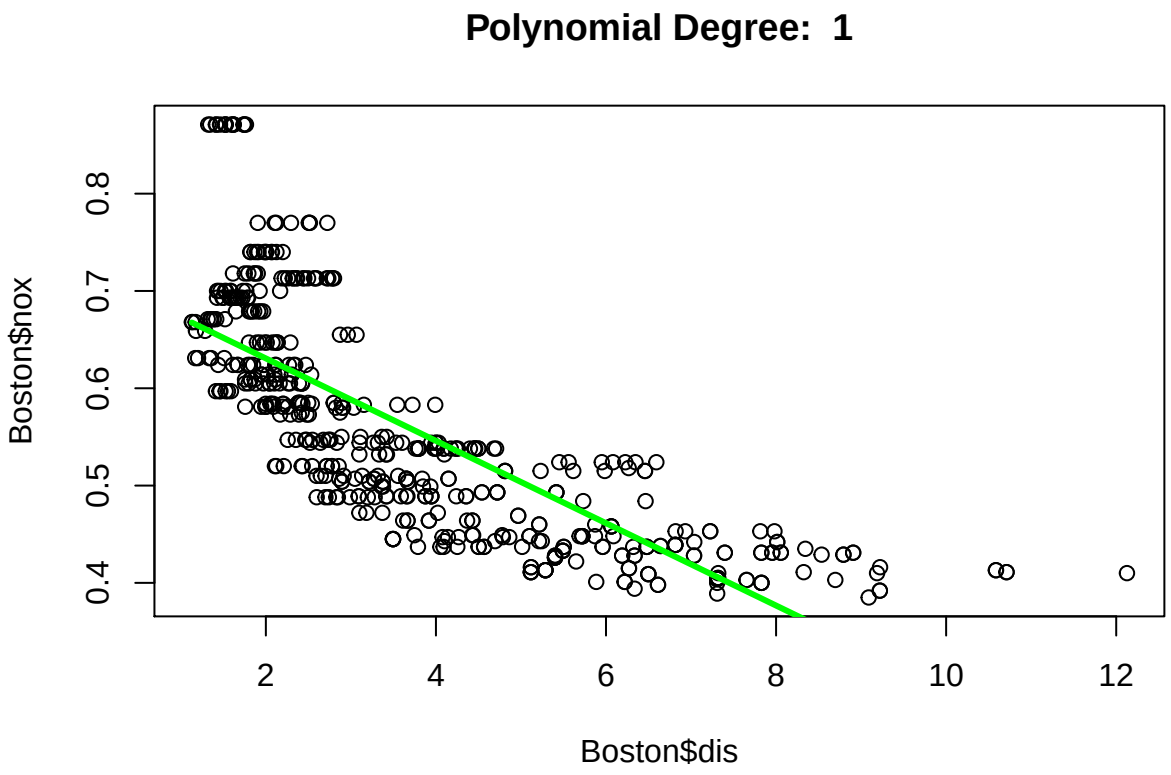
```r
# plot the resulting data.
plot(Boston$nox ~ Boston$dis)
# Plot the polynomial fits.
seq.dis <- seq(from = range(Boston$dis)[1], to = range(Boston$dis)[2], by = 0.01)
prediction <- predict(model.poly, data.frame(dis=seq.dis))
lines(seq.dis, prediction, col = "green", lwd = 3)
```

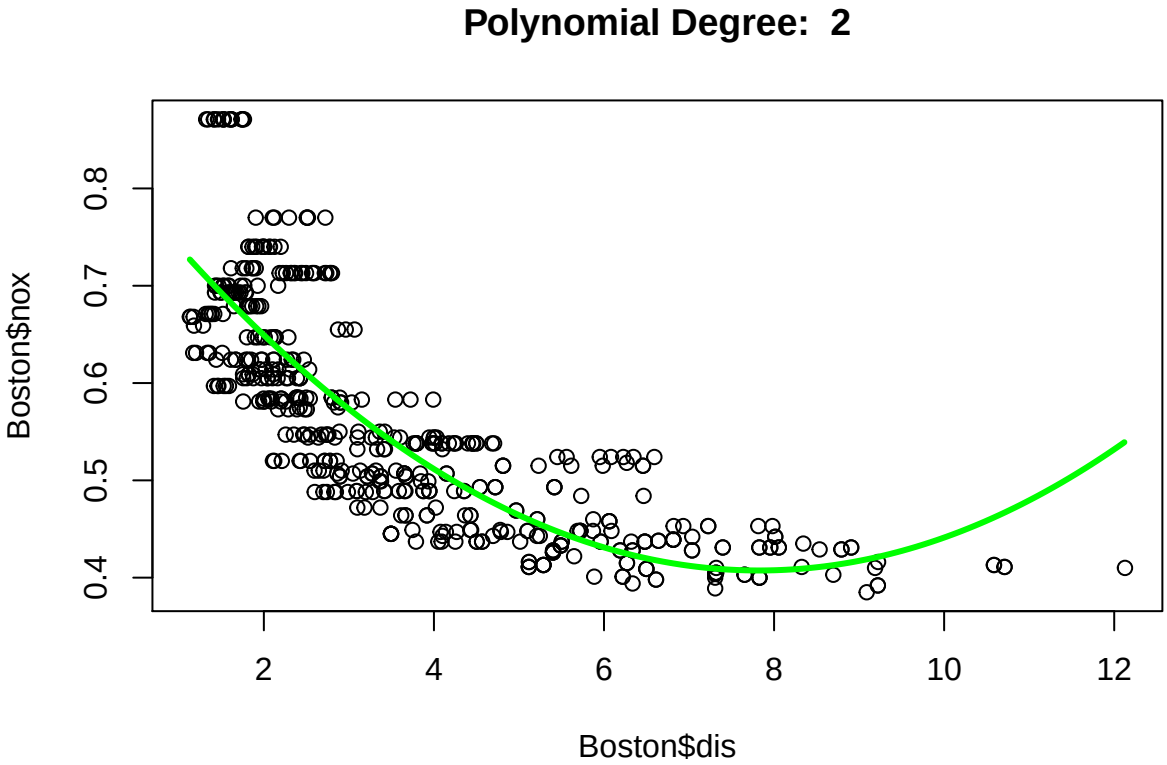


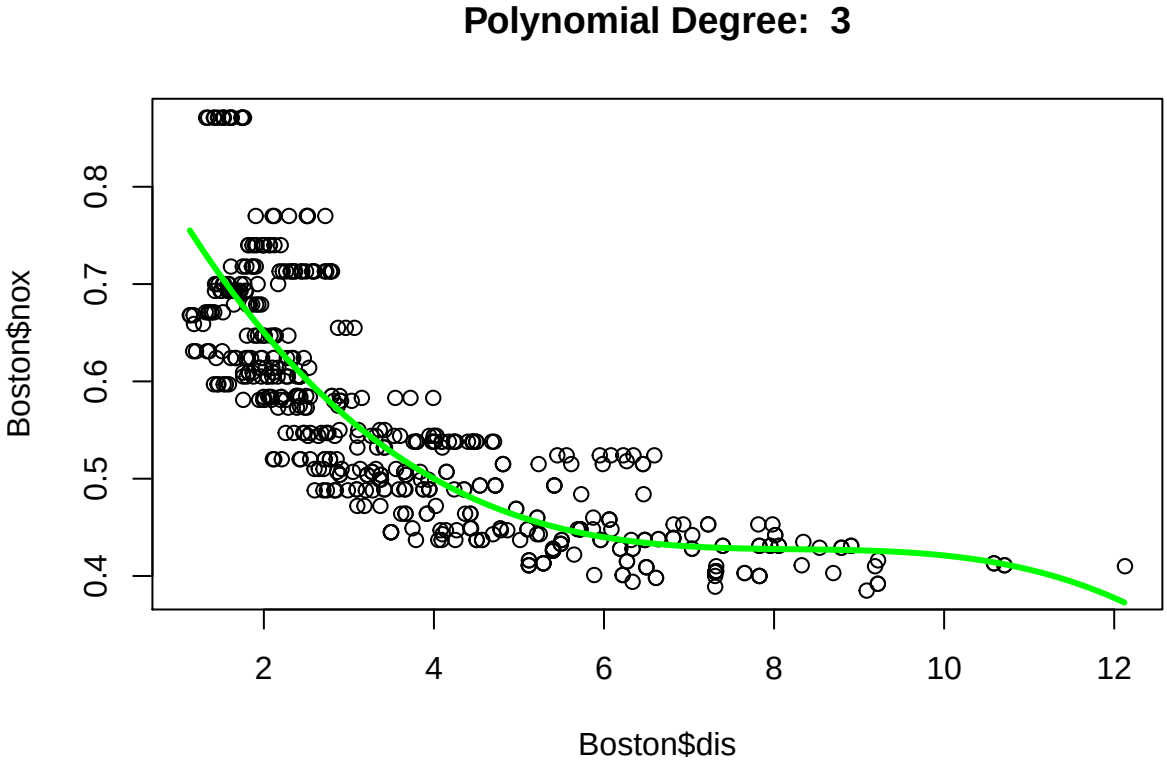
b)

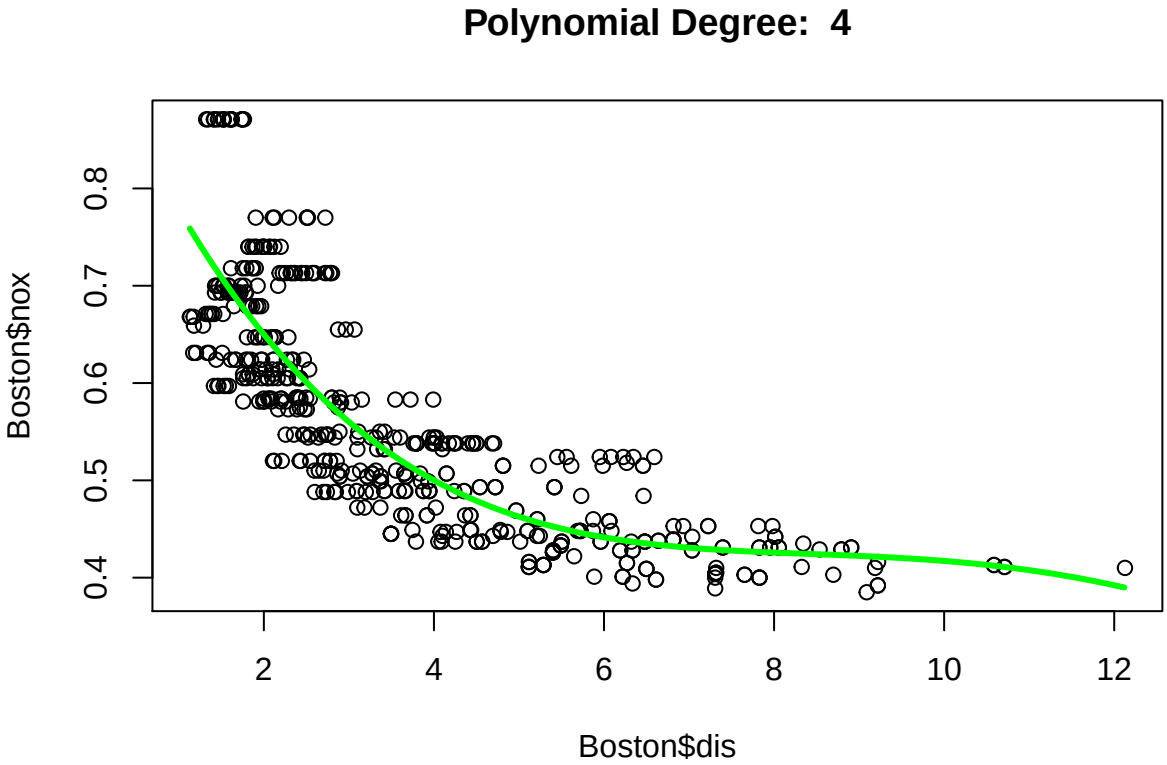
```r
library(data.table)
# create a vector with 10 elements in order to record residual sum of squares
# for each one of the 10 polynomial degrees.
myRss <- rep(NA, 10)
# fit 10 polynomial models and record their associated residual sum of squares.

for (i in 1:10) {
```

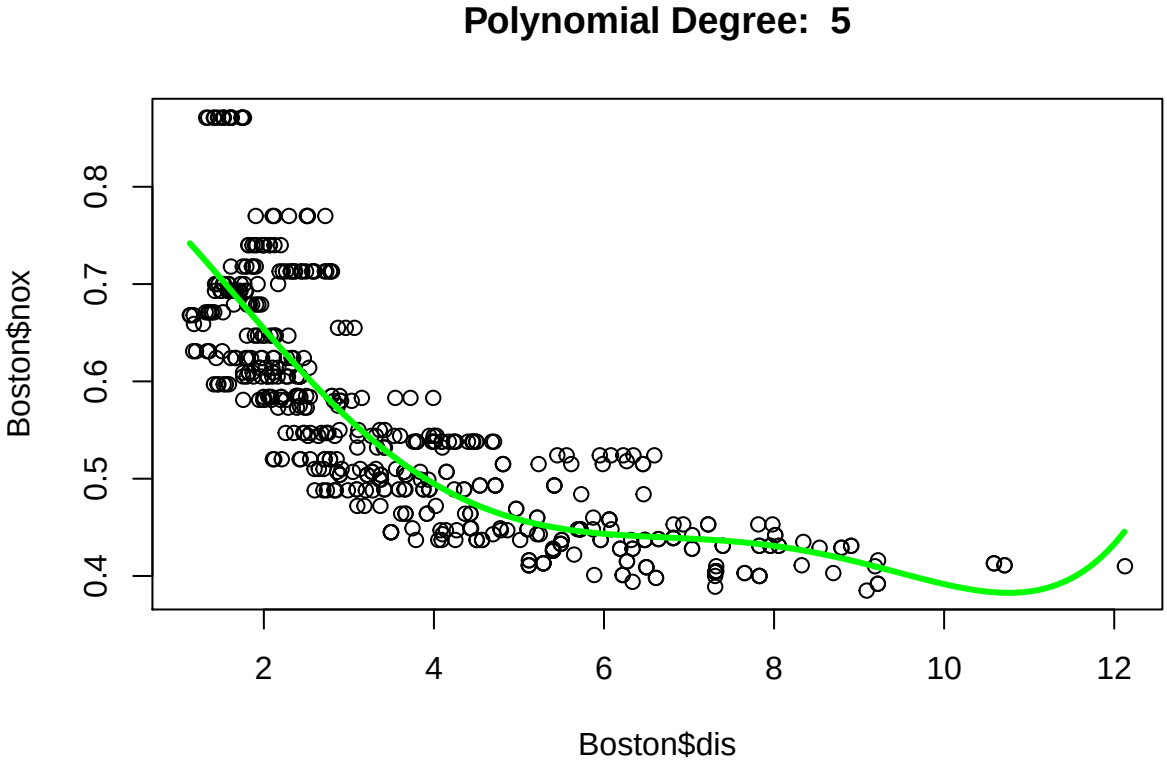
```r
    # fit a polynomial model.
    model.poly <- lm(nox ~ poly(dis, i), data = Boston)
    # plot the polynomial fit
    plot(Boston$nox ~ Boston$dis, main=paste("Polynomial Degree: ", i))
    seq.dis <- seq(from = range(Boston$dis)[1], to = range(Boston$dis)[2], by = 0.01)
    prediction <- predict(model.poly, data.frame(dis=seq.dis))
    lines(seq.dis, prediction, col = "green", lwd = 3)
    # record the associated residual sum of squares.
    myRss[i] <- sum(model.poly$residuals^2)
}
```



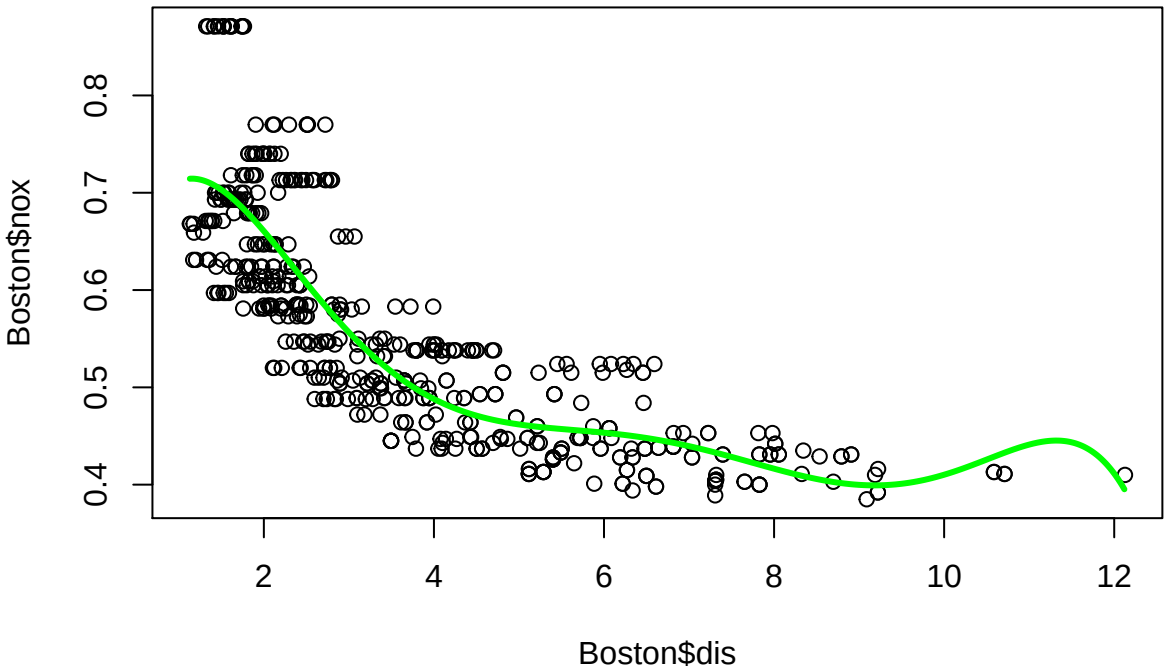

Polynomial Degree: 2
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis


Polynomial Degree: 3
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis

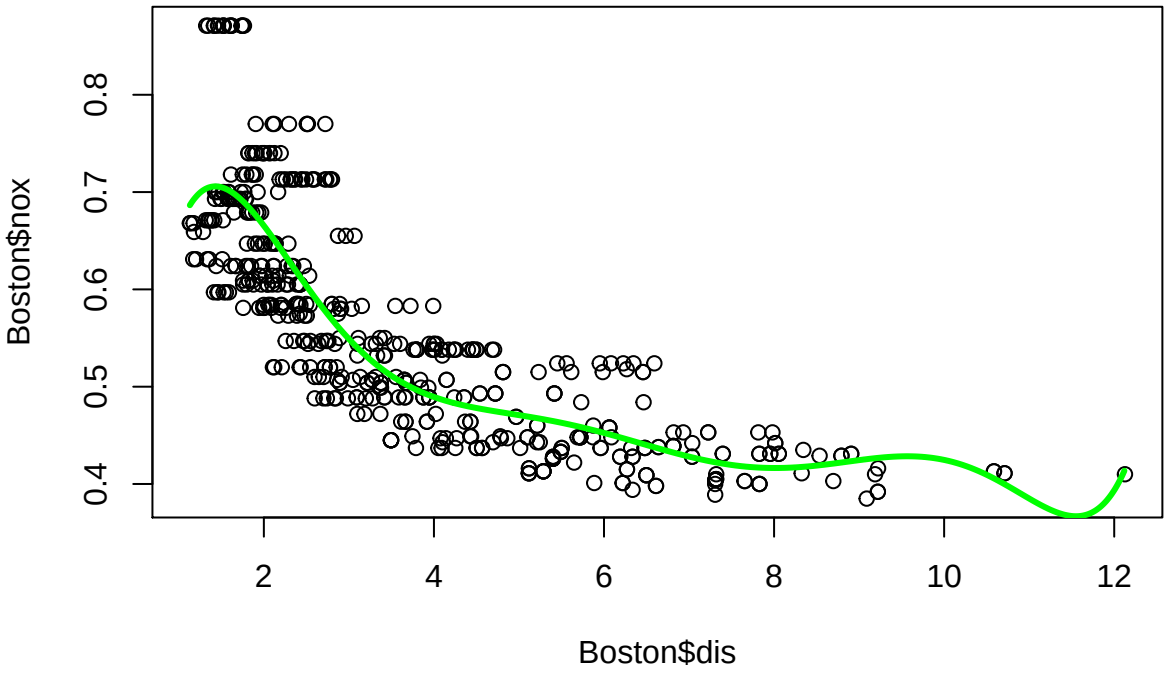

Polynomial Degree: 4
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis

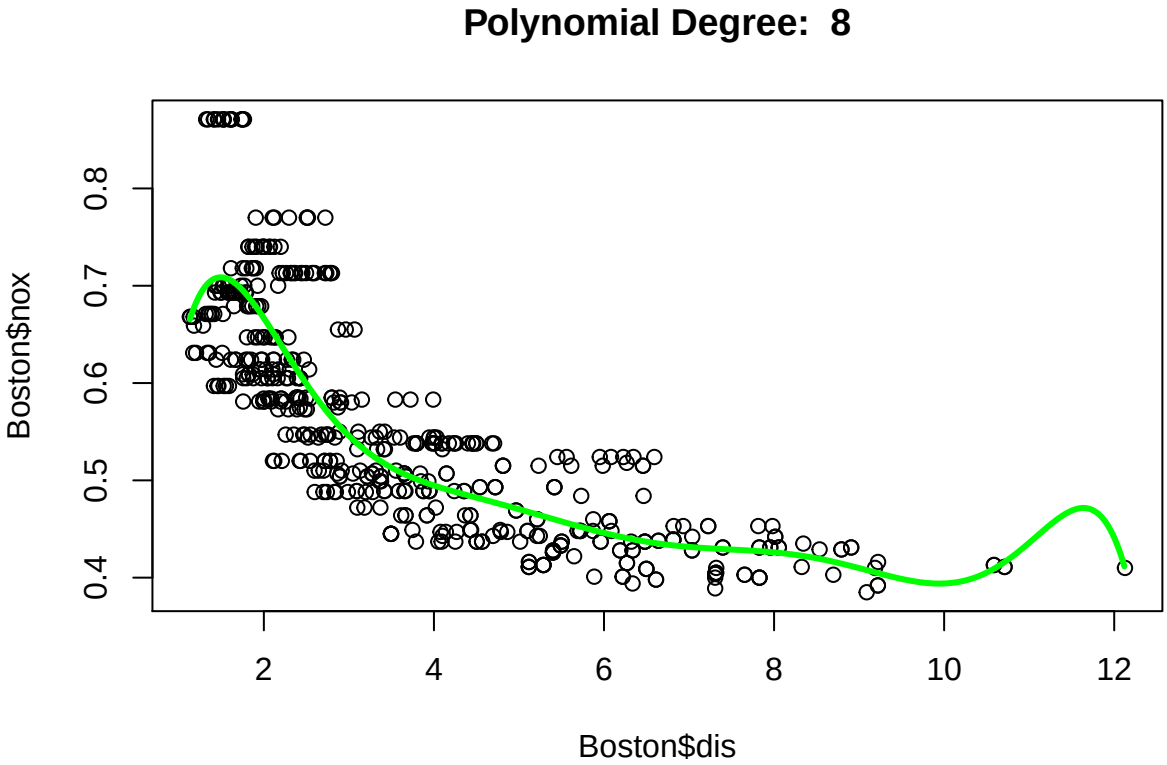

Polynomial Degree: 5
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis

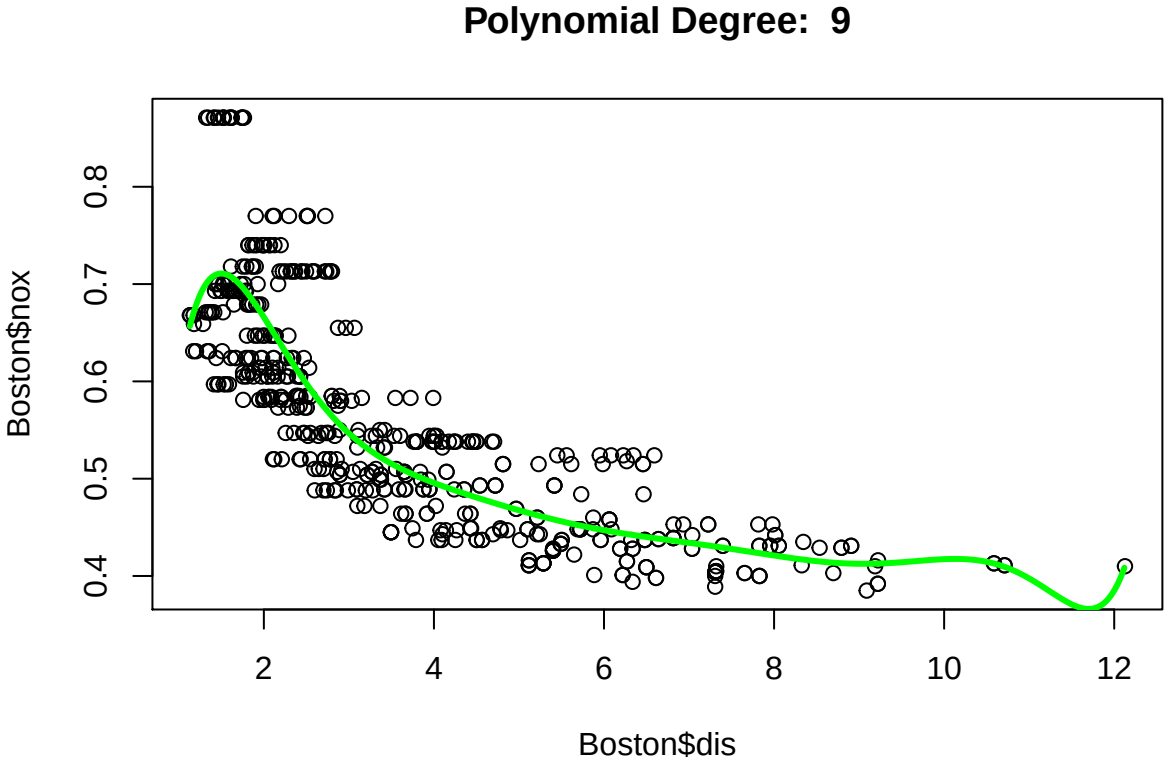
Polynomial Degree: 6

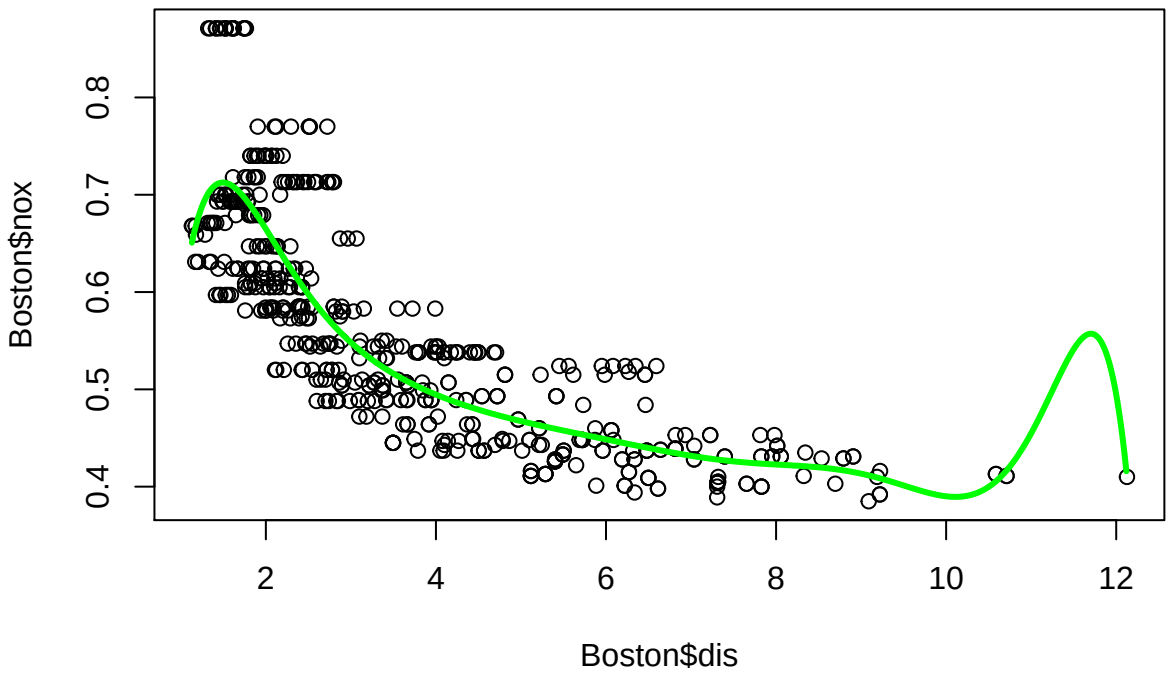


Polynomial Degree: 7

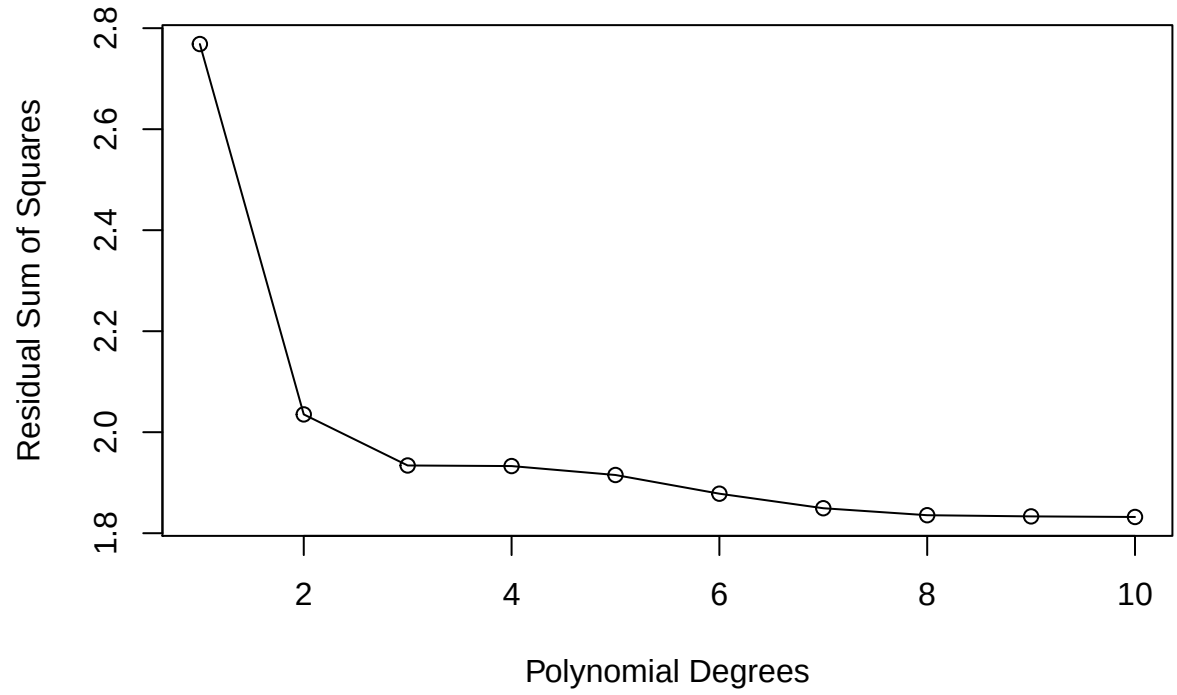



Polynomial Degree: 8
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis


Polynomial Degree: 9
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis




```
# report the RSS's by ploting them out.
plot(1:10, myRss, xlab = "Polynomial Degrees", ylab = "Residual Sum of Squares", type = "o")
```



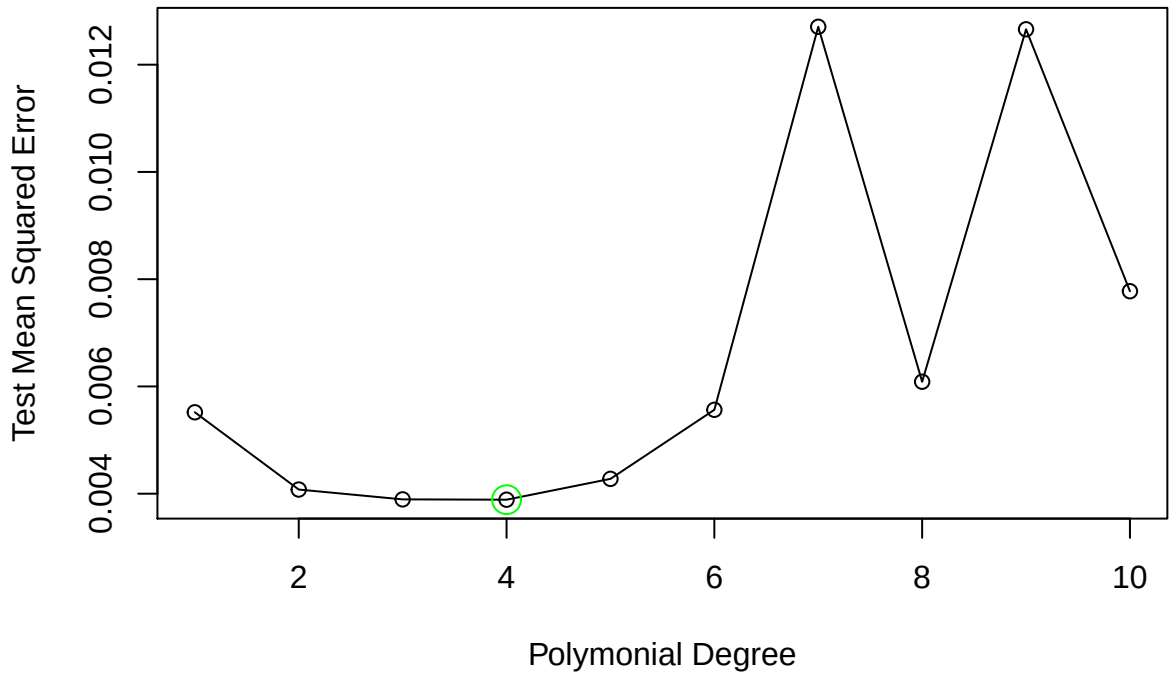

```
RSS<-myRss
# create a table
myTable<-data.table(RSS)
# take a look at the table
myTable[, Polynomial_Degree := .GRP, by = list(c(1,2,3,4,5,6,7,8,9,10))]
```

```
##          RSS Polynomial_Degree
##  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
```

Acording to the above table and the plots of RSS show, as the polynomial degree increase, RSS decrease. When polymomial degree is 10, the RSS is the smallest.

c)

```
set.seed(6)
library(boot)
# perform cross-validation for polymomial degrees from 1 to 10.
testMSE <- rep(NA, 10)
for (i in 1:10) {
    model.poly <- glm(nox ~ poly(dis, i), data = Boston)
    testMSE[i] <- cv.glm(Boston, model.poly, K = 12)$delta[1]
}
plot(1:10, testMSE, xlab = "Polymonial Degree", ylab = "Test Mean Squared Error", type = "o")
points(which.min(testMSE), testMSE[which.min(testMSE)], col = "green",
       pch = 1, cex = 2)
```



As you can see from the above plot, the test mean squared error is minimized when polymomial degree is 4. Therefore, the optimal degree for the polynomial is 4.

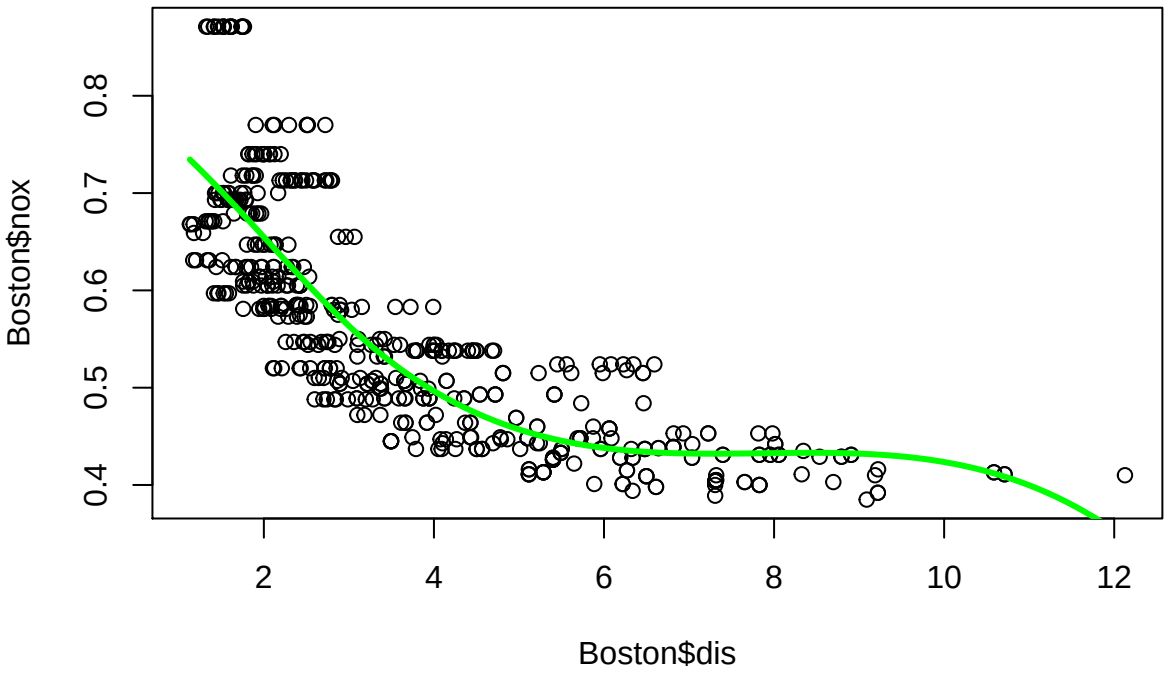
7.9 #9

d)

```
library(splines)
# fit a regression spline
model.lm <- lm(nox ~ bs(dis, df=4), data = Boston)
#bs(Boston$dis, df=4)
# Report the output for the fit
summary(model.lm)
```

```
##
## Call:
## lm(formula = nox ~ bs(dis, df = 4), data = Boston)
##
## 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
##
## Residual standard error: 0.06195 on 501 degrees of freedom
## Multiple R-squared:  0.7164, Adjusted R-squared:  0.7142
## F-statistic: 316.5 on 4 and 501 DF,  p-value: < 2.2e-16
```

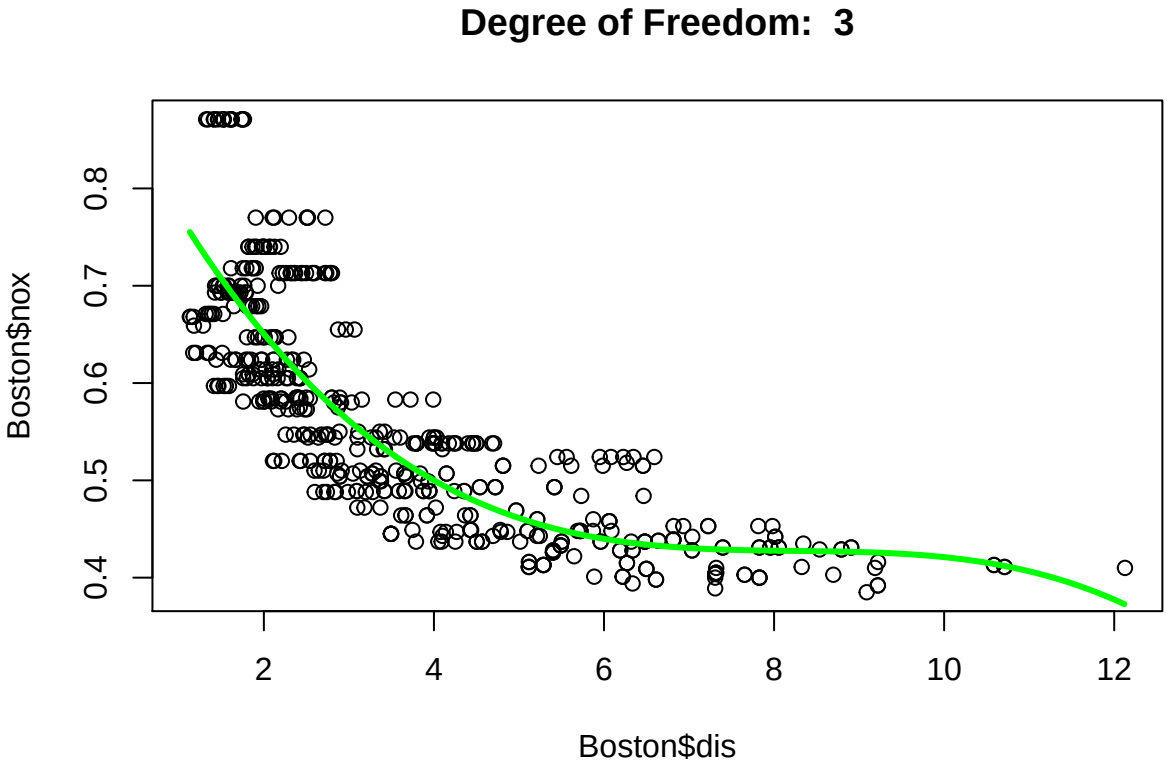
```
# preparation for the below plot
prediction <- predict(model.lm, data.frame(dis = seq.dis))
# Plot the resulting fit.
plot(Boston$nox ~ Boston$dis)
lines(seq.dis, prediction, col = "green", lwd = 3)
```

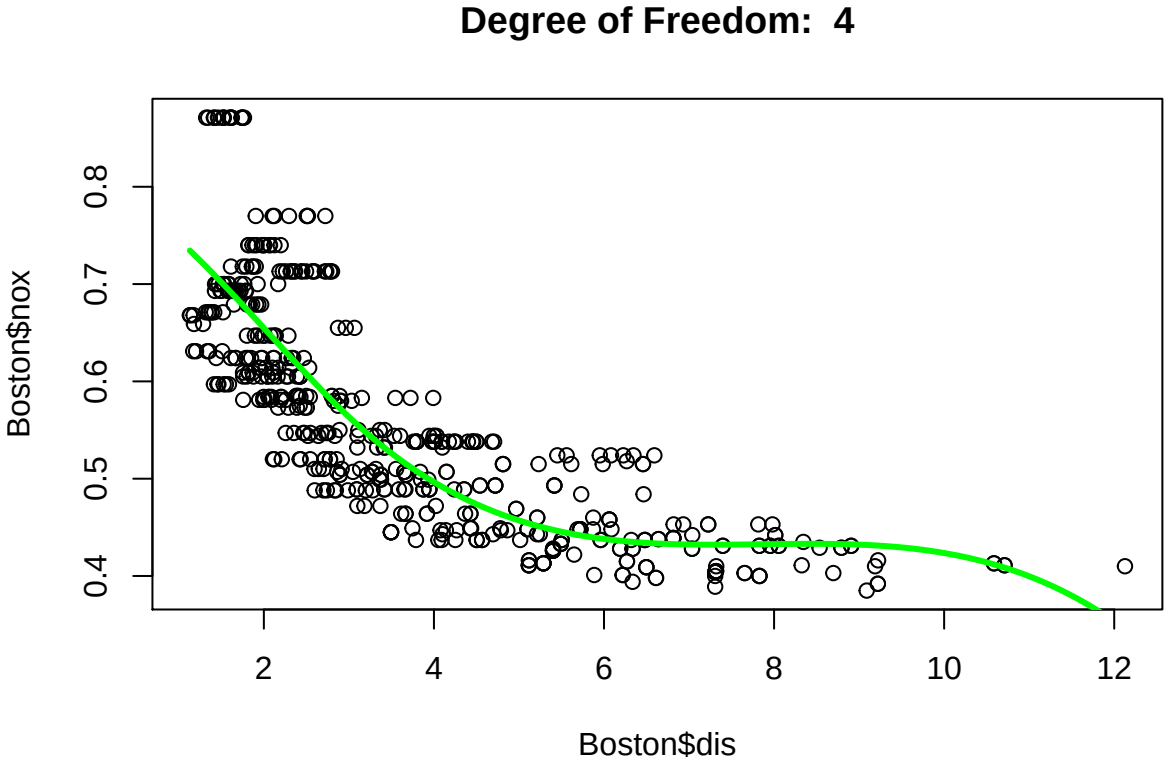


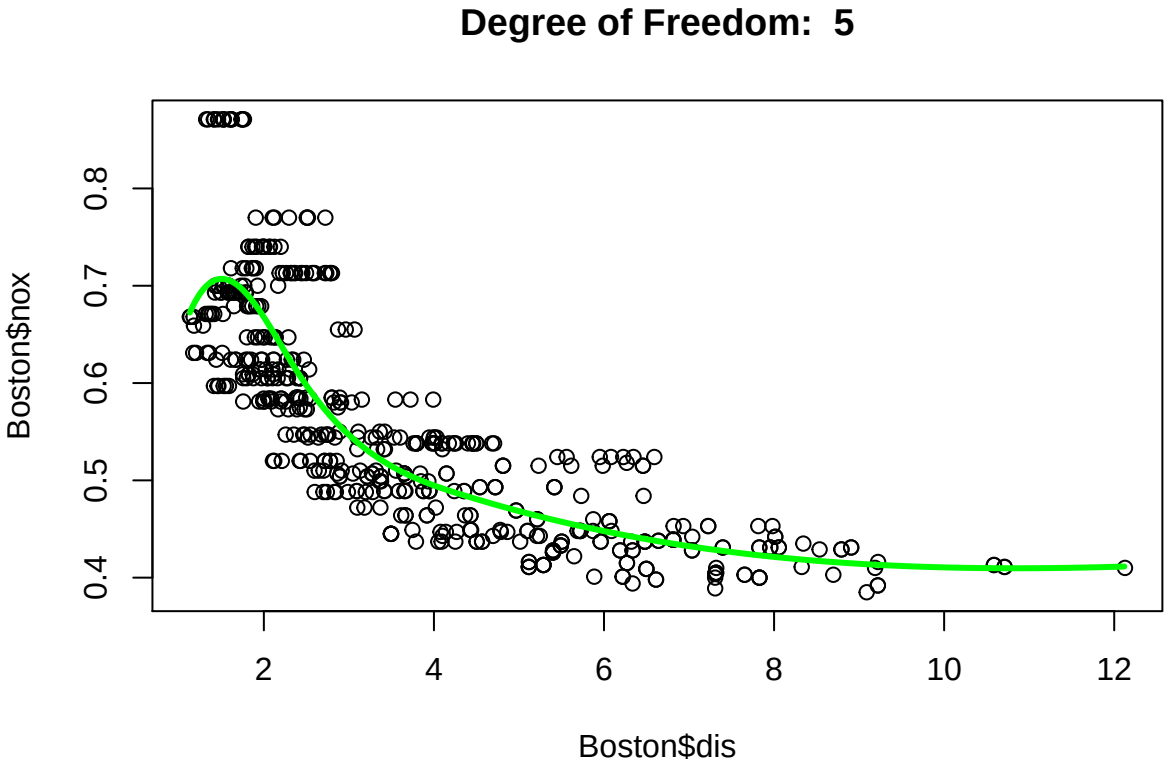
I choose the knots based on df = 4. Thus the knots are 1.296, 3.20745 and 12.1265 (this is obtained from bs(Boston$dis, df=4) and the output is too long to show here), each term in the fitted model are significant since their p-values are much smaller than alpha=0.05. Moreover, the model's adjusted R-squred is high at 0.71.

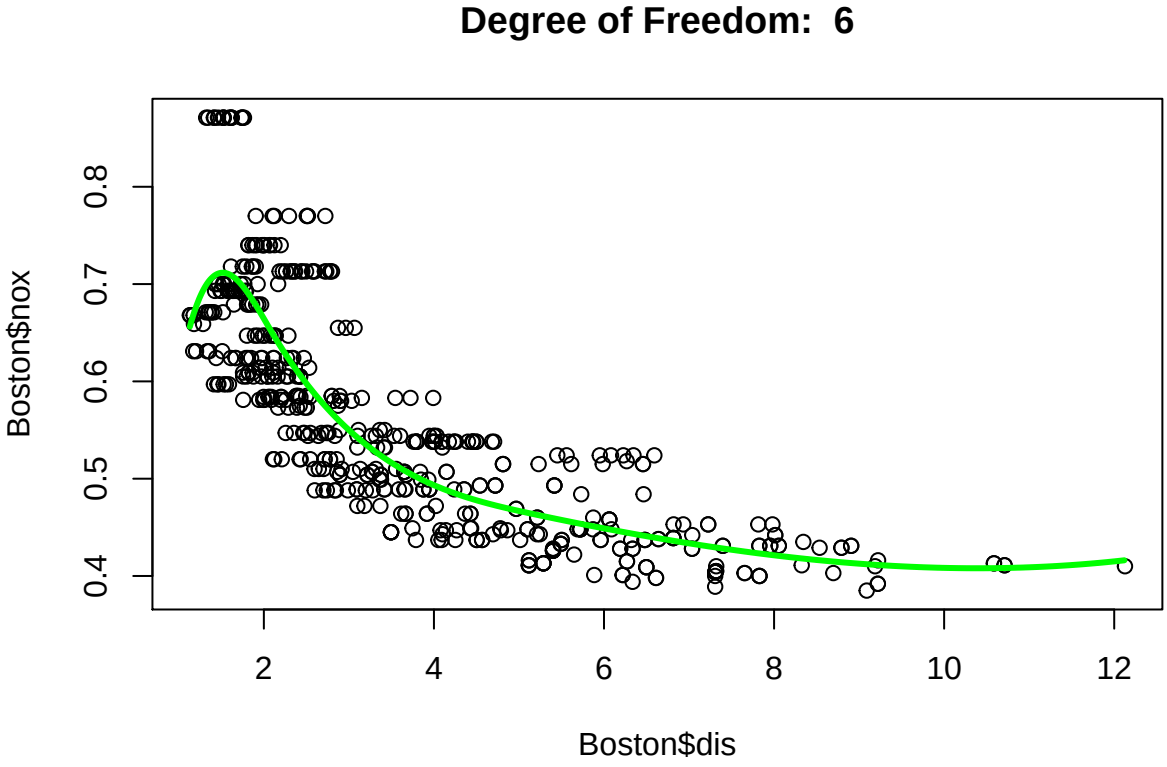
e)

```
myRss <- rep(NA, 20)
for (i in 3:20) {
    model.lm <- lm(nox ~ bs(dis, df = i), data = Boston)
    # plot the fit
    plot(Boston$nox ~ Boston$dis, main=paste("Degree of Freedom: ", i))
    seq.dis <- seq(from = range(Boston$dis)[1], to = range(Boston$dis)[2], by = 0.01)
    prediction <- predict(model.lm, data.frame(dis=seq.dis))
    lines(seq.dis, prediction, col = "green", lwd = 3)
    myRss[i] <- sum(model.lm$residuals^2)
}
```

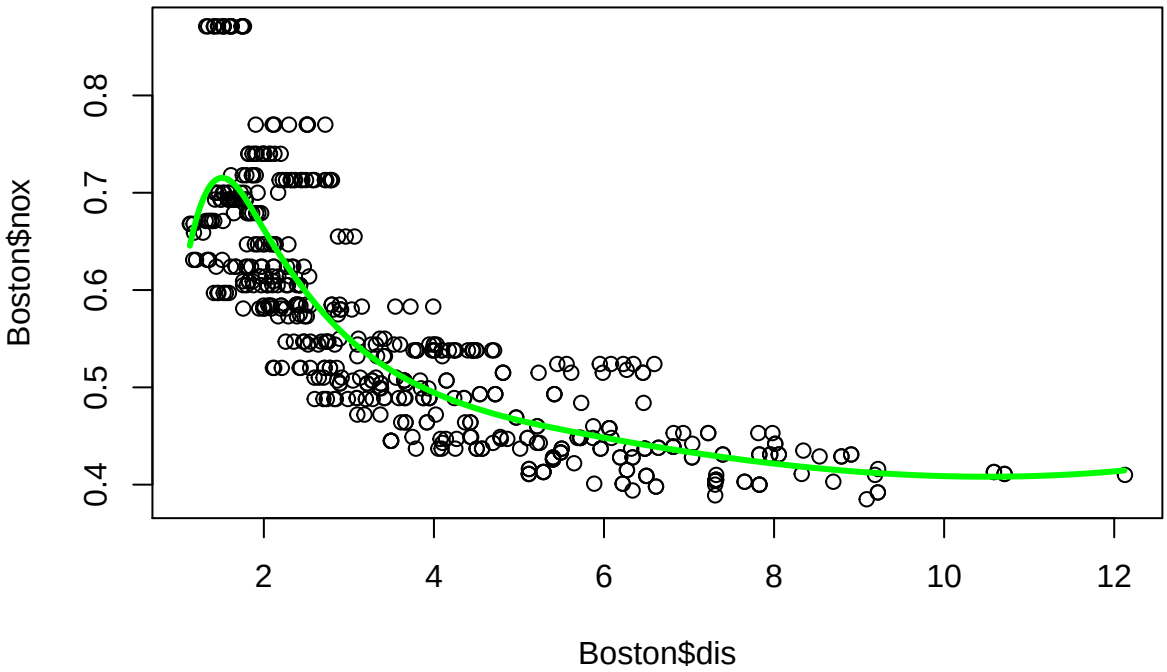

Degree of Freedom: 3
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis


Degree of Freedom: 4
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis

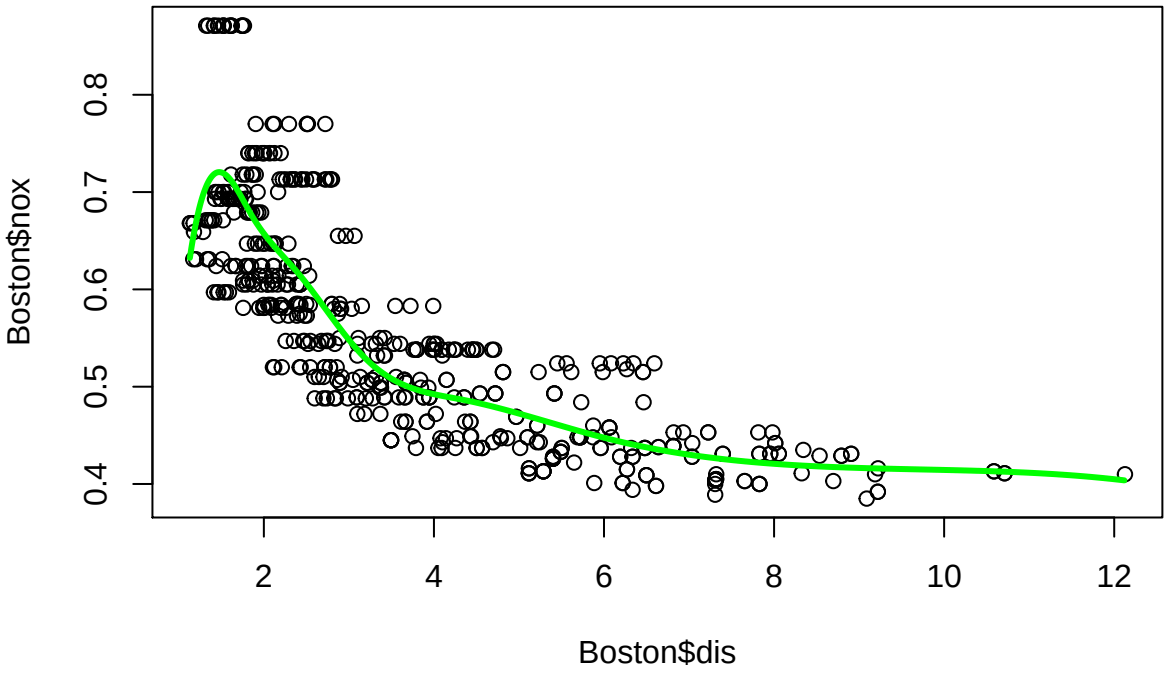

Degree of Freedom: 5
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis


Degree of Freedom: 6
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis

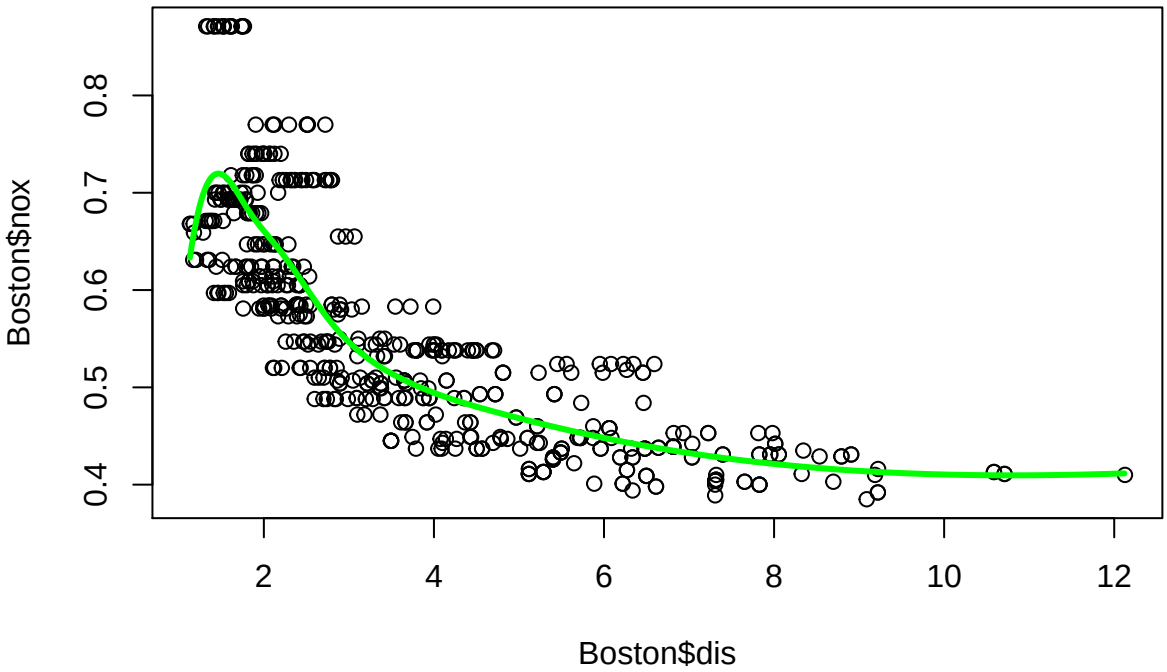
Degree of Freedom: 7



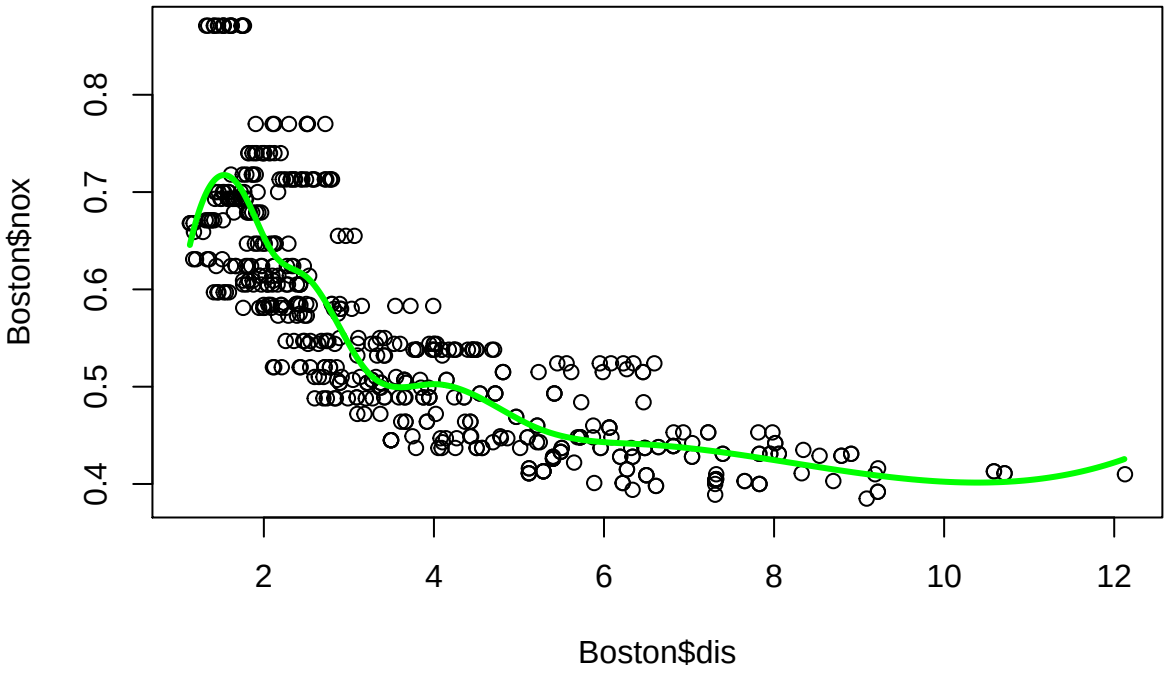
Degree of Freedom: 8



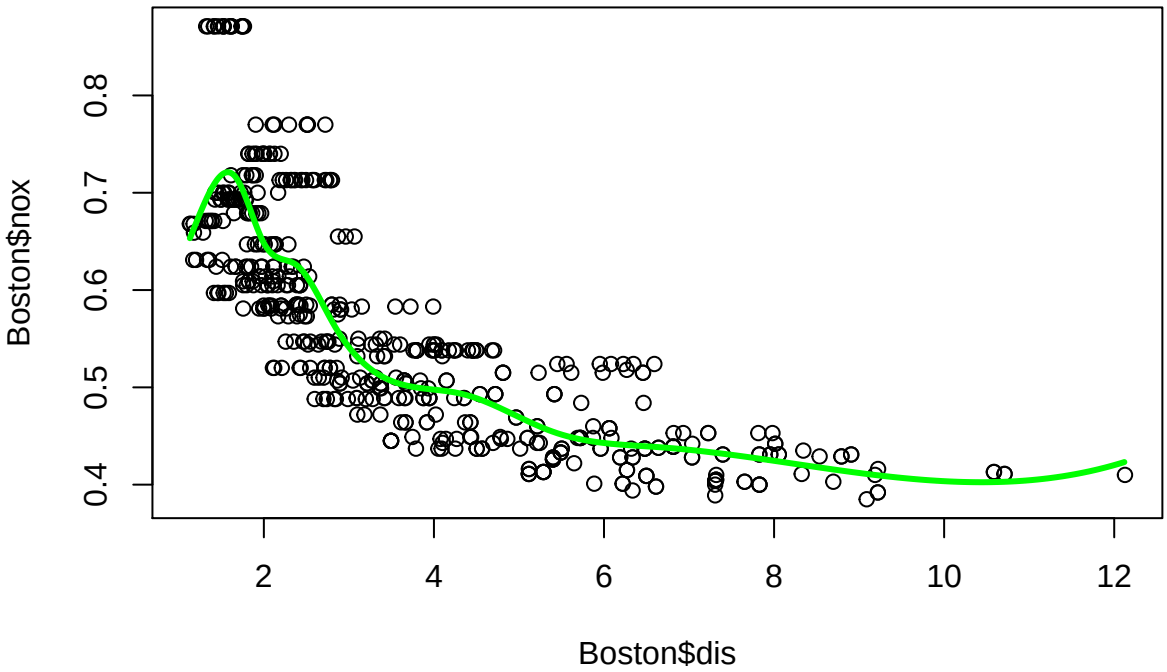
Degree of Freedom: 9



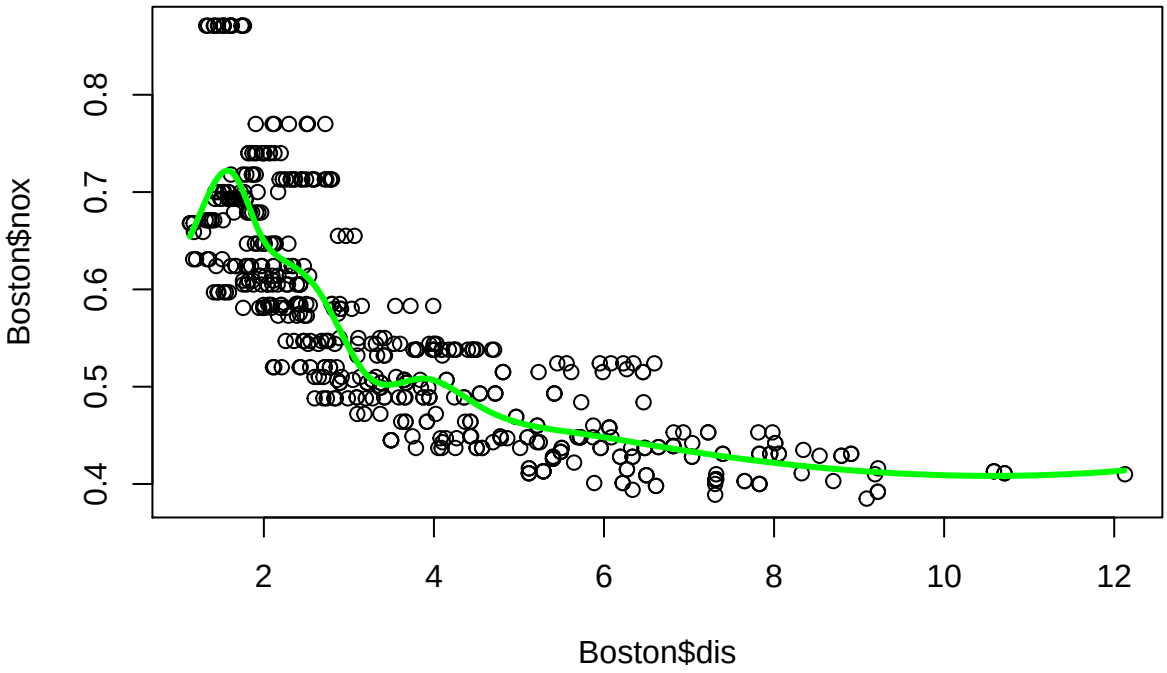
Degree of Freedom: 10



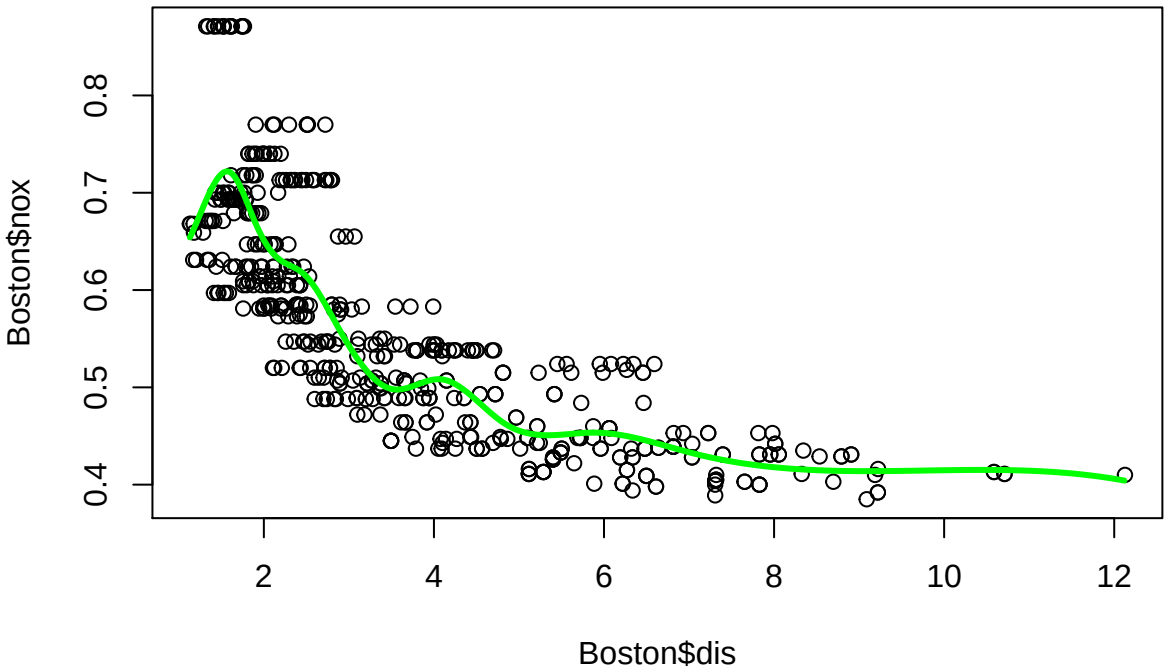
Degree of Freedom: 11



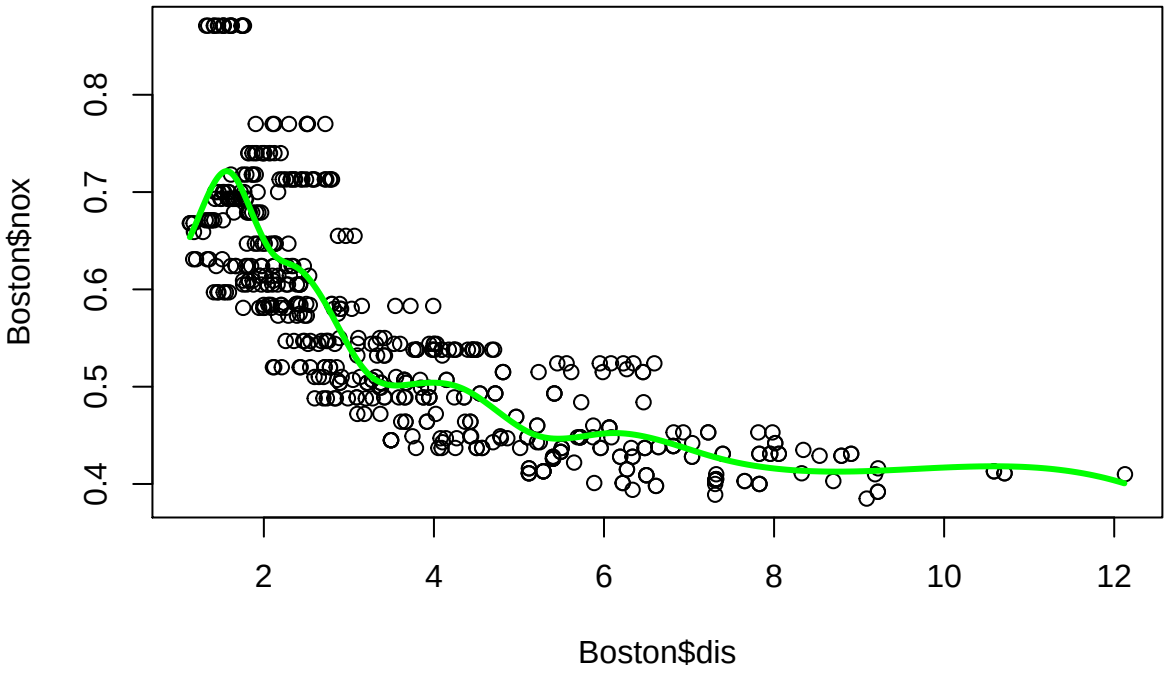
Degree of Freedom: 12

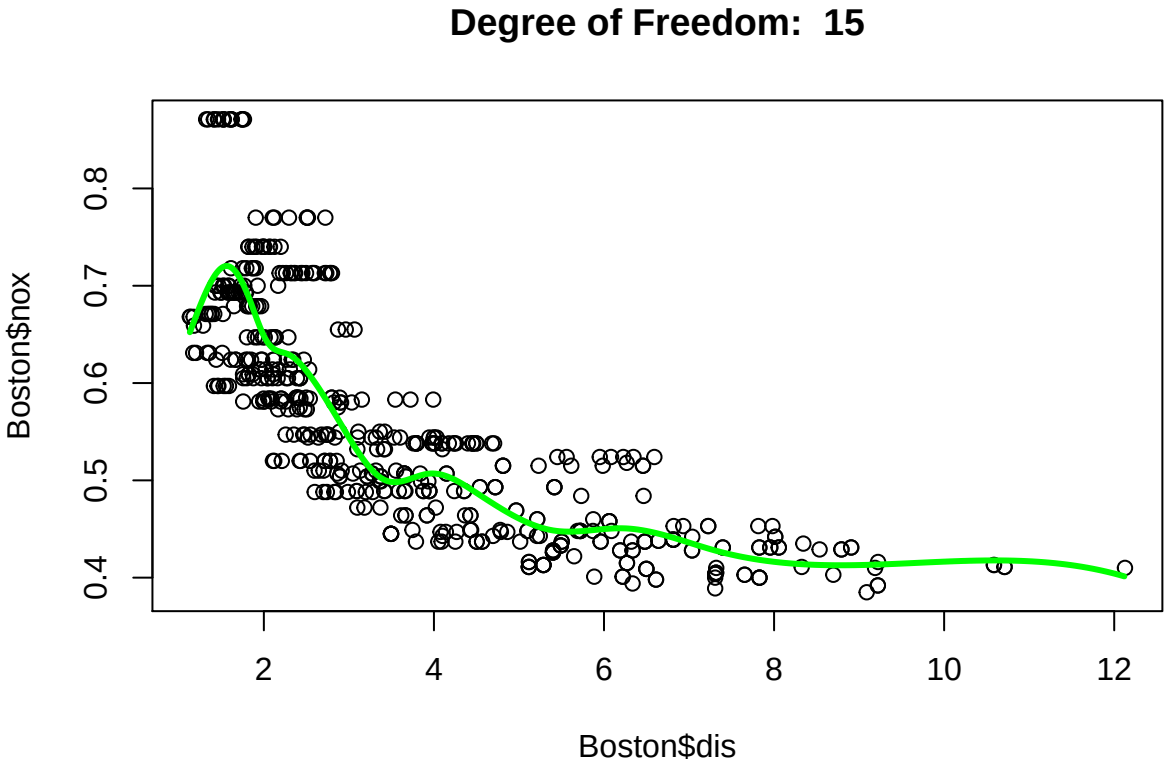


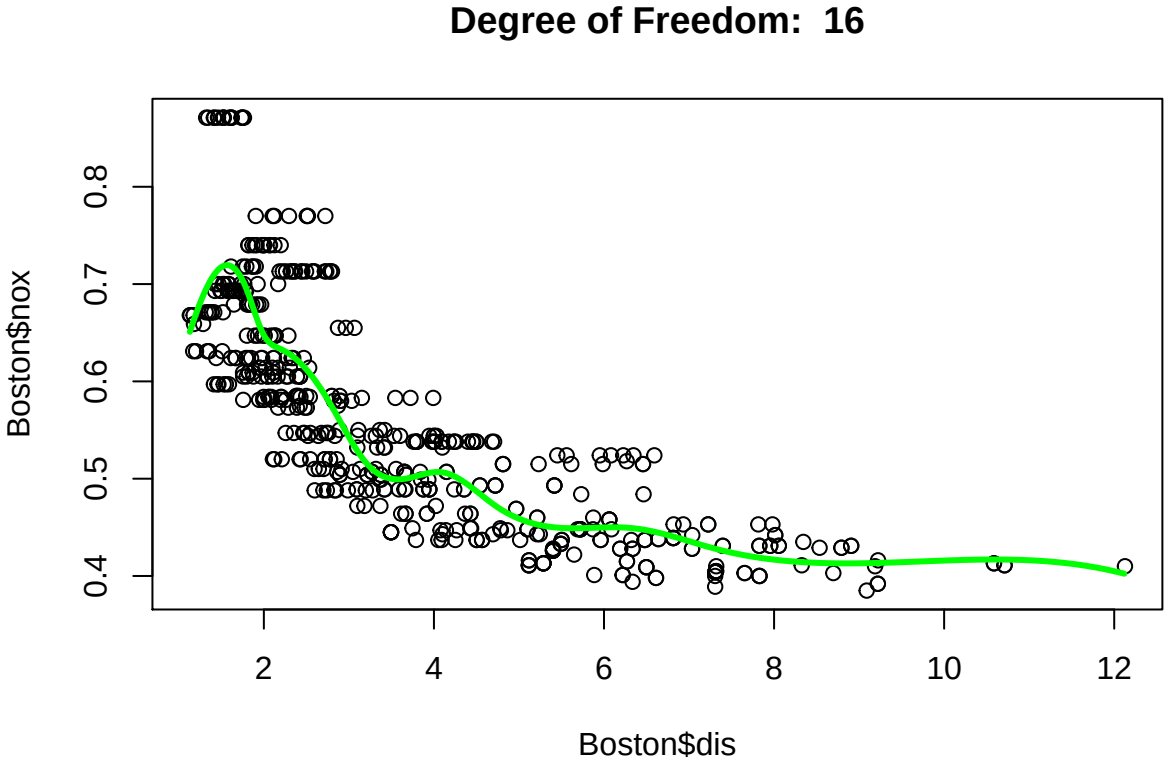
Degree of Freedom: 13

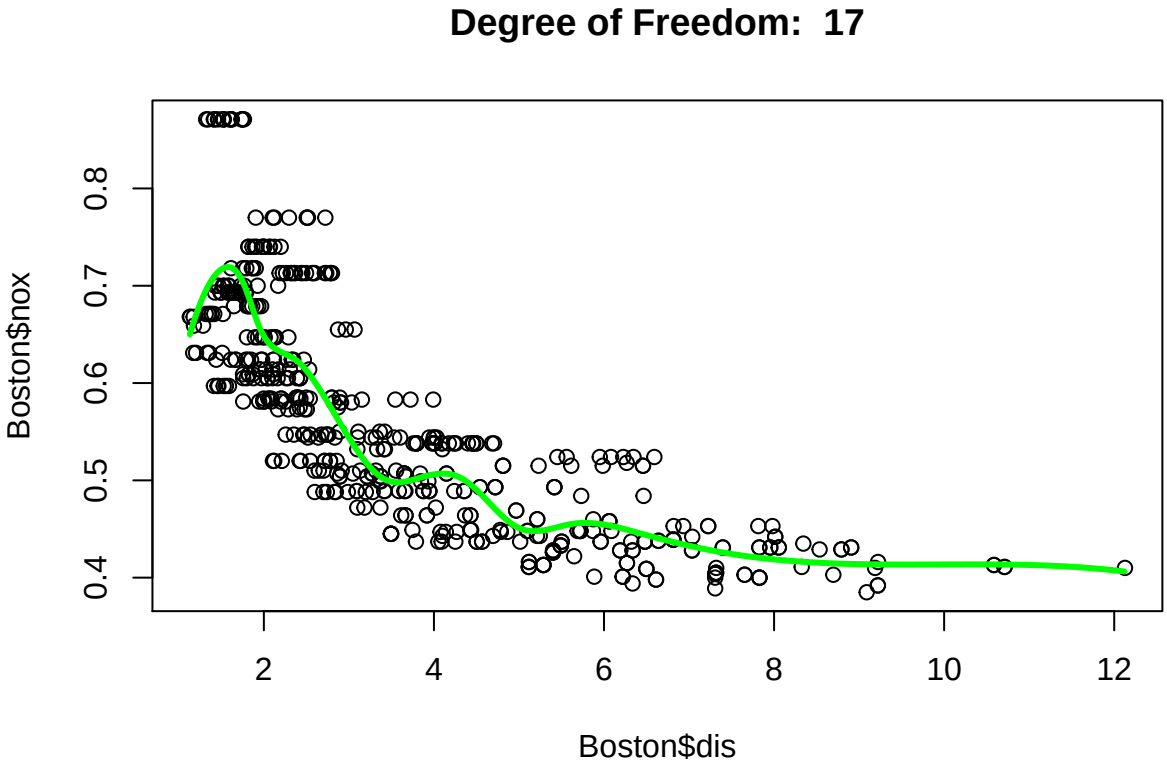


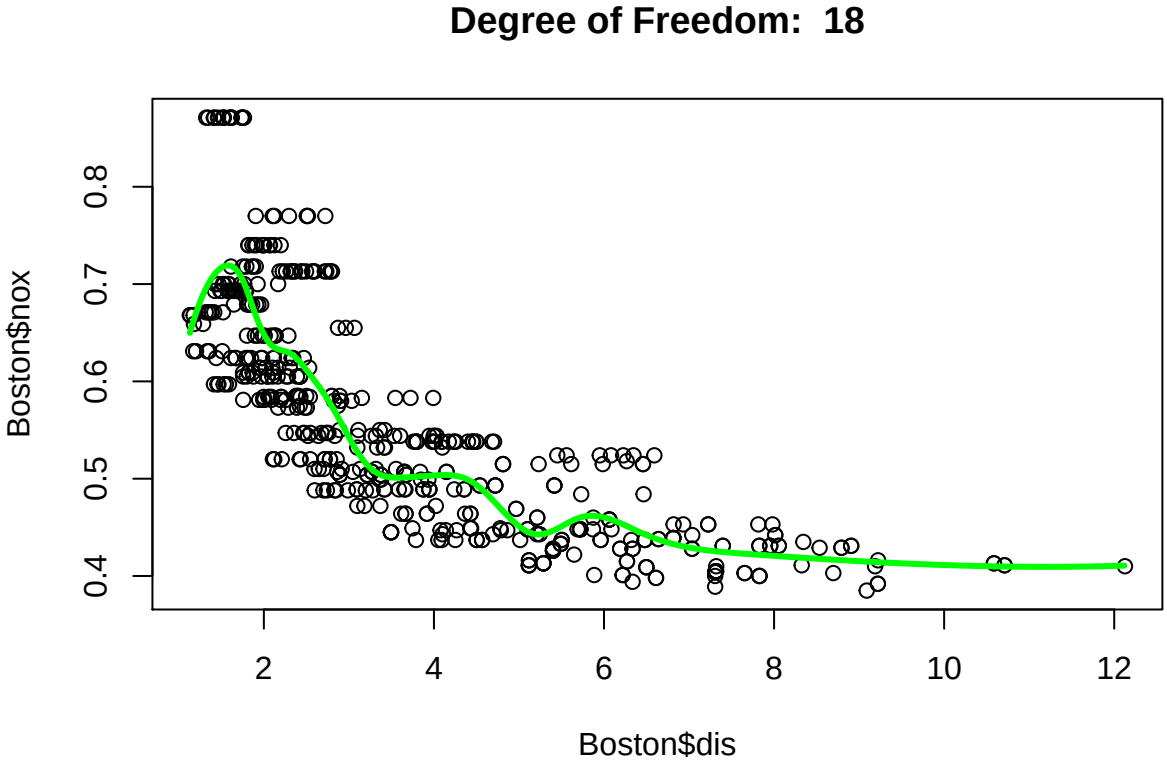
Degree of Freedom: 14

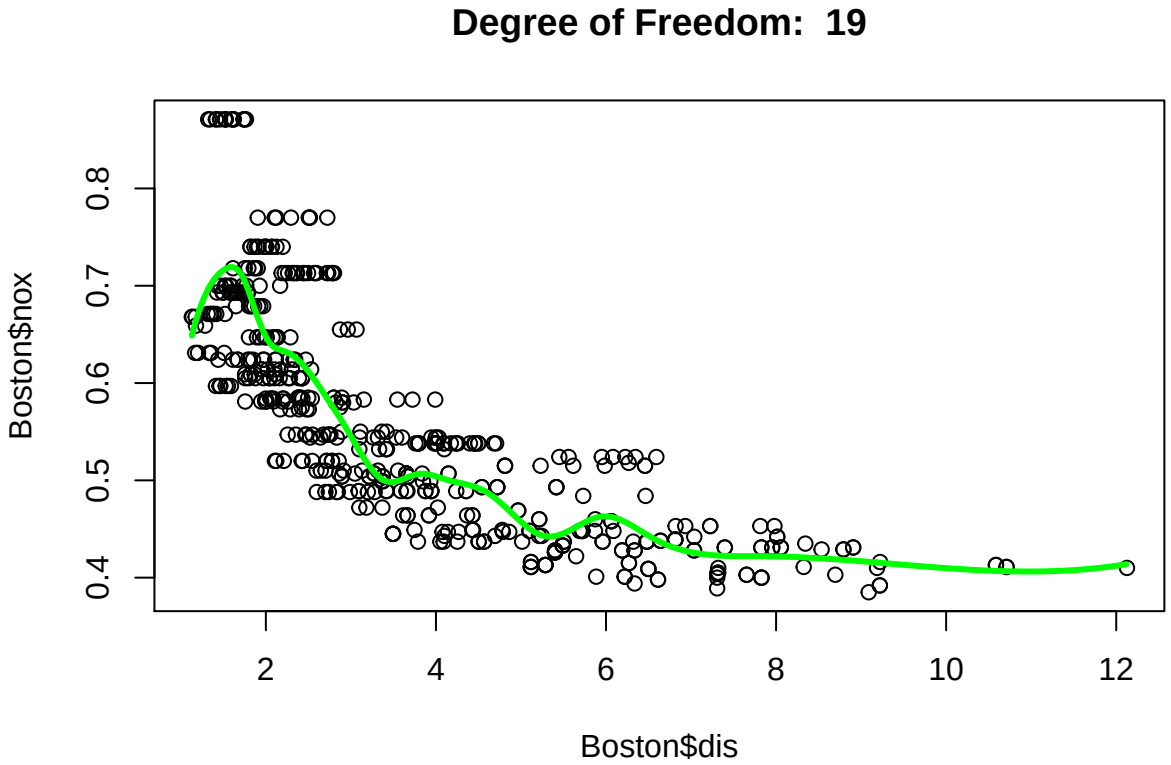


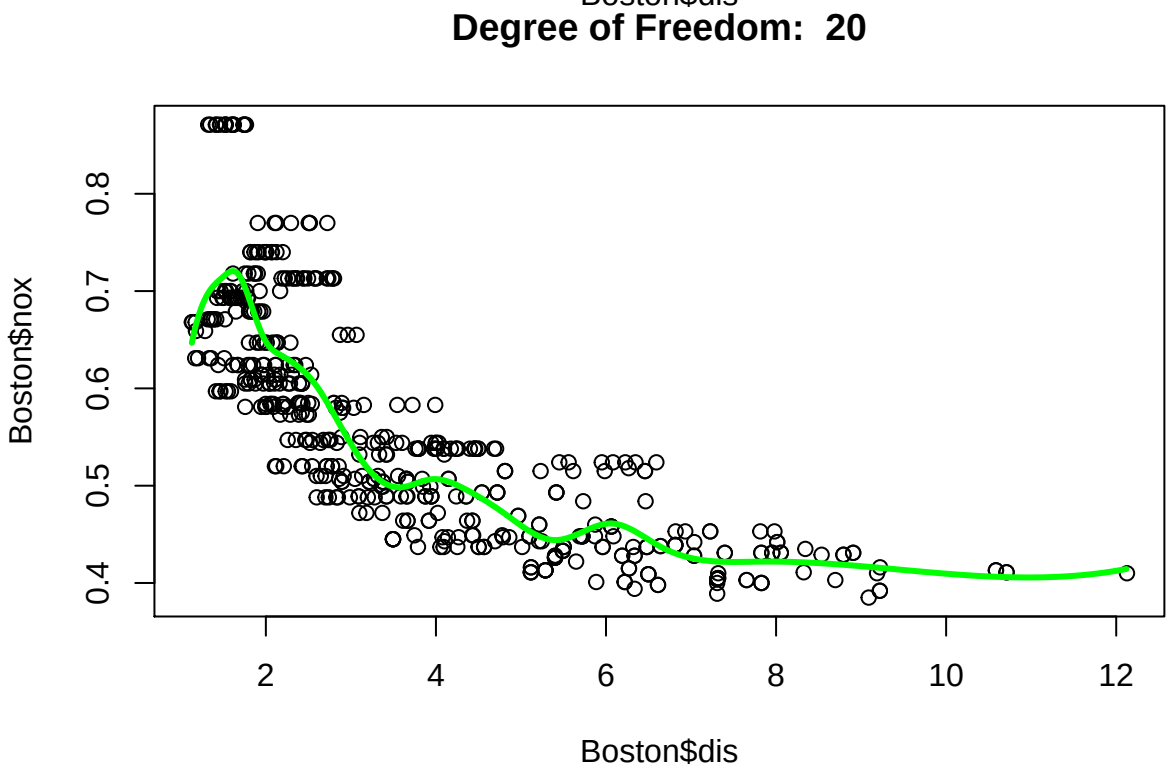

Degree of Freedom: 15
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis


Degree of Freedom: 16
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis

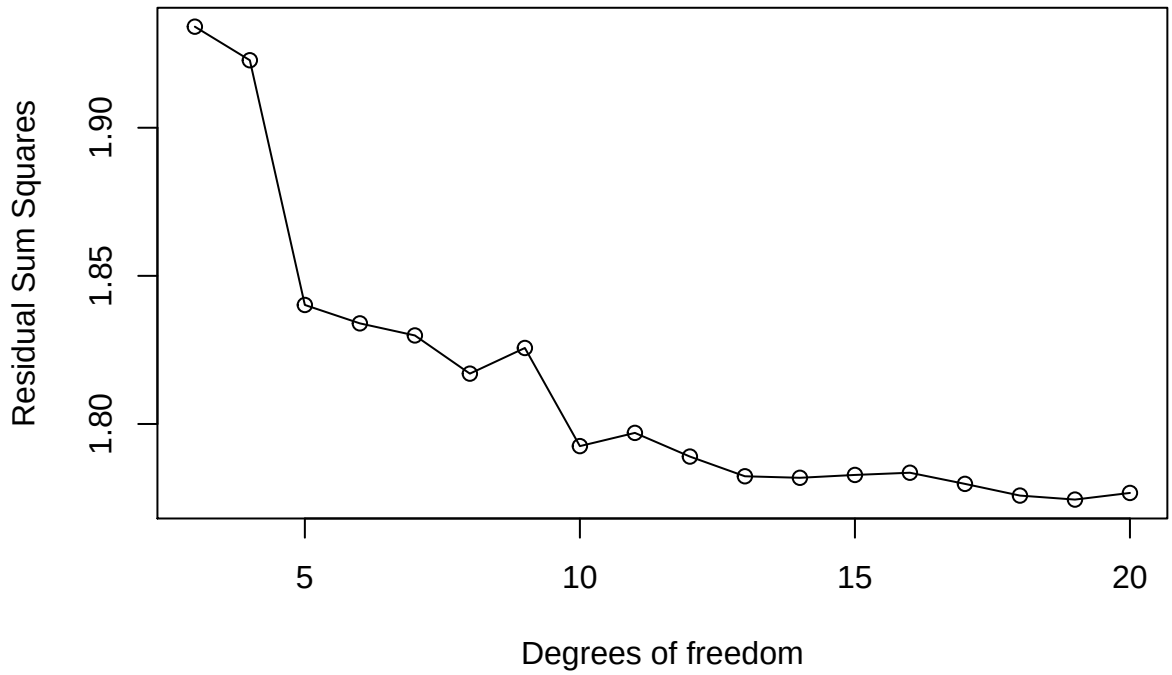

Degree of Freedom: 17
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis


Degree of Freedom: 18
Boston$nox
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
Boston$dis






```
plot(3:20, myRss[-c(1, 2)], xlab = "Degrees of freedom", ylab = "Residual Sum Squares", type = "o")
```




```
RSS<-myRss
# create a table
myTable<-data.table(RSS)
# take a look at the table
myTable[, Degree_of_Freedom := .GRP, by = list(c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20))]
```

```
##          RSS Degree_of_Freedom
##  1:       NA                 1
##  2:       NA                 2
##  3: 1.934107                 3
##  4: 1.922775                 4
##  5: 1.840173                 5
##  6: 1.833966                 6
##  7: 1.829884                 7
##  8: 1.816995                 8
##  9: 1.825653                 9
## 10: 1.792535                10
## 11: 1.796992                11
## 12: 1.788999                12
## 13: 1.782350                13
## 14: 1.781838                14
## 15: 1.782798                15
## 16: 1.783546                16
## 17: 1.779789                17
## 18: 1.775838                18
## 19: 1.774487                19
## 20: 1.776727                20
```

As the output shows, the RSS decrease generally as degrees of freedom increase. Although there are some fluctuations (for instance, from 10 degrees of freedom to 11 degrees of freedom, there is a slightly increase in RSS), the general trend of RSS value is decreasing.

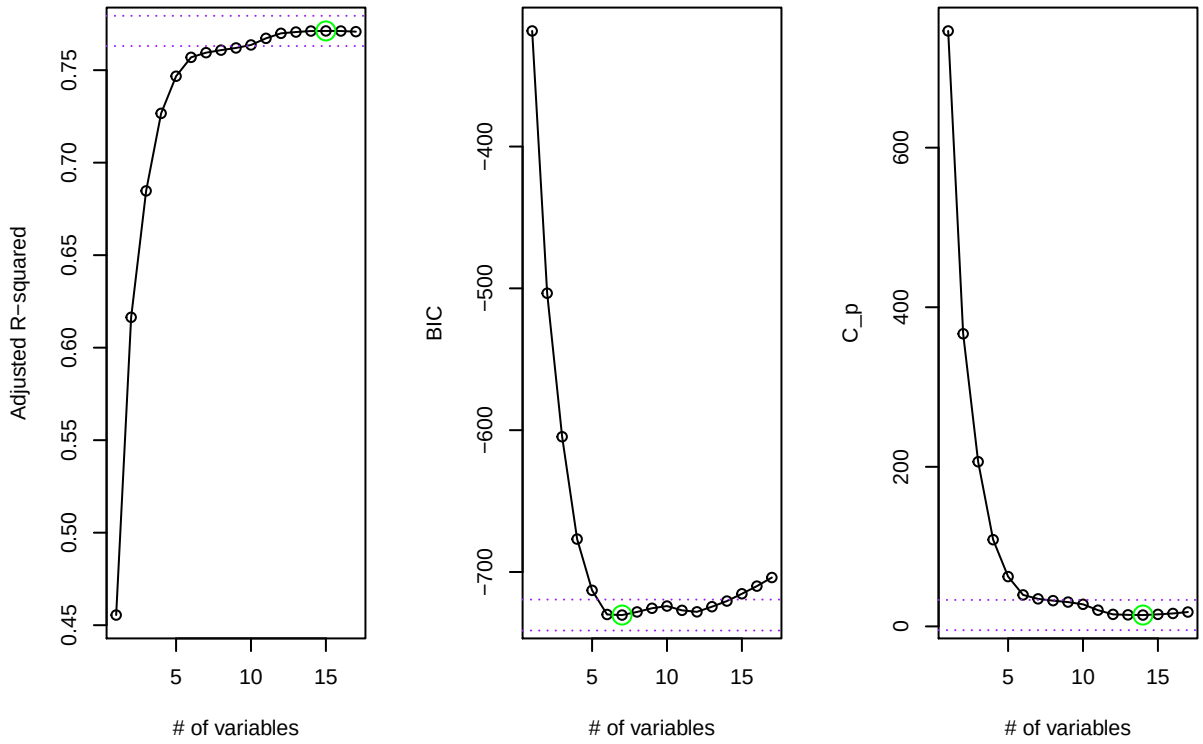
7.9 #10

a)

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.3.2
```

```
set.seed(2441)
attach(College)
# Record the total number of predictors.
n=length(College[1,])-1
# Use a 70-30 split into training and test data.
data.train <- sample(length(Outstate), length(Outstate) *0.7)
mytest <- College[-data.train, ]
mytrain <- College[data.train, ]
# perform forward stepwise selection on the training set.
model <- regsubsets(Outstate ~ ., data = mytrain, nvmax = n, method = "forward")
# record the summary of the model.
mysummary <- summary(model)
par(mfrow = c(1, 3))
# Plot Adjusted R-squared vs # of variables.
plot(mysummary$adjr2, xlab = "# of variables", ylab = "Adjusted R-squared", type = "o")
# plot the maximum point.
points(which.max(mysummary$adjr2), mysummary$adjr2[which.max(mysummary$adjr2)], col = "green",
       pch = 1, cex = 2)
# +/- 0.1 standard devitations of the maximum point.
abline(h = max(mysummary$adjr2)+ 0.1 * sd(mysummary$adjr2), lty = 3, col = "purple")
abline(h = max(mysummary$adjr2) - 0.1 * sd(mysummary$adjr2), lty = 3, col = "purple")
# Plot BIC vs # of variables.
plot(mysummary$bic, xlab = "# of variables", ylab = "BIC", type='o')
# plot the minimum point.
points(which.min(mysummary$bic), mysummary$bic[which.min(mysummary$bic)], col = "green",
       pch = 1, cex = 2)
# +/- 0.1 standard devitations of the minimum point.
abline(h = min(mysummary$bic) + 0.1 * sd(mysummary$bic), lty = 3, col = "purple")
abline(h = min(mysummary$bic) - 0.1 * sd(mysummary$bic), lty = 3, col = "purple")
# Plot Cp vs # of variables.
plot(mysummary$cp, xlab = "# of variables", ylab = "C_p", type = "o")
# plot the minimum point.
points(which.min(mysummary$cp), mysummary$cp[which.min(mysummary$cp)], col = "green",
       pch = 1, cex = 2)
# +/- 0.1 standard devitations of the minimum point.
abline(h = min(mysummary$cp) + 0.1 *sd(mysummary$cp), lty = 3, col = "purple")
abline(h = min(mysummary$cp) - 0.1 *sd(mysummary$cp), lty = 3, col = "purple")
```



As the output shows, adjusted R-squared achieved its highest at 15 variables, BIC achieved its lowest at 7 variables, Cp achieved its lowest at 14 variables. Theses are optimal number of variables for each of the criterion. However, they are different. Thus, I choose 11 variables model as the satisfactory model that uses just a subset of the predictors because this model's Adjusted R-squared, BIC and Cp are within 0.1 standard deviation of their optimal values of each of the three criterion.

```
# Take a look at the predictors that are selected from 11 variables model.
names(coef(model, id = 11))
```

```
##  [1] "(Intercept)" "PrivateYes"  "Apps"        "Accept"      "Top25perc"
##  [6] "Room.Board"  "Personal"    "PhD"         "S.F.Ratio"   "perc.alumni"
## [11] "Expend"      "Grad.Rate"
```

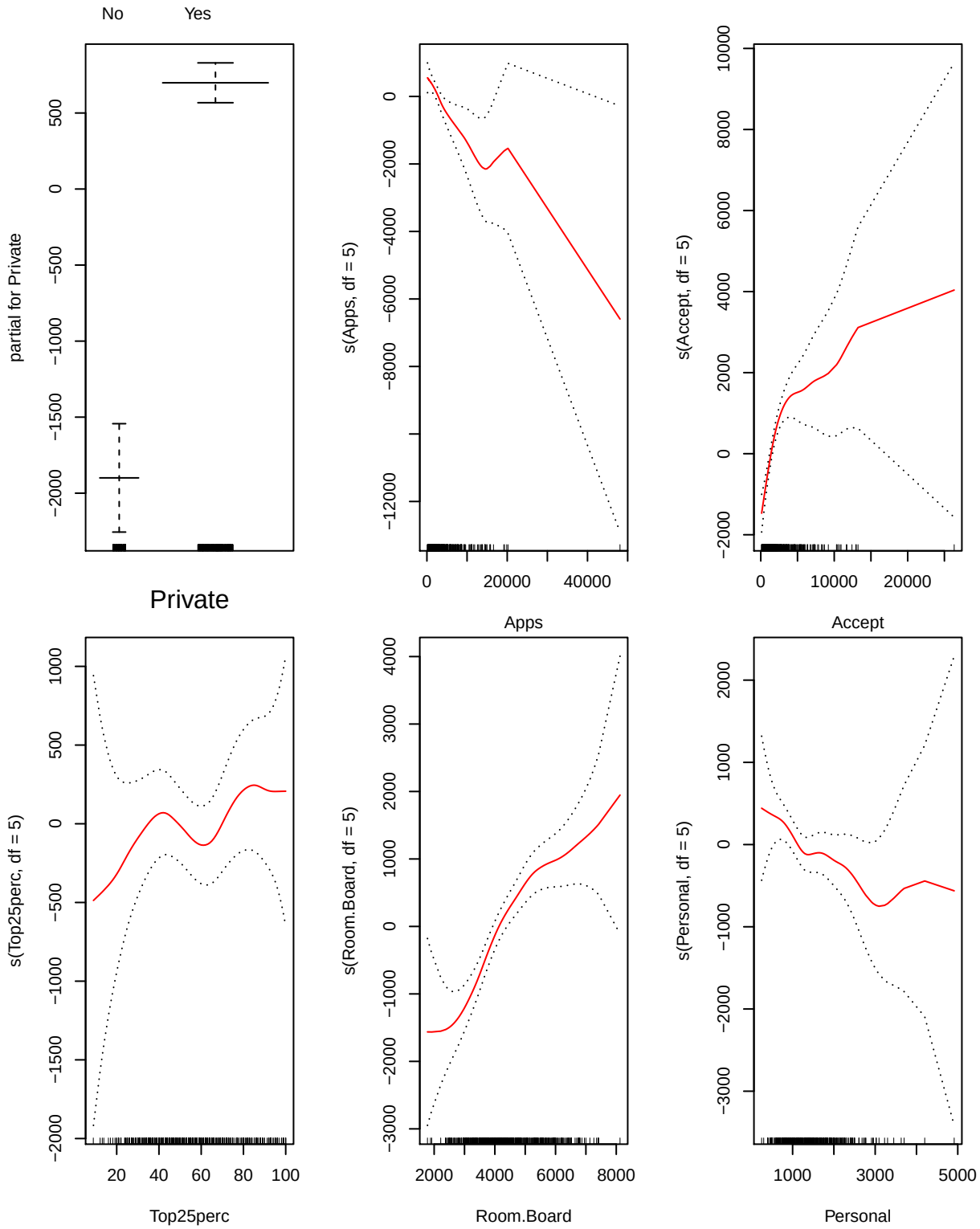
b)

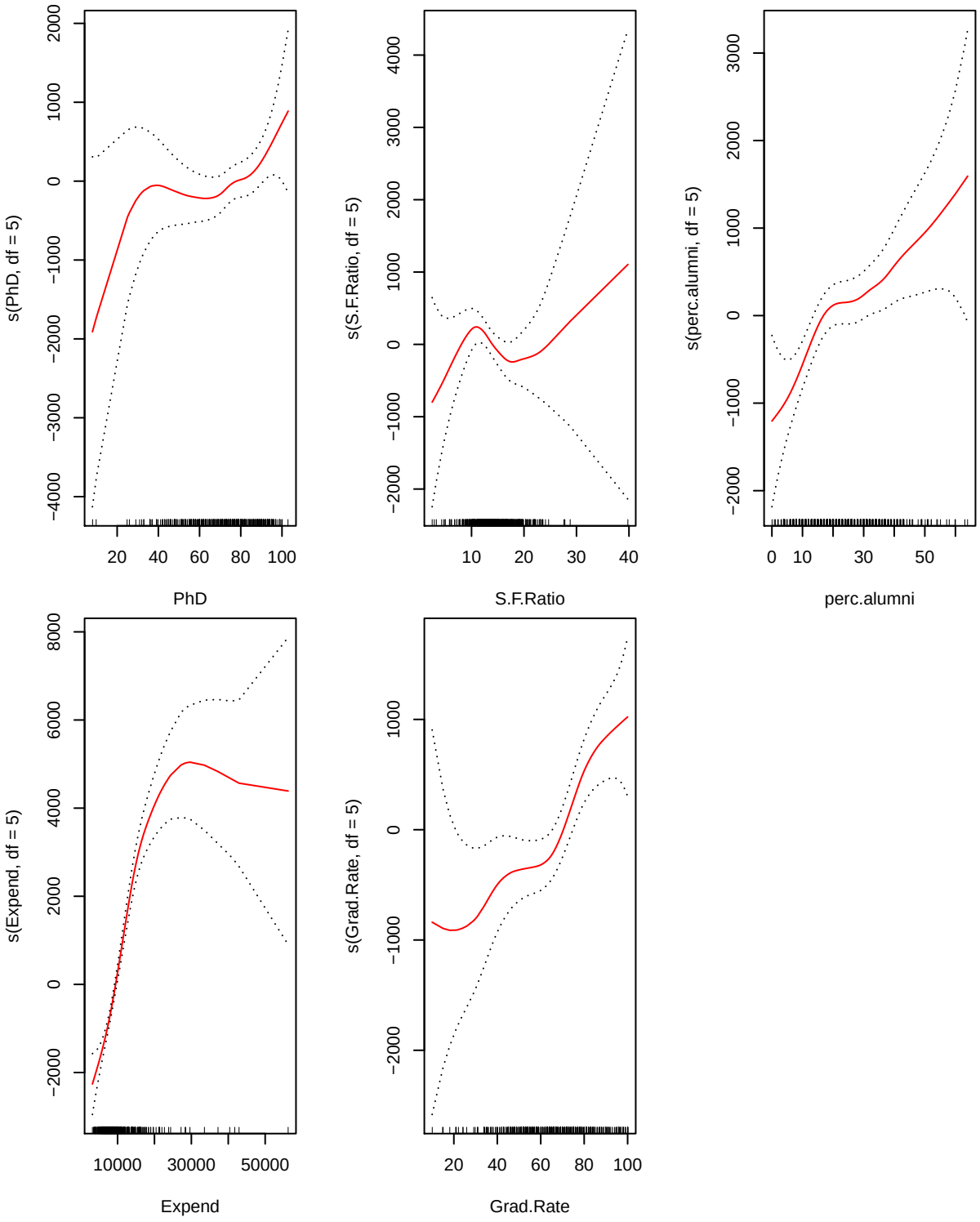
```
#install.packages("gam")
library(gam)
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.14
```

```
model <- gam(Outstate ~ Private + s(Apps, df = 5)+ s(Accept, df = 5) + s(Top25perc, df = 5) +
               s(Room.Board, df = 5) + s(Personal, df = 5) + s(PhD, df = 5)+ s(S.F.Ratio, df=5)
             +s(perc.alumni, df=5)+s(Expend, df=5)+s(Grad.Rate, df=5), data=mytrain)
par(mfrow = c(1, 3))
plot(model, se = T, col = "red")
```


No
Yes
partial for Private
Private
s(Apps, df = 5)
Apps
s(Accept, df = 5)
Accept
s(Top25perc, df = 5)
Top25perc
s(Room.Board, df = 5)
Room.Board
s(Personal, df = 5)
Personal



As the output shows, holding other variables constant, when 'Private' is yes, outstate tuision is high compared to when 'Private' is no. Holding other variables constant, when 'Apps' increase, outstate tuision decrease generally but fluctuate slightly in the middle. Holding other variables constant, when 'Accept' increase, outstate tuision increases in general. Holding other variables constant, when 'Top25perc' increase, outstate tuision increase moderately in general but fluctuate in the middle. Holding other variables constant, when 'Room Board' increase, outstate tuision increase in general. Holding other variables constant, when 'Personal' increase, outstate tuision decrease in general but fluctuate in the middle. Holding other variables constant,

when 'PhD' increase, outstate tuision increase in general but fluctuate in the middle. Holding other variables constant, when 'S.F.Ratio' increase, outstate tuision increase in general but fluctuate in the middle. Holding other variables constant, when 'perc alumni' increase, outstate tuision increase in general. Holding other variables constant, when 'Expend' increase, outstate tuision increase at first then followed by a decresing. Holding other variables constant, when 'Grad Rate' increase, outstate tuision increase in general but fluctuate slightly in the middle.

c)

```
# Calculate root mean squared of 11 predictors GAM model.
prediction <- predict(model, mytest)
error<-mean((mytest$Outstate - prediction)^2)
rms<- sqrt(mean((mytest$Outstate - prediction)^2))
cat("Root Mean Squared of 11 predictors GAM model:", rms)
```

```
## Root Mean Squared of 11 predictors GAM model: 2070.563
```

```
# Calculate root mean squared of standard linear model with all predictors included.
model.lm<-lm(Outstate ~., data=mytrain)
prediction.lm <- predict(model.lm, mytest)
rms.lm<- sqrt(mean((mytest$Outstate - prediction.lm)^2))
cat("Root Mean Squared for linear model with all predictors included:",rms.lm)
```

```
## Root Mean Squared for linear model with all predictors included: 2133.462
```

```
# Calculate test r-squared.
rsquqred <- 1 - error / mean((mytest$Outstate - mean(mytest$Outstate))^2)
cat("Test R squred of 11 predictors GAM model:",rsquqred)
```

```
## Test R squred of 11 predictors GAM model: 0.7575958
```

As the output shows, root mean squred (RMS) of 11 predictors GAM model is smaller than the standard linear model with all predictors inclueded. Test R squred is 0.7575958 which means 75.8% of the variability in 'Outstate' in the testing data set is described by the 11 predictors GAM model.