Assignment #7

Lindsay Brock, Gustavo Esparza, and Brian Schetzsle

10/17/2019

ISLR Chapter 7 Summary

Previously, we have improved linear models with dimension-reducing methods such as Ridge regression and Lasso that minimized the variance of estimates. However, all of these methods still function under the assumption of linearity, which can often provide a poor approximation of the data. In this chapter, we will move away from the Linear assumption based models and focus on alternative methods that can still be easily interpreted via inference and model prediction.

7.1: Polynomial Regression

An intuitive way to extend linear regression to a non-linear relationship between the predictor (X) and the response (Y) is to translate the standard linear model defined by $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$ to the polynomial function given by

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \cdots + \beta_d x_i^d + \epsilon_i \text{ where } \epsilon_i \text{ is the error term.}$$

This method, known as **Polynomial regression**, is able to produce very non-linear curves when the highest power d gets significantly large (IE a curve that has many local maximum and minimum values). By inspection of our coefficients and associated predictors, it can be observed that each x^i value is generated from the original x predictor by raising it to the ith power. Thus, estimating the coefficients can be performed via least squares (their Maximum Likelihood Estimates given the data).

Although it is possible to build polynomial regression models of large degrees, it is not usually recommended to extend beyond a power of three or four. This is due to the fact the regression curve can be overly flexible and produce undesirable shapes. Specifically, the end boundaries of our x variable will likely be predicted to be much lower or much higher than their actual values due to the end behavior of large degree polynomials.

We will now illustrate a 4^{th} degree polynomial curve fit (via least squares) for the variable **wage (in thousands of dollars)** as a function of **age** [displayed on the left]:

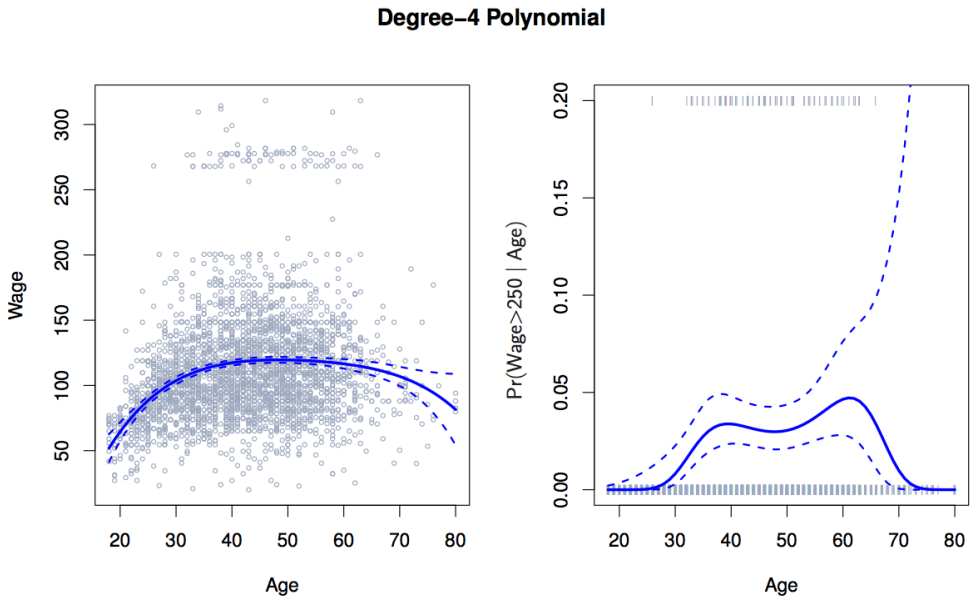


Figure 1: Polynomial Regression Model

Rather than focus on the regression estimates for each predictor age^i, we are more concerned with how the polynomial fits the data given our predictor and response. Overall, the regression curve (solid dark blue) does a fairly good job at predicting what appears to be the average/median wage for the age groups, specifically in the age $\approx$ 30-50 group. As the age variable tends towards the upper group of 70-80, we do observe less accuracy in modeling the relationship between age and wage. Looking only at the plotted data, it appears that there are two main clusters of wage groups, which implies that perhaps a categorical response is suitable for a logistic regression model (we will build such a model later in this section).

In addition to the main regression curve, a pair of dotted curves are plotted above and below the main curve. These two curves represent $\pm$ 2 $\times$ standard error for our regression curve, thus constructing a 95 % (under the assumption of Normality) Confidence Interval for our regression curve. The computation of this interval is quite interesting and can be explained as follows:

For an arbitrary value x_0, we have the following fitted polynomial regression value:

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \cdots + \hat{\beta}_j x_0^j$$

Now, we must consider the variance for $\hat{f}(x_0)$. As mentioned, we are able to use Least Squares in order to estimate each coefficient for our fitted model. In addition, we can estimate the variance of each coefficient estimate as well as the covariance between each of our coefficient estimates. By defining the covariance matrix of our coefficient estimates $\hat{C}$ and predictor vector l_0^T as follows:

$$\hat{C} = \begin{bmatrix} Var(\hat{\beta}_0) & Cov(\hat{\beta}_0, \hat{\beta}_1) & \dots & Cov(\hat{\beta}_0, \hat{\beta}_j) \\ \vdots & \ddots & \dots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ Cov(\hat{\beta}_j, \hat{\beta}_0) & \dots & \dots & Var(\hat{\beta}_j) \end{bmatrix}, \text{ and } l_0^T = (1, x_0, x_0^2, \dots, x_0^j)$$

The variance at x_0 is then estimated as : $\text{Var}[\hat{f}(x_0)] = l_0^T \cdot \hat{C} \cdot l_0$. This estimated variance can be square rooted to provide a point wise estimate for the standard error at x_0. By performing this standard

error computation over all of our x_0 values, we can plot the ± 2 standard error curves alongside our original regression curve. Referring back to figure 1, it is clear that the 95 % confidence interval for the final age group (65-80) is larger than the interval for all other ages. Although not dramatically large, it still provides an illustration of higher degree polynomials being unreliable on the boundaries of our x values.

Now, we will return to the idea of splitting the response variable of **wage** into two categories and building a logistic regression model based on our 4^{th} degree polynomial. From figure 1, it appears that a higher wage group consists of any subject that makes more than 250 thousand dollars a year. Thus making this binary distinction provides the following model:

$$Pr(y_i > 250|x_i) = \frac{exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d)}{1 + exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d)}$$

Using this Binomial Logistic Regression model yields the right hand plot in Figure 1. Similar to the lefthand plot, we see our regression curve as a solid curve and our 95 % Confidence Interval given by the two dotted curves. In addition, we can observe tick marks on the top and bottom of our graph that indicate which group the subject of age x belongs in. In this context, the blue curve indicates the probability that a subject of age x will belong to the wage group that makes more than 250 thousand dollars a year. It is important to make note of the extremely wide confidence interval observed for the 70-80 age group. This high variance shows that our logistic regression is not a strong model for categorizing our subjects into the two wage groups.

7.2: Step Functions

The utilization of Step functions is a useful alternative to Polynomials when one wants to avoid a global structure. That is, we do not want to consider a single function for all of the data provided. In such instances, we can divide our X values into *bins* (intervals) and fit a difference constant value to each bin. Essentially, by rejecting the idea of a global structure, we have converted our continuous variable into an ordinal categorical variable.

Going further, we can dissect the bins in such a way that there are defined **cutpoints** c_i that collectively construct X in the following manner:

$$\begin{aligned} C_0(X) &= I(X < c_1) \\ C_1(X) &= I(c_1 \leq X < c_2) \\ &\vdots \\ C_k(X) &= I(c_k \leq X) \end{aligned}$$

Here, I is defined as the indicator function that returns 1 if X is in the range of the bin and 0 otherwise, commonly known as a dummy variable for categorical data. It is now obvious that the $C_i(X)$ cutpoint variables can only produce a single response of "1", since all of the cutoff variables have no intersection. Then, considering $C_0(X)$ as the null outcome when all other $C_i(X)$ indicator functions produced a "0" result, we have the familiar least squares fit given by:

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \cdots + + \beta_k C_K(x_i) + \epsilon_i$$

The selection of $C_0(X)$ as the default outcome was arbitrary, but resulted in the standard format for regression models. We can now interpret β_0 as the average value of our response Y when our predictor X is in the range for step $C_0(X)$. Then, for all other outcomes, β_i is interpreted as the average change in Y when X is in the range for step $C_i(X)$, **relative** to the null value at β_0.

Once again, using the age and wage data, we can build a least squares step function. The resulting plot is provided below:

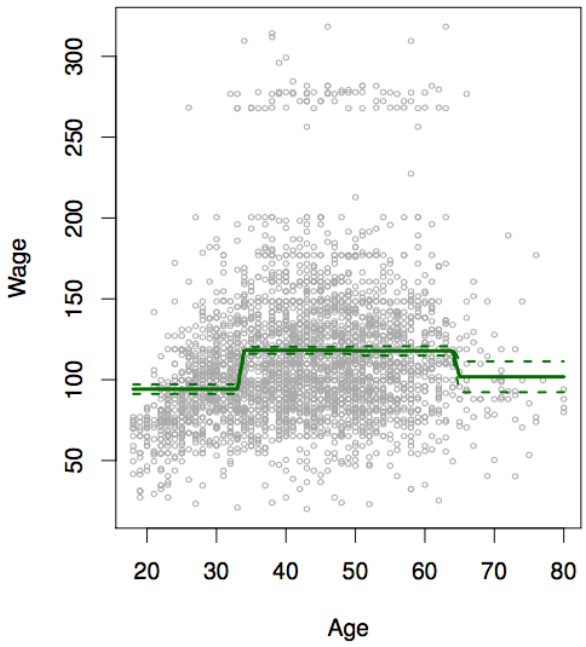


Figure 2: Step Function Regression

This example illustrates how inefficient a step function models our data when the bins are poorly divided. For instance, the first bin is defined as the age group between 20 and about 35. Thus, our step function estimates the mean wage to be around 100 thousand dollars. However, we can clearly see an increase in wage within that bin, which implies that the bin is possibly categorizing two separate age/income groups into the same group. It is apparent that the step function is optimal when considering data that has clear cutoffs within the predictor space and a general trend for the response.

7.3: Basis Functions

Thus far, we have covered polynomial regression and step functions. These two types of functions can be fit into the general category of **Basis Functions**. That is, we avoid the usual linear model of Y given X and opt for the model of form:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + +\beta_k C_b(x_i) + \epsilon_i$$

Where $b_i(X)$ is a predetermined function that is known. In polynomial regression our basis function was raising X to various powers and the Step Function utilized the Indicator as its respective basis function. As previously mentioned, all $b_i(X)$ values provided in *basis functions* are simply modifications of our original X and are therefore fair game for least squares estimation. As a result, we are then able to apply any inference that was provided by least squares (confidence intervals using standard error and statistical significance F-tests). This concept of *Basis Functions* will be explored further in the following sections.

7.4: Regression Splines

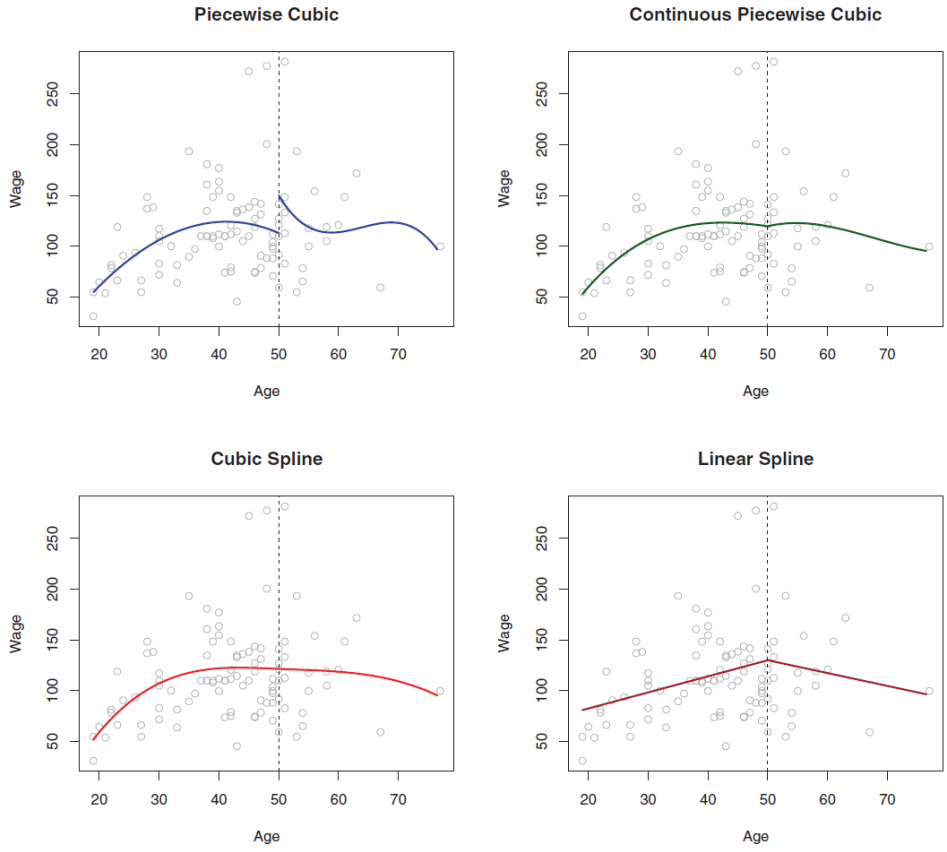
Expanding upon the polynomial regression and piecewise constant regression discussed above, the chapter continues with a discussion of a more flexible class of basis functions called regression splines. The first of this category is the *piecewise polynomial regression*. To perform this type of regression, separate low-degree polynomials must be fit over different regions of X. To illustrate this method let's look at an example of a piecewise cubic polynomial regression fit (below), where the coefficients (βs) change depending on different ranges of X. The values of X where the βs change (change points) are called *knots*.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i$$

If it has no knots, it would continue to be a standard cubic polynomial. With a single knot (only one change point) at a point c it would be modeled as,

$$y_i = \begin{cases} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 + \epsilon_i & if \quad x_i < c \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 + \epsilon_i & if \quad x_i \geq c \end{cases}$$

It essentially fits two different polynomial functions to the data each with a different set of βs, one along the support of $x_i < c$ and the other $x_i \geq c$, and can be fit using least squares applied to simple functions of the original predictor. As each function has four parameters (four βs) and there are two functions in the model, altogether it will have 8 *degrees of freedom* (8 parameters to estimate). Increasing the number of knots also allows for a more flexible piecewise polynomial and for every K knots there will be $K+1$ polynomial fits.



Continuing with our piecewise polynomial example, we will fit a data set called **Wage** to this cubic polynomial, specifically with **age** as our predictor and **wage** as the response, to illustrate how we are able to add *constraints* and *splines* to the model to fix the issues seen in the plots above. The dotted line in each

plot represents the knot. In the top left plot, the model is too flexible which makes the resulting fitted curve discontinuous. The plot in the top right depicts a curve that has been fit under the constraint that it must be continuous, fixing the discontinuity issue of the first plot but still not quite as smooth as we would like it to be. Moving on to the lower left plot (called a *cubic spline*), we are *adding* two more constraints (that the first and second *derivatives* of the piecewise polynomials be continuous) for a total of three constraints. It is important to note that for every constraint a degree of freedom is removed, so now that three constraints have been added we have 5 degrees of freedom left. (The use of these degrees of freedom is discussed later.) Another way to determine the degrees of freedom for cubic splines is to add 4 to the number of knots it contains. Our example had 1 knot and adding that to 4 we end up with 5 degrees of freedom. Finally, the lower right plot (the *linear spline*) is created by fitting a line in each region of the predictor space between the knots while requiring a continuity constraint for each knot.

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i$$

To explain how to fit these last two spline plots we can use the previously explained basis function model to represent a regression spline. Above is the model of a cubic spline with K knots and pre-selected basis functions $b_1, b_2, \ldots, b_{K+3}$, which can be fit using least squares. Although there are many ways, the most direct approach to represent a cubic spline using the equation above is to begin with the polynomial terms (x, x^2, x^3) and then a *truncated power basis* function per knot. The function is described below where ξ is the value of the knot, and because of the support this term is turned on and off, so to speak, depending on the value of x.

$$h(x, \xi) = (x - \xi)_+^3 = \begin{cases} (x-\xi)^3 & if \quad x > \xi \\ 0 & otherwise \end{cases}$$

Recall the model $y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i$, adding this term as $\beta_4 h(x, \xi)$ will keep the equation from becoming discontinuous up until the 3^{rd} derivative because $(x - \xi)^3$ is raised to the third power. The final model with K knots will have the terms $X, X^2, X^3, h(X, \xi_1), h(X, \xi_2), \ldots, h(X, \xi_K)$. Since the model uses an intercept and 3 predictors, it will be estimating $K + 4$ regression coefficients. This results in $K + 4$ degrees of freedom for a cubic spline with K knots.

When the value of X is unusually large or small when compared to the others, splines can end up having a large variance. This can be fixed using a *natural spline*, or a regression spline with extra *boundary constraints*, which means it would be required to be linear in the region where X is smaller than the smallest knot or larger than the largest knot. Adding this constraint allows the splines to produce more stable estimates in these regions.

Now that we know how to use them, we must be able to decide the locations of the knots. An initial approach would be to place them arbitrarily, more of them where we might think the function would change rapidly and fewer where it is more stable. Another approach used more often, is to place them in a uniform fashion. This can be done using software and specifying the degrees of freedom where it automatically selects the knots and places them at uniform quantiles of the data, i.e. using 4 degrees of freedom they are placed at the $25^{th}, 50^{th}$, and 75^{th} percentiles.

Using this approach also leads to the question of how to select the degrees of freedom or number of knots. Again, an arbitrary approach is to select different degrees of freedom/knots and view the fitted curve it produces to select which is best. As always, cross-validation can also be used. Beginning with a starting number of knots, the data is split into testing and training groups where the training data is used to fit a spline which is then used to make predictions on the testing data. This is repeated until each observation has been included in the testing group at least once, and finally the overall CV RSS is calculated. This will be done for different quantities of knots until the smallest RSS is selected indicating the optimal number of knots.

When compared to polynomial regression, regression splines often give better fits resulting in more accurate predictions. This is due to its increased flexibility arising from the use of the knots while keeping

the degree of the polynomials fixed which produces a more stable result, whereas polynomial regression must use terms with exponents of an extremely high degree to produce flexible fits that are less stable. With the use of splines we are able to place even more knots, specifically concentrating the knots in regions where the function changes rapidly and placing fewer where the function is more stable, which increases flexibility and accuracy even further.

7.5: Smoothing Splines

Moving on from regression splines, the chapter continues with another method of producing splines called the *smoothing spline* which is a function represented by $g()$ that minimizes the equation below.

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

When fitting a model, our goal is to find some function that fits our data well so that we are able to make accurate predictions. We do this by minimizing the RSS which in this case is $\sum_{i=1}^{n}(y_i - g(x_i))^2$. We can minimize the RSS to zero by selecting just the right function $g()$ but this most often leads to a gross overfit of data due to its over-flexibility. The goal of this method is to not only find some function of g that fits the data well but is also smooth. This is where the second term in the equation above comes into play, where the λ is a non-negative *tuning parameter*.

As you may have already noticed, it looks awfully similar to the equations for ridge regression and lasso. That is because it includes a *loss function* $\left(\sum_{i=1}^{n}(y_i - g(x_i))^2\right)$ and a *penalty term* $\left(\lambda \int g''(t)^2 dt\right)$. Since the loss term minimizes the RSS it produces a function g that fits well while the penalty term does as its name describes and penalizes the variability of that function g. We use its second derivative because it represents the amount in which the slope is changing which can also be interpreted as a measure of its *roughness*. It is integrated (essentially summed) over the range of t which calculates the total change in the function $g'(t)$ over that range. If g is smooth, then $\int g''(t)^2 dt$ will be small because there is less of a change in slope over the specified range. If g is bumpy with various changes in slope, $\int g''(t)^2 dt$ will be large because those changes are all summed together. Adding the λ helps to control the smoothness of g, with increasing λs creating a smoother g eventually becoming the straight line produced by least squares. Just as in the ridge and lasso summary from our last assignment, this λ controls the bias-variance trade-off of using the smoothing spline.

It is also important to note some special properties about this function g. It is a piecewise cubic polynomial with continuous first and second derivatives at each knot and the knots are at each unique value of the predictor, instead of simply placed in uniform positions. The minimizing g is also a natural cubic spline with knots at each unique value of the predictor but is not the same natural cubic spline discussed previously in this chapter. Instead it is a *shrunken version* with the λ controlling the amount of shrinkage.

With so many knots at each unique value of x_i, you might be wondering if the corresponding excessive degrees of freedom might eventually cause an issue. The degrees of freedom end up being regulated by λ which controls the roughness of the smoothing spline and the resulting *effective degrees of freedom* (df_λ) actually decrease in number (from n to 2) as $\lambda \to \infty$. Normally the degrees of freedom come from the total number of free parameters (i.e. the number of coefficients in a polynomial or cubic spline) but in this case these parameters are constrained (shrunken down) by λ. As the df_λ increase, the flexibility of the smoothing spline also increases which means lower-bias and higher-variance and vice-versa. To truly understand what effective degrees of freedom are, we must explain them mathematically.

$$\hat{\boldsymbol{g}}_\lambda = \boldsymbol{S}_\lambda \boldsymbol{y}$$

In the equation above $\hat{\boldsymbol{g}}$ represents the solution to minimizing $\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$ for an appropriately selected λ. Specifically, $\hat{\boldsymbol{g}}$ is an n-vector which contains the fitted values of the smoothing spline using training data. $\boldsymbol{S}_\lambda$ is an $n \times n$ vector which is dependent upon the selected λ (for which there is

a formula but it is not given) that when multiplied by the response vector $\boldsymbol{y}$, results in these fitted values. Using this information, the following effective degrees of freedom are just the sum of the diagonal elements of the matrix $\boldsymbol{S}_\lambda$ as described in the equation below.

$$df_\lambda = \sum_{i=1}^{n} \{\boldsymbol{S}_\lambda\}_{ii}$$

Again, this brings us to the discussion of selecting the appropriate λ. Cross-validation, specifically leave-one-out, can always provide an answer by finding the λ that minimizes the RSS. It is just as efficient as computing a single fit and it uses the formula below.

$$RSS_{cv}(\lambda) = \sum_{i=1}^{n} (y_i - \hat{g}_\lambda^{(-i)}(x_i))^2 = \sum_{i=1}^{n} \left[\frac{y_i - \hat{g}_\lambda(x_i)}{1 - \{\boldsymbol{S}_\lambda\}_{ii}} \right]^2$$

In this equation $\hat{g}_\lambda^{(-i)}(x_i)$ is the fitted value for the smoothing spline using the training data while leaving out one observation (the i^{th} observation (x_i, y_i)), evaluated at x_i. On the right side of the equation, $\hat{g}_\lambda(x_i)$ is the smoothing spline function that is fit using *all* of the training data (not leaving any out), evaluated at x_i. What this formula is really describing is that we are able to calculate the RSS for a particular λ by fitting the smoothing spline to all of the original data, subtracting it from the response, and dividing it by $1 - \{\boldsymbol{S}_\lambda\}_{ii}$. This is done for a range of λs and the one with the smallest RSS wins! It is much simpler than the initial leave-one-out strategy. In the end we are ultimately looking for a λ that will result in the fewest possible effective degrees of freedom as a simpler model is preferred when compared to a more complex model, unless that complex model is affirmed by the data.

7.6: Local Regression

Local regression is another method of fitting a curve to data that is sensitive to local patterns, meaning that it has the flexibility to model local trends that might not show up at the global scale. Basically, you choose a point x_0 and perform a regression (the book says non-linear but you could also use a linear function) using only data in the neighborhood of x_0; you then calculate the predicted value at x_0 according to this regression. Data that are closer to x_0 are given more weight in the regression so the method of fitting is weighted least squares.

Unlike the previous methods, local regression does not yield a closed-form function of the predictors, $f(X)$. Instead, it is an algorithm for computing the fitted value at any particular value of the predictors, $X = x_0$. You can get a curve by calculating the fitted value over a range of X and then connecting those fitted lines, perhaps with splines. Because the full training data is necessary for every prediction, local regression is sometimes called a *memory-based* procedure.

It is necessary in local regression to decide whether to fit a constant, linear, or quadratic regression in the neighborhood of x_0. But even more important is how the neighborhood of x_0 is defined. The *span* s of local regression is the fraction of the training data that will be included. If the k closest points to x_0 from a total set of n are included in the local regression then it has span $s = \frac{k}{n}$. We can use cross-validation to find an optimal span or we can choose some other value directly. Small spans are easily swayed by local fluctuations in the data while large spans fit general, global trends. If the span equals 1, i.e. all data is considered in the local regression at any point, then you just get a normal regression.

A useful generalization of local regression is to choose different spans for different features in a multivariate setting. For instance, you can fit a multiple linear regression that is global for some variables (large span) and local in others, such as time. This gives the model the flexibility to adapt to more recently collected data. The coefficients are essentially allowed to change over time so this type of extension of the local regression is called a *varying coefficient model*. Local regression can also be generalized to higher dimensions, so that local variations across several variables can be modeled. However, this technique, like so many others, suffers

from the *curse of dimensionality*; beyond 3 or 4 dimensions there is often not enough data close to x_0 for a regression to be meaningful.

7.7: Generalized Additive Models

The previous methods dealt primarily with modeling the response as a function of a single predictor, though it did mention the possibility of extending the local regression to higher dimensions. This section introduces a general framework for modeling the response as a sum of functions of multiple predictors.

7.7.1 GAMs for Regression Problems

The multiple linear regression model has the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon_i$$

An easy generalization of this is to replace $\beta_j x_{ij}$ with a smooth non-linear function $f_j(x_{ij})$. Note that if $f_j(x_{ij}) = \beta_j x_{ij}$ then we're back to the multiple linear regression setting. This is an example of a generalized additive model (GAM); it is called additive because each function $f_j(x_{ij})$ is added together to get the response.

The functions $f_j(x_{ij})$ need to be determined, but the previous sections gave us lots of ways to fit a curve to a single variable. The book does not go into great detail about how exactly these curves are fit. If natural splines are used to model continuous variables, least squares can apparently be used to fit those curves to the response. If smoothing splines are used, a method called *backfitting* adjusts the curves to the response, which cycles through the predictors and updates their fit holding the other predictors fixed. The R package **gam()** takes care of this automatically.

GAMs do a good job modeling non-linear relationships in data; a standard linear regression would miss these relationships and GAMs save us from having to try a bunch of tranformations on the data to get it linear. This flexibility can yield better predictions. The additive nature of GAMs also allows for easy interpretation because we can look at the impact of each variable on the response in isolation from the other variables. GAMs additivity, however, can be a drawback because it misses important interactions between variables. If there is reason to believe some variables are related, interaction terms, $X_j \times X_k$, or even low-dimension interaction functions, $f_{jk}(X_j, X_k)$, can be included in the model.

7.7.2 GAMs for Classification Problems

Similar to the logistic regression, which has the form

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p$$

GAMs can also be used in classification problems when they take the form

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + f_1(X_1) + f_2(X_2) + ... + f_p(X_p)$$

The section concludes with an example of fitting a logistic regression GAM to some data.

ISLR Questions

7.9 a

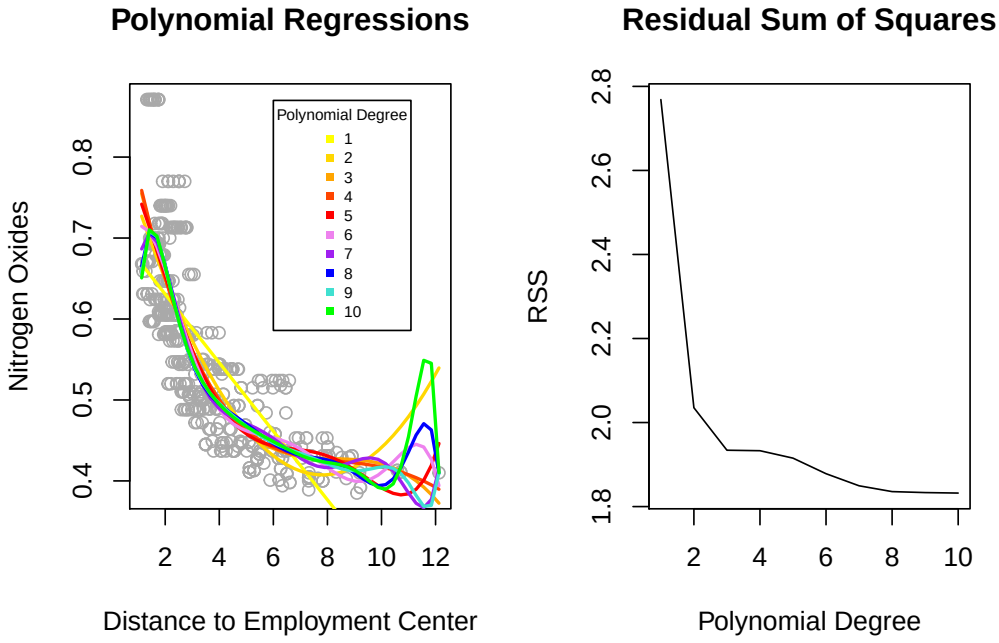
> Use the poly() function to fit a cubic polynomial regression to predict nox using dis. Report the regression output, and plot the resulting data and polynomial fits.

```
##
## Call:
## lm(formula = nox ~ poly(dis, 3, raw = TRUE), 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)                 0.9341281  0.0207076  45.110  < 2e-16 ***
## poly(dis, 3, raw = TRUE)1 -0.1820817  0.0146973 -12.389  < 2e-16 ***
## poly(dis, 3, raw = TRUE)2  0.0219277  0.0029329   7.476 3.43e-13 ***
## poly(dis, 3, raw = TRUE)3 -0.0008850  0.0001727  -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
```

Above is the summary information of a a cubic polynomial regression model fit using *nox* as the response and *dis* as the predictor from the Boston data set. Below is a plot of the resulting data and polynomial fits. It plots the *distance* against *nox* with the blue line representing the results of the model fit using least squares and the blue dotted lines showing an estimated 95% confidence interval. Instead of looking at the coefficients, we use this plot to better understand the relationship between *dis* and *nox*.

7.9 b

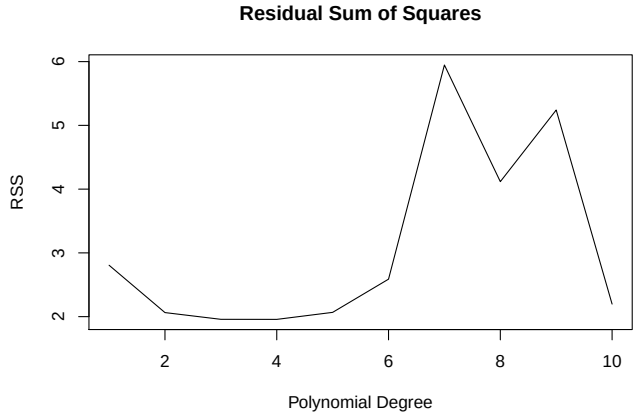
Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.



Note that the residaul sum of squares decreases as the degree of the polynomial model increases; this makes sense because more degrees give us more flexibility to fit the data but we run the risk of overfitting. Getting cross-validated residual sum of squares will show when this is the case.

7.9 c

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



The above plot shows the cross-validated residual sum of squares as a function of the polynomial degree. It looks like the optimal degree to avoid overfitting is 3 or 4, where RSS is minimized.

7.9 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.

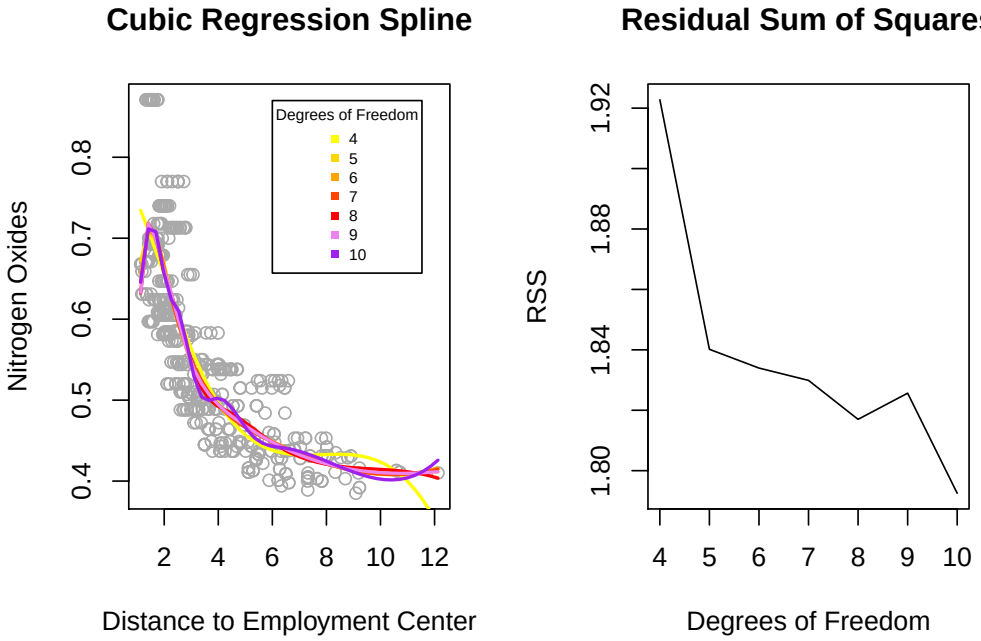
```
##
## Call:
## lm(formula = nox ~ bs(dis, df = 4))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.124622 -0.039259 -0.008514  0.020850  0.193891
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.73447    0.01460  50.306  < 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
```

```
##     50%
## 3.20745
```

Interpretation of the cubic spline's fit to nox is difficult, but it does look like the four basis functions are significant in the model. A cubic spline with four degrees of freedom can only have one knot (); bs() automatically put the knot at the 50th percentile. Below is a plot of this cubic regression spline with the single knot marked with a dotted line.

7.9 e

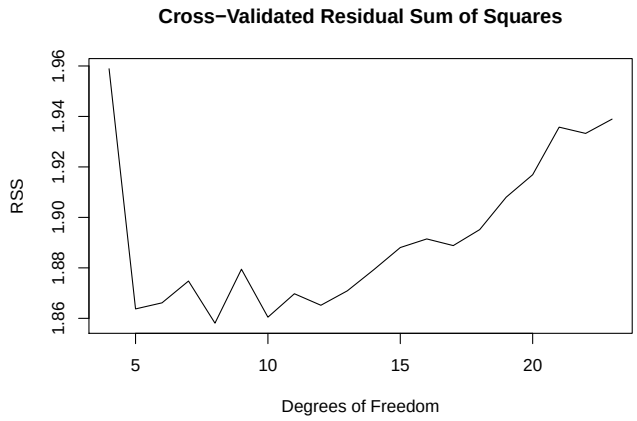
Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.



Like in part b, the residual sum of squares decreases as we add knots to our model because we have more flexibility fitting the data. However, there is a risk of overfitting, as cross-validation will show. Most of the models look similar, though, except for the yellow one, which has 4 degrees of freedom; it has weird behavior at the margins and thus yields a large RSS.

7.9 f

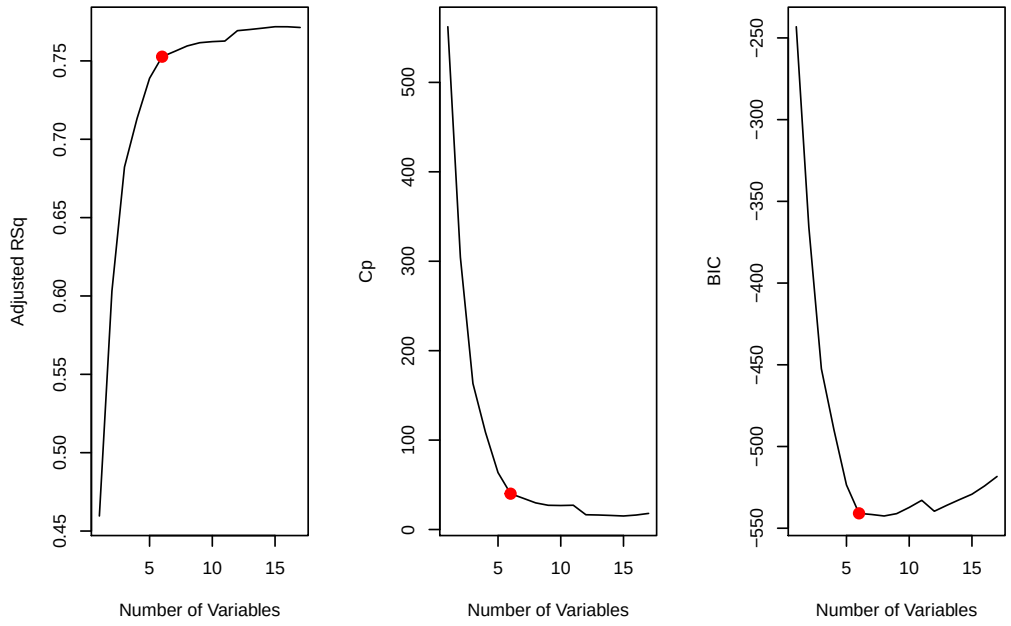
Perform cross-validation or another approach in order to select the best degrees of freedom for a regression spline on this data. Describe your results.



We weren't getting any evidence of overfitting at degrees of freedeom examined in e, so we increased our scope to 24 degrees of freedom. It is clear from the above graph that overfitting becomes an issue after 10 degrees, so that is the optimal model.

7.10 a

Split the data into a training set and a test set. Using out-of-state tuition as the response and the other variables as the predictors, perform forward stepwise selection on the training set in order to identify a satisfactory model that uses just a subset of the predictors.



Using the forward stepwise selection method the above plots were produced using C_p, Adjusted r^2, and BIC. The red dots represent the number of coefficients to be used in the best model (where the curves begin to taper off). The following subset of predictors was selected for a best fit model.

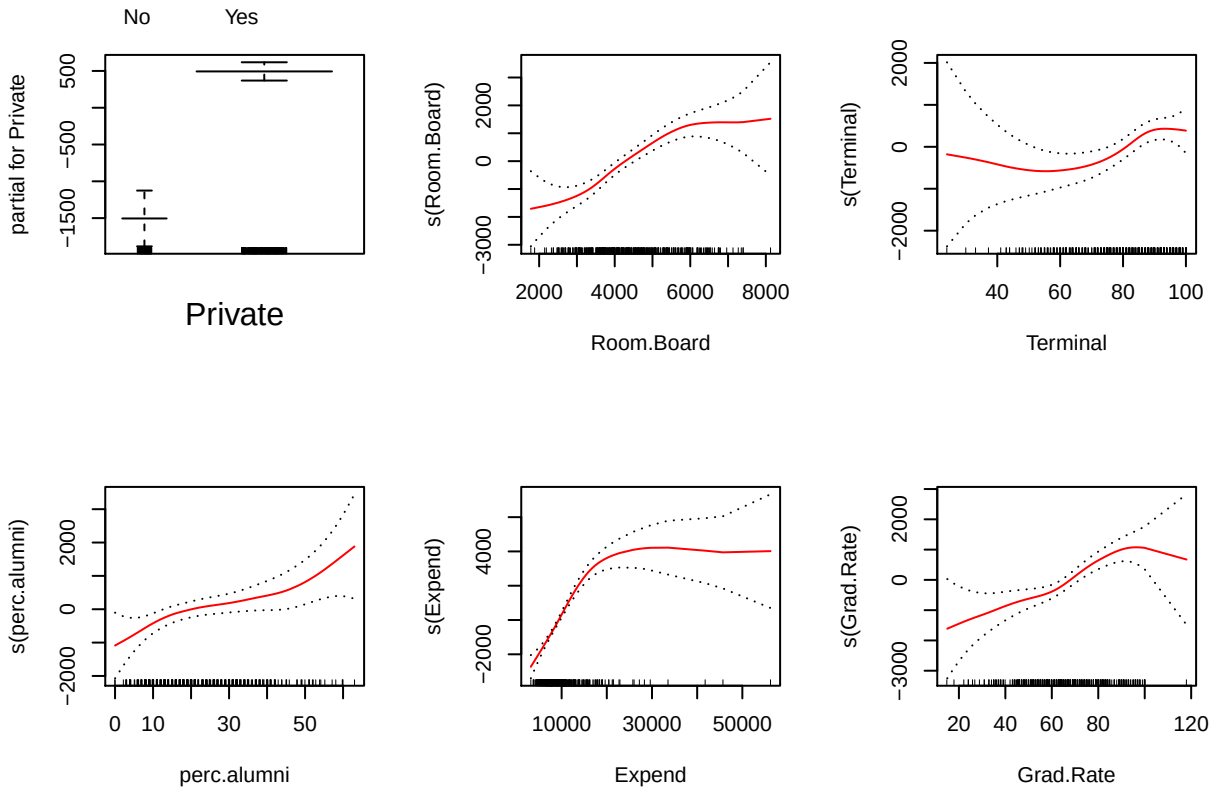
```
##   (Intercept)    PrivateYes    Room.Board           PhD   perc.alumni
## -3805.4296853  2271.7192945     1.0348276    39.7854954    45.2546798
##        Expend     Grad.Rate
##     0.1943534    34.8415421
```

7.10 b

Fit a GAM on the training data, using out-of-state tuition as the response and the features selected in the previous step as the predictors. Plot the results, and explain your findings.

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

```
## Loaded gam 1.16.1
```



Above are the plots produced after fitting a GAM on the training set, using the predictors selected in Part A. Smoothing splines have been fit to most of the variables except for Private, as it is a categorical variable. Each plot shows how the response behaves holding all other variables fixed besides the predictor represented. Looking at the first plot, it seems that out of state tuition is noticeably higher for private schools compared to non-private. Out of state tuition seems to increase for Perc.Alumni and Room.Board (both of which are the closest to being linearly related with the response). Out of state tuition seems to increase and then plateau with Expend. Finally, Out of state tuition seems to fluctuate with Grad.Rate and Terminal.

7.10 c

Evaluate the model obtained on the test set, and explain the results obtained.

Now using the testing data we are able to calculate the RSS to evaluate the model. With an RSS value of 0.8029 the model seems to fit pretty well, explaining a majority of the variation in the original data.

7.10 d

For which variables, if any, is there evidence of a non-linear relationship with the response?

```
##
## Call: gam(formula = Outstate ~ Private + s(Room.Board) + s(Terminal) +
##     s(perc.alumni) + s(Expend) + s(Grad.Rate), data = College[index,
```

```
##     ])
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -6433.03 -1004.72   -10.34  1179.02  4542.43
##
## (Dispersion Parameter for gaussian family taken to be 3143083)
##
##     Null Deviance: 6234672636 on 412 degrees of freedom
## Residual Deviance: 1228946906 on 391.0004 degrees of freedom
## AIC: 7374.21
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                  Df     Sum Sq    Mean Sq F value    Pr(>F)
## Private           1 1376476714 1376476714 437.938 < 2.2e-16 ***
## s(Room.Board)     1 1403563704 1403563704 446.556 < 2.2e-16 ***
## s(Terminal)       1  426926989  426926989 135.831 < 2.2e-16 ***
## s(perc.alumni)    1  269614661  269614661  85.780 < 2.2e-16 ***
## s(Expend)         1  508717480  508717480 161.853 < 2.2e-16 ***
## s(Grad.Rate)      1  103854343  103854343  33.042 1.818e-08 ***
## Residuals       391 1228946906    3143083
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                 Npar Df  Npar F     Pr(F)
## (Intercept)
## Private
## s(Room.Board)         3  2.5111    0.0583 .
## s(Terminal)           3  1.8688    0.1343
## s(perc.alumni)        3  0.9402    0.4212
## s(Expend)             3 22.0492 3.278e-13 ***
## s(Grad.Rate)          3  1.7645    0.1534
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can find evidence of a non-linear relationship between the predictors and the response using the summary of our GAM model, specifically the Anova for Nonparametric Effects section. Performing a hypothesis test with a null of a linear relationship vs the alternative of a non-linear relationship, we can use the Pr(F) as a p-value against our usual significance level of 0.05. Since the predictor Expend is well below this value we can reject the null and conclude it does not have a linear relationship with out of state tuition. Grad.Rate is close enough to the significance level but slightly lower which may require more investigation but for this assignment will be considered linear, along with Terminal, perc.alumni, and Room.Board. A similar conclusion can also be drawn from the plots in Part B.

Code Appendix

7.9 a

```
fit = lm(nox~poly(dis,3,raw=TRUE), data=Boston)

distlims = range(dis)
dists = seq(distlims[1],distlims[2],length.out=40)
preds = predict(fit,newdata=list(dis=dists),se=TRUE)
se.bands = cbind(preds$fit+2*preds$se.fit, preds$fit-2*preds$se.fit)

plot(dis,nox,col="darkgrey", xlab="Distance to Employment Center",
     ylab="Nitrogen Oxides", main="Cubic Regression")
lines(dists,preds$fit,lwd=2,col="blue")
matlines(dists,se.bands,lwd=1,col="blue",lty=3)
```

7.9 b

```
par(mfrow=c(1,2))
plot(dis,nox,col="darkgrey", xlab="Distance to Employment Center",
     ylab="Nitrogen Oxides", main="Polynomial Regressions")
cols = c("yellow","gold","orange","orangered","red","violet","purple","blue","turquoise","green")
rss = rep(0,10)
for(i in 1:10){
  fit = lm(nox~poly(dis,i,raw=TRUE), data=Boston)
  preds = predict(fit,newdata=list(dis=dists))
  lines(dists,preds,lwd=2,col=cols[i])
  rss[i] = sum(fit$residuals^2)
}
legend(x = 6, y = 0.87, legend =c(1:10), col =cols, cex=0.6, pch = 15,
       title="Polynomial Degree")

plot(seq(1:10),rss,type="l",main="Residual Sum of Squares",
     xlab="Polynomial Degree",ylab="RSS")
```

7.9 c

```
n = length(dis)

test.index = vector("list",10)
temp = sample(1:n,n,replace=FALSE)
for (i in 1:9){
  test.index[[i]] = temp[((i-1)*trunc(n/10)+1):(i*trunc(n/10))]
}
test.index[[10]] = temp[(9*trunc(n/10)+1):n]

rss = rep(0,10)

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

```
    fit = lm(nox~poly(dis,j,raw=TRUE), data=Boston[-test.index[[i]],])
    preds = predict(fit,newdata=Boston[test.index[[i]],])
    rss[j] = rss[j] + sum((nox[test.index[[i]]]-preds)^2)
  }
}

plot(seq(1:10),rss,type="l",main="Residual Sum of Squares",
     xlab="Polynomial Degree",ylab="RSS")
```

7.9 d

```
fit = lm(nox~bs(dis,df=4))
preds = predict(fit,newdata=list(dis=dists),se=TRUE)
plot(dis,nox,col="darkgrey", xlab="Distance to Employment Center",
     ylab="Nitrogen Oxides", main="Cubic Spline Regression")
lines(dists,preds$fit,lwd=2)
lines(dists,preds$fit-2*preds$se.fit, lty="dashed")
lines(dists,preds$fit+2*preds$se.fit, lty="dashed")
abline(v = attr(bs(dis,df=4),"knots"), col = "red", lty = 2)

summary(fit)

attr(bs(dis,df=4),"knots")
```

7.9 e

```
rss = rep(0,7)

par(mfrow=c(1,2))
plot(dis,nox,col="darkgrey", xlab="Distance to Employment Center",
     ylab="Nitrogen Oxides", main="Cubic Regression")
for (i in 1:7){
  df = i+3
  fit = lm(nox~bs(dis,df=df))
  preds = predict(fit,newdata=list(dis=dists),se=TRUE)
  lines(dists,preds$fit,lwd=2,col=cols[i])
  rss[i] = sum(fit$residuals^2)
}
plot(seq(4,10),rss,type="l",main="Residual Sum of Squares",
     xlab="Degrees of Freedom",ylab="RSS")
```

7.9 f

```
rss = rep(0,20)

for (j in 1:20){
  df = j+3
  for (i in 1:10){
    fit = lm(nox~bs(dis,df=df), data=Boston[-test.index[[i]],])
```

```
    preds = predict(fit, newdata=Boston[test.index[[i]],])
    rss[j] = rss[j] + sum((nox[test.index[[i]]]-preds)^2)
  }
}
plot(seq(4,23),rss,type="l",main="Cross-Validated Residual Sum of Squares",
     xlab="Degrees of Freedom",ylab="RSS")
```

7.10 a

```
library(ISLR)
library(leaps)
data = data.frame(model.matrix(~., data = College)[,-1])
attach(data)

n = dim(data)[1]
index = sample(c(TRUE, FALSE), n, replace = T, prob = c(0.5, 0.5))
train = data[index,]
test = data[!index,]

forward=regsubsets(Outstate~., data = train, method = 'forward', nvmax = 18)
fwd_sum = summary(forward)

par(mfrow = c(1,3))
plot(fwd_sum$adjr2 ,xlab ="Number of Variables", ylab="Adjusted RSq",type="l")
points (6, fwd_sum$adjr2[6], col ="red",cex =2, pch =20)

plot(fwd_sum$cp ,xlab ="Number of Variables", ylab="Cp",type="l")
points (6, fwd_sum$cp[6], col ="red",cex =2, pch =20)

plot(fwd_sum$bic ,xlab ="Number of Variables", ylab="BIC",type="l")
points (6, fwd_sum$bic[6], col ="red",cex =2, pch =20)
```

7.10 b

```
library(gam)
gam1 = gam(Outstate~Private+s(Room.Board)+s(Terminal)+s(perc.alumni)+s(Expend)+s(Grad.Rate),
           data = College[index,])
par(mfrow = c(2,3))
plot(gam1, se = TRUE, col = "red")
```

7.10 c

```
preds = predict(gam1, newdata = College[index,])

sse = sum((preds - College[index,9])^2)
sst = sum((mean(College[index,9]) - College[index,9])^2)
rss = 1 - sse/sst
```

7.10 d

```
summary(gam1)
```