Notes on Statistical Learning

John I. Marden

Contents

Chapter 1

Introduction

These notes are based on a course in statistical learning using the text **The Elements of Statistical Learning** by Hastie, Tibshirani and Friedman (2001) (The first edition). Hence, everything throughout these pages implicitly uses that book as a reference. So keep a copy handy! But everything here is my own interpretation.

What is machine learning?

> In artificial intelligence, machine learning involves some kind of machine (robot, computer) that modifies its behavior based on experience. For example, if a robot falls down every time it comes to a stairway, it will learn to avoid stairways. E-mail programs often learn to distinguish spam from regular e-mail.
>
> In statistics, machine learning uses statistical data to learn. Generally, there are two categories:
>
> **Supervised learning** data consists of example (y, x)'s, the training data. The machine is a function built based on the data that takes in a new x, and produces a guess of the corresponding y. It is **prediction** if the y's are continuous, and **classification** or **categorization** if the y's are categories.
>
> **Unsupervised learning** is **clustering**. The data consists of example x's, and the machine is a function that groups the x's into clusters.

What is data mining?

> Looking for relationships in large data sets. Observations are "baskets" of items. The goal is to see what items are associated with other items, or which items' presence implies the presence of other items. For example, at Walmart, one may realize that people who buy socks also buy beer. Then Walmart would be smart to put some beer cases near the socks, or vice versa. Or if the government is spying on everyone's e-mails, certain words (which I better not say) found together might cause the writer to be sent to Guantanamo.

The difference for a statistician between supervised machine learning and regular data analysis is that in machine learning, the statistician does not care about the estimates of parameters nor hypothesis tests nor which models fit best. Rather, the focus is on finding some function that does a good job of predicting y from x. Estimating parameters, fitting models, etc., may indeed be important parts of developing the function, but they are not the objective.

Chapter 2

Linear models

To ease into machine learning, we start with regular linear models. There is one dependent variable, the y, and p explanatory variables, the x's. The data, or training sample, consists of n independent observations:

$$(y_1, \underline{x}_1), (y_2, \underline{x}_2), \ldots, (y_N, \underline{x}_N). \tag{2.1}$$

For individual i, y_i is the value of the one-dimensional dependent variable, and

$$\underline{x}_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix} \tag{2.2}$$

is the $p \times 1$ vector of values for the explanatory variables. Generally, the y_i's are continuous, but the x_{ij}'s can be anything numerical, e.g., 0-1 indicator variables, or functions of another variable (e.g., x, x^2, x^3).

The linear model is

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + e_i. \tag{2.3}$$

The β_j's are parameters, usually unknown and to be estimated. The e_i's are the errors or residuals. We will assume that

- The e_i's are independent (of each other, and of the $\underline{x}_i$'s);
- $E[e_i] = 0$ for each i;
- $Var[e_i] = \sigma_e^2$ for each i.

There is also a good chance we will assume they are normally distributed.

From STAT424 and 425 (or other courses), you know what to do now: estimate the β_j's and σ_e^2, decide which β_j's are significant, do F-tests, look for outliers and other violations of the assumptions, etc.

Here, we may do much of that, but with the goal of prediction. Suppose $(y^{New}, \underline{x}^{New})$ is a new point, satisfying the same model and assumptions as above (in particular, being independent of the observed $\underline{x}_i$'s). Once we have the estimates of the β_i's (based on the observed data), we **predict** y^{New} from $\underline{x}^{New}$ by

$$\widehat{y}^{New} = \widehat{\beta}_0 + \widehat{\beta}_1 x_1^{new} + \cdots + \widehat{\beta}_p x_p^{new}. \tag{2.4}$$

The prediction is good if $\widehat{y}^{New}$ is close to y^{New}. We do not know y^{New}, but we can hope. But the key point is

> The estimates of the parameters are good if they give good predictions. We don't care if the $\widehat{\beta}_j$'s are close to the β_j's; we don't care about unbiasedness or minimum variance or significance. We just care whether we get good predictions.

2.1 Good predictions: Squared error loss and in-sample error

We want the predictions to be close to the actual (unobserved) value of the dependent variable, that is, we want $\widehat{y}^{New}$ close to y^{New}. One way to measure closeness is by using squared error:

$$(y^{New} - \widehat{y}^{New})^2. \tag{2.5}$$

Because we do not know y^{New} (yet), we might look at the expected value instead:

$$E[(Y^{New} - \widehat{Y}^{New})^2]. \tag{2.6}$$

But what is that the expected value over? Certainly Y^{New}, but the Y_i's and $\underline{X}_i$'s in the sample, as well as the $\underline{X}^{New}$, could all be considered random. There is no universal answer, but for our purposes here we will assume that the $\underline{x}_i$'s are fixed, and all the Y_i's are random.

The next question is what to do about $\underline{x}^{New}$? If you have a particular $\underline{x}^{New}$ in mind, then use that:

$$E[(Y^{New} - \widehat{Y}^{New})^2 \mid \underline{X}_1 = \underline{x}_1, \ldots, \underline{X}_N = \underline{x}_N, \underline{X}^{New} = \underline{x}^{New}]. \tag{2.7}$$

But typically you are creating a predictor for many new x's, and likely you do not know what they will be. (You don't know what the next 1000 e-mails you get will be.) A reasonable approach is to assume the new x's will look much like the old ones, hence you would look at the errors for N new $\underline{x}_i$'s being the same as the old ones. Thus we would have N new cases, $(\underline{y}_i^{New}, \underline{x}_i^{New})$, but where $\underline{x}_i^{New} = \underline{x}_i$. The N expected errors are averaged, to obtain what is called the **in-sample error**:

$$ERR_{in} = \frac{1}{N} \sum_{i=1}^{N} E[(Y_i^{New} - \widehat{Y}_i^{New})^2 \mid \underline{X}_1 = \underline{x}_1, \ldots, \underline{X}_N = \underline{x}_N, \underline{X}_i^{New} = \underline{x}_i^{New}]. \tag{2.8}$$

In particular situations, you may have a more precise knowledge of what the new x's would be. By all means, use those values.

We will drop the conditional part of the notation for simplicity.

2.2 Matrices and least-squares estimates

Ultimately we want to find estimates of the parameters that yield a low ERR_{in}. We'll start with the least squares estimate, then translate things to matrices. The estimates of the β_j's depends on just the training sample. The **least squares** estimate of the parameters are the b_j's that minimize the objective function

$$obj(b_0, \ldots, b_p) = \sum_{i=1}^{N} (y_i - b_0 - b_1 x_{i1} - \cdots - b_p x_{ip})^2. \tag{2.9}$$

The function is a nice convex function in the b_j's, so setting the derivatives equal to zero and solving will yield the minimum. The derivatives are

$$\begin{aligned} \frac{\partial}{\partial b_0}\, obj(b_0, \ldots, b_p) &= -2 \sum_{i=1}^{N} (y_i - b_0 - b_1 x_{i1} - \cdots - b_p x_{ip}); \\ \frac{\partial}{\partial b_j}\, obj(b_0, \ldots, b_p) &= -2 \sum_{i=1}^{N} x_{ij} (y_i - b_0 - b_1 x_{i1} - \cdots - b_p x_{ip}), j \geq 1. \end{aligned} \tag{2.10}$$

Write the equations in matrix form, staring with

$$\begin{aligned} \begin{pmatrix} y_1 - b_0 - b_1 x_{11} - \cdots - b_p x_{1p} \\ y_2 - b_0 - b_1 x_{21} - \cdots - b_p x_{2p} \\ \vdots \\ y_N - b_0 - b_1 x_{N1} - \cdots - b_p x_{Np} \end{pmatrix} &= \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} - \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Np} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_p \end{pmatrix} \\ &\equiv \underline{y} - \mathbf{X}\underline{b}. \end{aligned} \tag{2.11}$$

Take the two summations in equations (2.10) (without the -2's) and set to 0 to get

$$\begin{aligned} (1, \cdots, 1)(\underline{y} - \mathbf{X}\underline{b}) &= 0; \\ (x_{1j}, \cdots, x_{Nj})(\underline{y} - \mathbf{X}\underline{b}) &= 0, \quad j \geq 1. \end{aligned} \tag{2.12}$$

Note that the vectors in (2.12) on the left are the columns of $\mathbf{X}$ in (2.11), yielding

$$\mathbf{X}'(\underline{y} - \mathbf{X}\underline{b}) = 0. \tag{2.13}$$

That equation is easily solved:

$$\mathbf{X}'\underline{y} = \mathbf{X}'\mathbf{X}\underline{b} \quad \Rightarrow \quad \underline{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\underline{y}, \tag{2.14}$$

at least if $\mathbf{X}'\mathbf{X}$ is invertible. If it is not invertible, then there will be many solutions. In practice, one can always eliminate some (appropriate) columns of $\mathbf{X}$ to obtain invertibility. Generalized inverses are available, too.

Summary. In the linear model

$$\underline{Y} = \mathbf{X}\underline{\beta} + \underline{e}, \tag{2.15}$$

where

$$\underline{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \quad \text{and} \quad \underline{e} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{pmatrix}, \tag{2.16}$$

the least squares estimate of $\underline{\beta}$, assuming $\mathbf{X}'\mathbf{X}$ is invertible, is

$$\widehat{\underline{\beta}}_{LS} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\underline{y}. \tag{2.17}$$

2.3 Mean vectors and covariance matrices

Before calculating the ERR_{in}, we need some matrix results. If $\underline{Z}$ is a vector or $\mathbf{Z}$ is a matrix, then so is its mean:

$$\text{Vector:} \quad E[\underline{Z}] = E\left[\begin{pmatrix} Z_1 \\ \vdots \\ Z_K \end{pmatrix}\right] = \begin{pmatrix} E[Z_1] \\ \vdots \\ E[Z_K] \end{pmatrix}; \tag{2.18}$$

$$\text{Matrix:} \quad E[\mathbf{Z}] = E\left[\begin{pmatrix} Z_{11} & Z_{12} & \cdots & Z_{1L} \\ Z_{21} & Z_{22} & \cdots & Z_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{K1} & Z_{K2} & \cdots & Z_{KL} \end{pmatrix}\right] = \begin{pmatrix} E[Z_{11}] & E[Z_{12}] & \cdots & E[Z_{1L}] \\ E[Z_{21}] & E[Z_{22}] & \cdots & E[Z_{2L}] \\ \vdots & \vdots & \ddots & \vdots \\ E[Z_{K1}] & E[Z_{K2}] & \cdots & E[Z_{KL}] \end{pmatrix} \tag{2.19}$$

Turning to variances and covariances, suppose that $\mathbf{Z}$ is a $K \times 1$ vector. There are K variances and $\binom{K}{2}$ covariances among the Z_j's to consider, recognizing that $\sigma_{jk} = Cov[Z_j, Z_k] = Cov[Z_k, Z_j]$. By convention, we will arrange them into a matrix, the **variance-covariance matrix**, or simply **covariance matrix** of $\mathbf{Z}$:

$$\mathbf{\Sigma} = Cov[\mathbf{Z}] = \begin{pmatrix} Var[Z_1] & Cov[Z_1, Z_2] & \cdots & Cov[Z_1, Z_K] \\ Cov[Z_2, Z_1] & Var[Z_2] & \cdots & Cov[Z_2, Z_K] \\ \vdots & \vdots & \ddots & \vdots \\ Cov[Z_K, Z_1] & Cov[Z_K, Z_2] & \cdots & Var[Z_K] \end{pmatrix}, \tag{2.20}$$

so that the elements of $\mathbf{\Sigma}$ are the σ_{jk}'s.

The means and covariance matrices for linear (or affine) transformations of vectors and matrices work as for univariate variables. Thus if $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are fixed matrices of suitable sizes, then

$$E[\mathbf{AZB} + \mathbf{C}] = \mathbf{A}E[\mathbf{Z}]\mathbf{B} + \mathbf{C}. \tag{2.21}$$

For a vector $\underline{Z}$,

$$Cov[\mathbf{A}\underline{Z} + \underline{c}] = \mathbf{A}\underline{Z}\mathbf{A}'. \tag{2.22}$$

The $\underline{c}$ is constant, so does not add to the variance.

Because sums of squares are central to our analyses, we often need to find

$$E[\|\underline{Z}\|^2], \tag{2.23}$$

where for $K \times 1$ vector $\underline{z}$,

$$\|\underline{z}\|^2 = \underline{z}'\underline{z} = z_1^2 + \cdots + z_K^2 \tag{2.24}$$

is the squared norm. Using the fact that $Var[W] = E[W^2] - E[W]^2$,

$$\begin{aligned} E[\|\underline{Z}\|^2] &= E[Z_1^2 + \cdots + Z_K^2] \\ &= E[Z_1^2] + \cdots + E[Z_K^2] \\ &= E[Z_1]^2 + Var[Z_1]^2 + \cdots + E[Z_K]^2 + Var[Z_K]^2 \\ &= \|E[\underline{Z}]\|^2 + trace(Cov[\underline{Z}]), \end{aligned} \tag{2.25}$$

because the trace of a matrix is the sum of the diagonals, which in the case of a covariance matrix are the variances.

2.4 Prediction using least-squares

When considering the in-sample error for the linear model, we have the same model for the training sample and the new sample:

$$\underline{Y} = \mathbf{X}\underline{\beta} + \underline{e} \text{ and } \underline{Y}^{New} = \mathbf{X}\underline{\beta} + \underline{e}^{New}. \tag{2.26}$$

The e_i's and e_i^{New}'s are independent with mean 0 and variance σ_e^2. If we use the least-squares estimate of $\underline{\beta}$ in the prediction, we have

$$\widehat{\underline{Y}}^{New} = \mathbf{X}\widehat{\underline{\beta}}_{LS} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\underline{Y} = \mathbf{H}\underline{Y}, \tag{2.27}$$

where $\mathbf{H}$ is the "hat" matrix,

$$\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'. \tag{2.28}$$

Note that this matrix is idempotent, which means that

$$\mathbf{HH} = \mathbf{H}. \tag{2.29}$$

The errors in prediction are the $Y_i^{New} - \widehat{Y}_i^{New}$. Before getting to the ERR_{in}, consider the mean and covariance's of these errors. First,

$$E[\underline{Y}] = E[\mathbf{X}\underline{\beta} + \underline{e}] = \mathbf{X}\underline{\beta}, \quad E[\underline{Y}^{New}] = E[\mathbf{X}\underline{\beta} + \underline{e}^{New}] = \mathbf{X}\underline{\beta}, \tag{2.30}$$

because the expected values of the e's are all 0 and we are assuming $\mathbf{X}$ is fixed, and

$$E[\widehat{\underline{Y}}^{New}] = E[\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\underline{Y}] = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'E[\underline{Y}] = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}\underline{\beta} = \mathbf{X}\underline{\beta}, \tag{2.31}$$

because the $\mathbf{X}'\mathbf{X}$'s cancel. Thus,

$$E[\underline{Y}^{New} - \widehat{\underline{Y}}^{New}] = \underline{0}_N \text{ (the } N \times 1 \text{ vector of 0's).} \tag{2.32}$$

This zero means that the errors are **unbiased**. They may be big or small, but on average right on the nose. Unbiasedness is ok, but it is really more important to be close.

Next, the covariance matrices:

$$Cov[\underline{Y}] = Cov[\mathbf{X}\underline{\beta} + \underline{e}] = Cov[\underline{e}] = \sigma_e^2 \mathbf{I}_N \text{ (the } N \times N \text{ identity matrix),} \tag{2.33}$$

because the e_i's are independent, hence have zero covariance, and all have variance σ_e^2. Similarly,

$$Cov[\underline{Y}^{New}] = \sigma_e^2 \mathbf{I}_N. \tag{2.34}$$

Less similar,

$$\begin{aligned} Cov[\widehat{\underline{Y}}^{New}] &= Cov[\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\underline{Y}] \\ &= \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'Cov[\underline{Y}]\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' \quad \text{(Eqn. (2.22))} \\ &= \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\sigma_e^2\mathbf{I}_N\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' \\ &= \sigma_e^2\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' \\ &= \sigma_e^2\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' \\ &= \sigma_e^2\mathbf{H}. \end{aligned} \tag{2.35}$$

Finally, for the errors, note that $\underline{Y}^{new}$ and $\widehat{\underline{Y}}^{New}$ are independent, because the latter depends on the training sample alone. Hence,

$$\begin{aligned} Cov[\underline{Y}^{New} - \widehat{\underline{Y}}^{New}] &= Cov[\underline{Y}^{new}] + Cov[\widehat{\underline{Y}}^{New}] \text{ (notice the +)} \\ &= \sigma_e^2\mathbf{I}_N + \sigma_e^2\mathbf{H} \\ &= \sigma_e^2(\mathbf{I}_N + \mathbf{H}). \end{aligned} \tag{2.36}$$

Now,

$$\begin{aligned} N \cdot ERR_{in} &= E[\|\underline{Y}^{New} - \widehat{\underline{Y}}^{New}\|^2] \\ &= \|E[\underline{Y}^{New} - \widehat{\underline{Y}}^{New}]\|^2 + trace(Cov[\underline{Y}^{New} - \widehat{\underline{Y}}^{New}]) \text{ (by (2.25))} \\ &= trace(\sigma_e^2(\mathbf{I}_N + \mathbf{H})) \quad \text{(by (2.36) and (2.32))} \\ &= \sigma_e^2(N + trace(\mathbf{H})). \end{aligned} \tag{2.37}$$

For the trace, recall that $\mathbf{X}$ is $N \times (p+1)$, so that

$$trace(\mathbf{H}) = trace(\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}') = trace((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}) = trace(\mathbf{I}_{p+1}) = p+1. \tag{2.38}$$

Putting that answer in (2.37) we obtain

$$ERR_{in} = \sigma_e^2 + \sigma_e^2 \frac{p+1}{N}. \tag{2.39}$$

This expected in-sample error is a simple function of three quantities. We will use it as a benchmark. The goal in the rest of this section will be to find, if possible, predictors that have lower in-sample error.

There's not much we can do about σ_e^2, since it is the inherent variance of the observations. Taking a bigger training sample will decrease the error, as one would expect. The one part we can work with is the p, that is, try to reduce p by eliminating some of the explanatory variables. Will that strategy work? It is the subject of the next subsection.

2.5 Subset selection and Mallows' C_p

In fitting linear models (or any models), one often looks to see whether some of the variables can be eliminated. The purpose is to find a simpler model that still fits. Testing for the significance of the β_j's, either singly or in groups (e.g., testing for interaction in analysis of variance), is the typical approach. Here, we will similarly try to eliminate variables, but based on predictive power, not significance.

We will use data on diabetes patients to illustrate.[1] There are $N = 442$ patients, and $p = 10$ baseline measurements, which are the predictors. The dependent variable is a measure of the progress of the disease one year after the baseline measurements were taken. The ten predictors include age, sex, BMI[2], blood pressure, and six blood measurements (hdl, ldl, glucose, etc.) denoted `S1, ..., S6`. The prediction problem is to predict the progress of the disease for the next year based on these measurements.

Below is the output for the least squares fit of the linear model:

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -334.56714   67.45462  -4.960 1.02e-06 ***
AGE           -0.03636    0.21704  -0.168 0.867031
SEX          -22.85965    5.83582  -3.917 0.000104 ***
BMI            5.60296    0.71711   7.813 4.30e-14 ***
BP             1.11681    0.22524   4.958 1.02e-06 ***
S1            -1.09000    0.57333  -1.901 0.057948 .
S2             0.74645    0.53083   1.406 0.160390
S3             0.37200    0.78246   0.475 0.634723
S4             6.53383    5.95864   1.097 0.273459
S5            68.48312   15.66972   4.370 1.56e-05 ***
S6             0.28012    0.27331   1.025 0.305990
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 54.15 on 431 degrees of freedom
```

[1] The data can be found at `http://www-stat.stanford.edu/~hastie/Papers/LARS/`

[2] Body mass index, which is $703 \times (Weight)/(Height)^2$, if weight is in pounds and height in inches. If you are 5'5", your BMI is just your weight in pounds divided by 6.

```
Multiple R-Squared: 0.5177,     Adjusted R-squared: 0.5066
F-statistic: 46.27 on 10 and 431 DF,  p-value: < 2.2e-16
```

The "Estimate"'s are the $\widehat{\beta}_j$'s. It looks like there are some variables that could be dropped from the model, e.g., Age and S3. which have very small t-statistics. Some others appear to be very necessary, namely Sex, BMI, BP and S5. It is not as clear for the other variables. Which subset should we use? The one that yields the lowest in-sample error. The in-sample error for the predictor using all 10 variables is given in (2.39). What about the predictor that uses eight variables, leaving out age and S3? Or the one that uses just BMI and BP?

For any subset of variables, one can in principle calculate the in-sample error. Remember that the error has to be calculated assuming that the full model is true, even though the estimates are based on the smaller model. That is, equation (2.26) is still true, with the complete $\mathbf{X}$. Let $\mathbf{X}^*$ denote the matrix with just the columns of x-variables that are to be used in the prediction, and let p^* be the number of such variables. Using least squares on just $\mathbf{X}^*$, and using those estimates to predict $\underline{Y}^{New}$, we obtain as in (2.27) and (2.28) the prediction

$$\widehat{\underline{Y}}^{New*} = \mathbf{H}^*\underline{Y}, \quad \text{where} \quad \mathbf{H}^* = \mathbf{X}^*(\mathbf{X}^{*\prime}\mathbf{X}^*)^{-1}\mathbf{X}^{*\prime}. \tag{2.40}$$

The calculations proceed much as in Section 2.4. For the bias,

$$E[\widehat{\underline{Y}}^{New*}] = \mathbf{H}^*E[\underline{Y}] = \mathbf{H}^*\mathbf{X}\underline{\beta}, \tag{2.41}$$

hence

$$E[\underline{Y}^{New} - \widehat{\underline{Y}}^{New*}] = \mathbf{X}\underline{\beta} - \mathbf{H}^*\mathbf{X}\underline{\beta} = (\mathbf{I}_N - \mathbf{H}^*)\underline{\beta}. \tag{2.42}$$

Notice that there may be bias, unlike in (2.32). In fact, the predictor is unbiased if and only if the β_j's corresponding to the explanatory variables left out are zero. That is, there is no bias if the variables play no role in the true model. The covariance is

$$Cov[\widehat{\underline{Y}}^{New*}] = \mathbf{H}^*Cov[\underline{Y}]\mathbf{H}^* = \sigma_e^2\mathbf{H}^*, \tag{2.43}$$

so by independent of the new and the old,

$$Cov[\underline{Y}^{New} - \widehat{\underline{Y}}^{New*}] = Cov[\underline{Y}^{New}] + Cov[\widehat{\underline{Y}}^{New*}] = \sigma_e^2(\mathbf{I}_N + \mathbf{H}^*). \tag{2.44}$$

Denoting the in-sample error for this predictor by ERR_{in}^*, we calculate

$$\begin{aligned} N \cdot ERR_{in}^* &= E[\|\underline{Y}^{New} - \widehat{\underline{Y}}^{New*}\|^2] \\ &= \|\mathbf{X}\underline{\beta} - \mathbf{H}^*\mathbf{X}\underline{\beta}\|^2 + trace(\sigma_e^2(\mathbf{I}_N + \mathbf{H}^*)) \\ &= \underline{\beta}'\mathbf{X}'(\mathbf{I}_N - \mathbf{H}^*)\mathbf{X}\underline{\beta} + \sigma_e^2(N + trace(\mathbf{H}^*)). \end{aligned} \tag{2.45}$$

This time, $trace(\mathbf{H}^*) = p^*$. Compare this error to the one using all x's in (2.39):

$$\begin{aligned} ERR_{in}^* &= \tfrac{1}{N}\,\underline{\beta}'\mathbf{X}'(\mathbf{I}_N - \mathbf{H}^*)\mathbf{X}\underline{\beta} + \sigma_e^2 + \sigma_e^2\,\tfrac{p^*+1}{N}; \\ ERR_{in} &= 0 + \sigma_e^2 + \sigma_e^2\,\tfrac{p+1}{N}; \\ \text{Error} &= \text{Bias}^2 + \text{Inherent variance} + \text{Estimation variance}. \end{aligned} \tag{2.46}$$

The lesson is that by reducing the number of variables used for prediction, you can lower the variance, but at the risk of increasing the bias. Thus there is a bias/variance tradeoff. To quantify the tradeoff, in order to see whether it is a worthwhile one, one would need to know $\underline{\beta}$ and σ_e^2, which are unknown parameters. The next best choice is to estimate these quantities, or more directly, estimate the in-sample errors.

2.5.1 Estimating the in-sample errors

For machine learning in general, a key task is estimation of the prediction error. Chapter 7 in the text reviews a number of approaches. In the present situation, the estimation is not too difficult, although one must be careful.

If we observed the $\underline{Y}^{New}$, we could estimate the in-sample error directly:

$$\widehat{ERR}_{in}^* = \frac{1}{N} \, \|\underline{y}^{New} - \widehat{\underline{y}}^{New*}\|^2. \tag{2.47}$$

(Of course, if we knew the new observations, we would not need to do any prediction.) We do observe the training sample, which we assume has the same distribution as the new vector, hence a reasonable estimate would be the **observed error**,

$$\overline{err}^* = \frac{1}{N} \, \|\underline{y} - \widehat{\underline{y}}^{New*}\|^2. \tag{2.48}$$

Is this estimate a good one? We can find its expected value, much as before.

$$E[\overline{err}^*] = \frac{1}{N}(\|E[\underline{Y} - \widehat{\underline{Y}}^{New*}]\|^2 + trace(Cov[\underline{Y} - \widehat{\underline{Y}}^{New*}])). \tag{2.49}$$

We know that $E[\underline{Y}] = \mathbf{X}\underline{\beta}$, so using (2.41) and (2.42), we have that

$$\|E[\underline{Y} - \widehat{\underline{Y}}^{New*}]\|^2 = \|(\mathbf{I}_N - \mathbf{H}^*)\mathbf{X}\underline{\beta}\|^2 = \underline{\beta}'\mathbf{X}'(\mathbf{I}_N - \mathbf{H}^*)\mathbf{X}\underline{\beta}. \tag{2.50}$$

For the covariance, we note that $\underline{Y}$ and $\widehat{\underline{Y}}^{New*}$ are *not* independent, since they are both based on the training sample. Thus

$$\begin{aligned} Cov[\underline{Y} - \widehat{\underline{Y}}^{New*}] &= Cov[\underline{Y} - \mathbf{H}^*\underline{Y}] \\ &= Cov[(\mathbf{I}_N - \mathbf{H}^*)\underline{Y}] \\ &= (\mathbf{I}_N - \mathbf{H}^*)Cov[\underline{Y}](\mathbf{I}_N - \mathbf{H}^*)' \\ &= \sigma_e^2(\mathbf{I}_N - \mathbf{H}^*), \end{aligned} \tag{2.51}$$

and

$$trace(Cov[\underline{Y} - \widehat{\underline{Y}}^{New*}]) = \sigma_e^2(N - p^* - 1). \tag{2.52}$$

Putting (2.50) and (2.52) into (2.49) yields the answer. Comparing to the actual ERR_{in}^*:

$$\begin{aligned} E[\overline{err}^*] &= \tfrac{1}{N} \, \underline{\beta}'\mathbf{X}'(\mathbf{I}_N - \mathbf{H}^*)\mathbf{X}\underline{\beta} \;+\; \sigma_e^2 \;-\; \sigma_e^2 \, \tfrac{p^*+1}{N}; \\ ERR_{in}^* &= \tfrac{1}{N} \, \underline{\beta}'\mathbf{X}'(\mathbf{I}_N - \mathbf{H}^*)\mathbf{X}\underline{\beta} \;+\; \sigma_e^2 \;+\; \sigma_e^2 \, \tfrac{p^*+1}{N}. \end{aligned} \tag{2.53}$$

These equations reveal an important general phenomenon:

The error in predicting the observed data is an underestimate of the error in predicting the new individuals.

This effect should not be too surprising, as the predictor is chosen in order to do well with the observed data.

To estimate the ERR^*_{in}, we have to bump up the observed error a bit. From (2.53),

$$ERR^*_{in} = E[\overline{err}^*] + 2\sigma_e^2 \, \frac{p^*+1}{N}. \tag{2.54}$$

If we can estimate σ_e^2, then we will be all set. The usual regression estimate is the one to use,

$$\widehat{\sigma}_e^2 = \frac{\|\underline{y} - \underline{\widehat{y}}\|^2}{N-p-1} = \frac{\text{Residual Sum of Squares}}{N-p-1}. \tag{2.55}$$

Here, $\underline{\widehat{y}} = \mathbf{X}\underline{\widehat{\beta}}_{LS}$, which is the same as $\underline{\widehat{y}}^{New}$ (2.27) when using all the predictors. The estimate is unbiased,

$$E[\widehat{\sigma}_e^2] = \sigma_e^2, \tag{2.56}$$

because

$$E[\underline{Y} - \underline{\widehat{Y}}^{New}] = (\mathbf{I}_N - \mathbf{H})\mathbf{X}\underline{\beta} = 0 \;\; (\text{because } \mathbf{HX} = \mathbf{X}), \tag{2.57}$$

and

$$trace(Cov[\underline{Y} - \underline{\widehat{Y}}^{New}]) = \sigma_e^2 trace(\mathbf{I}_N - \mathbf{H}) = \sigma_e^2(N-p-1). \tag{2.58}$$

So from (2.54),

$$\widehat{ERR}^*_{in} = \overline{err}^* + 2\widehat{\sigma}_e^2 \, \frac{p^*+1}{N} \tag{2.59}$$

is an unbiased estimator for ERR^*_{in}. (Just to emphasize: The first term is the error calculating using the subset of the explanatory variables under consideration, while the second term is the error using the full set of predicators.) One way to think of this estimate is as follows:

$$\text{Prediction error} = \text{Observed error} + \text{Penalty for estimating parameters.} \tag{2.60}$$

Turn to the diabetes example. Fitting the full model to the data, we find

$$\widehat{\sigma}_e^2 = \frac{\text{Residual Sum of Squares}}{N-p-1} = \frac{1263986}{442-10-1} = 2932.682. \tag{2.61}$$

The output above already gave $\widehat{\sigma}_e = 54.15$, the "residual standard error." Here, $\overline{err} = 1263986/442 = 2859.697$, so that the ERR_{in} for the full model is estimated by

$$\widehat{ERR}_{in} = \overline{err} + 2\widehat{\sigma}_e^2 \, \frac{p+1}{N} = 2859.697 + 2 \cdot 2932.682 \, \frac{11}{442} = 3005.667. \tag{2.62}$$

Now we can see whether dropping some of the variables leads to a lower estimated error. Let us leave out Age and S3. The results:

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -305.9214    31.1218  -9.830  < 2e-16 ***
SEX          -23.0730     5.7901  -3.985 7.92e-05 ***
BMI            5.5915     0.7152   7.818 4.11e-14 ***
BP             1.1069     0.2206   5.018 7.65e-07 ***
S1            -0.8540     0.2817  -3.031  0.00258 **
S2             0.5542     0.3457   1.603  0.10968
S4             4.6944     4.4525   1.054  0.29232
S5            63.0731    10.9163   5.778 1.45e-08 ***
S6             0.2781     0.2703   1.029  0.30401
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 54.04 on 433 degrees of freedom
Multiple R-Squared: 0.5175,     Adjusted R-squared: 0.5086
F-statistic: 58.04 on 8 and 433 DF,  p-value: < 2.2e-16
```

Here, $p^* = 8$, which explains the degrees of freedom: $N - p^* - 1 = 442 - 9 = 433$. The observed error is

$$\overline{err}^* = \frac{1264715}{442} = 2861.345. \tag{2.63}$$

The estimate of in-sample error is then

$$\widehat{ERR}^*_{in} = \overline{err}^* + 2\widehat{\sigma}^2_e \, \frac{p^*+1}{N} = 2861.345 + 2 \cdot 2932.682 \, \frac{8+1}{442} = 2980.775. \tag{2.64}$$

That estimate is a little lower than the 3005.67 using all ten variables, which suggests the for prediction purposes, it is fine to drop those two variables.

2.5.2 Finding the best subset

The method for finding the best subset of variables to use involves calculating the estimated in-sample error for each possible subset, then using the subset with the lowest estimated error. Or one could use one of the subsets which has an error close to the lowest. WIth $p = 10$, there are $2^{10} = 1024$ possible subsets. Running through them all might seem daunting, but computers are good at that. A clever "leaps-and-bounds" algorithm (Furnival and Wilson [1974]) does not need to try all possible subsets; it can rule out some of them as it goes along. In *R*, and implementation is the `leaps` function, which is what we use here. It actually calculates **Mallows' $\mathbf{C_p}$** statistic for each subset, which is defined as

$$C_{p^*} = \frac{\text{Residual Sums of Squares}^*}{\widehat{\sigma}^2_e} + 2(p^*+1) - N, \tag{2.65}$$

which is a linear function of $\widehat{Err}^*_{in}$.

Here are the results for some selected subsets, including the best:

Subset	p^*	$\overline{err}^*$	Penalty	$\widehat{ERR}^*_{in}$	
0010000000	1	3890.457	26.54	3916.997	
0010000010	2	3205.190	39.81	3245.001	
0011000010	3	3083.052	53.08	3136.132	
0011100010	4	3012.289	66.35	3078.639	
0111001010	5	2913.759	79.62	2993.379	
0111100010	5	2965.772	79.62	3045.392	
0111110010	6	2876.684	92.89	2969.574	***
0111100110	6	2885.248	92.89	2978.139	
0111110110	7	2868.344	106.16	2974.504	
0111110111	8	2861.346	119.43	2980.776	
1111111111	10	2859.697	145.97	3005.667	

(2.66)

The "Subset" column indicates by 1's which of the ten variables are in the predictor. Note that as the number of variables increases, the observed error decreases, but the penalty increases. The last two subsets are those we considered above. The best is the one with the asterisks. Here is the regression output for the best:

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -313.7666    25.3848 -12.360  < 2e-16 ***
SEX          -21.5910     5.7056  -3.784 0.000176 ***
BMI            5.7111     0.7073   8.075 6.69e-15 ***
BP             1.1266     0.2158   5.219 2.79e-07 ***
S1            -1.0429     0.2208  -4.724 3.12e-06 ***
S2             0.8433     0.2298   3.670 0.000272 ***
S5            73.3065     7.3083  10.031  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 54.06 on 435 degrees of freedom
Multiple R-Squared: 0.5149,     Adjusted R-squared: 0.5082
F-statistic: 76.95 on 6 and 435 DF,  p-value: < 2.2e-16
```

All the included variables are highly significant. Even though the model was not chosen on the basis of interpretation, one can then see which variables have a large role in predicting the progress of the patient, and the direction of the role, e.g., being fat and having high blood pressure is not good. It is still true that association does not imply causation.

2.5.3 Using R

To load the diabetes data into R, you can use

```
diab <- read.table("http://www-stat.stanford.edu/~hastie/Papers/LARS/diabetes.data",header=T)
```

We'll use `leaps`, which is an R package of functions that has to be installed and loaded into R before you can use it. In Windows, you go to the *Packages* menu, and pick *Load package ....* Pick *leaps*. If *leaps* is not there, you have to install it first. Go back to the *Packages* menu, and choose *Install package(s) ....* Pick a location near you, then pick *leaps*. Once that gets installed, you still have to load it. Next time, you should be able to load it without installing it first.

The function `leaps` will go through all the possible subsets (at least the good ones), and output their C_p's. For the diabetes data, the command is

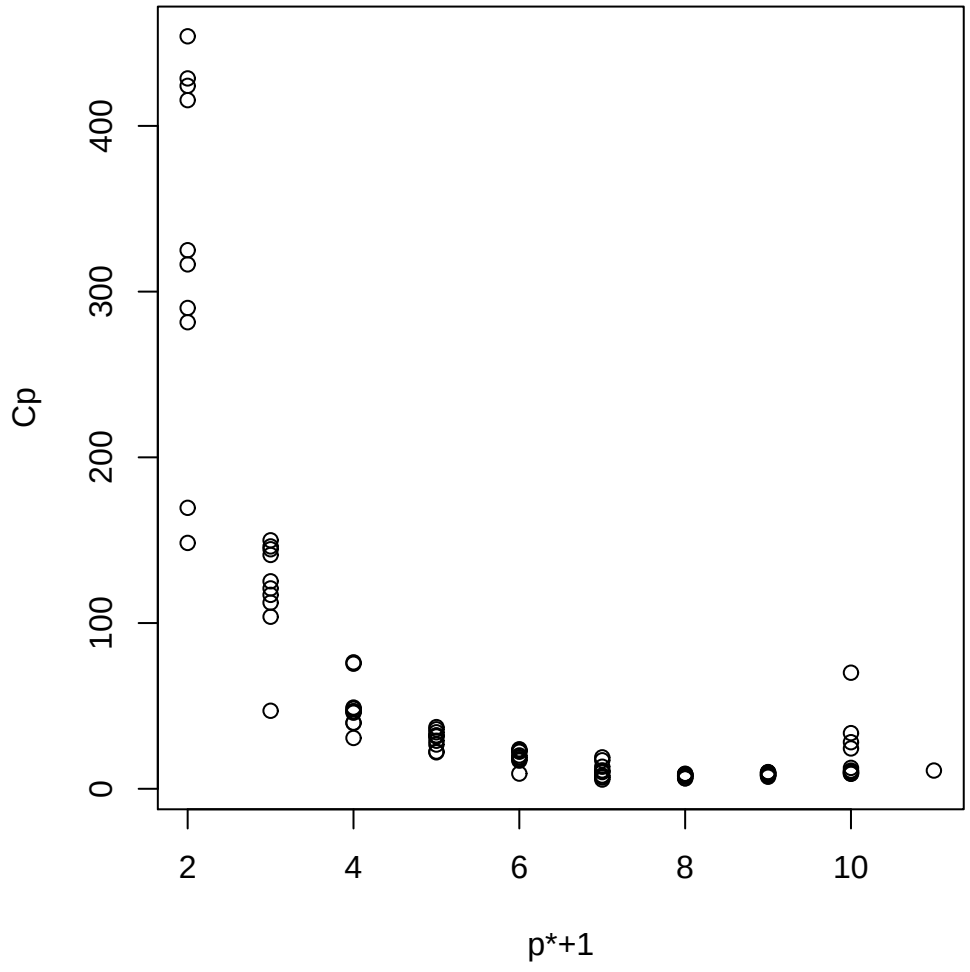
```
diablp <- leaps(diab[,1:10],diab[,11],nbest=10)
```

The `nbest=10` means it outputs the best 10 fits for each p^*. Then `diablp` contains

- `diablp$which`, a matrix with each row indicating which variables are in the model.
- `diablp$Cp`, the C_p statistics corresponding to those models; and
- `diablp$size`, the number of variables in each model, that is, the $p^* + 1$'s.

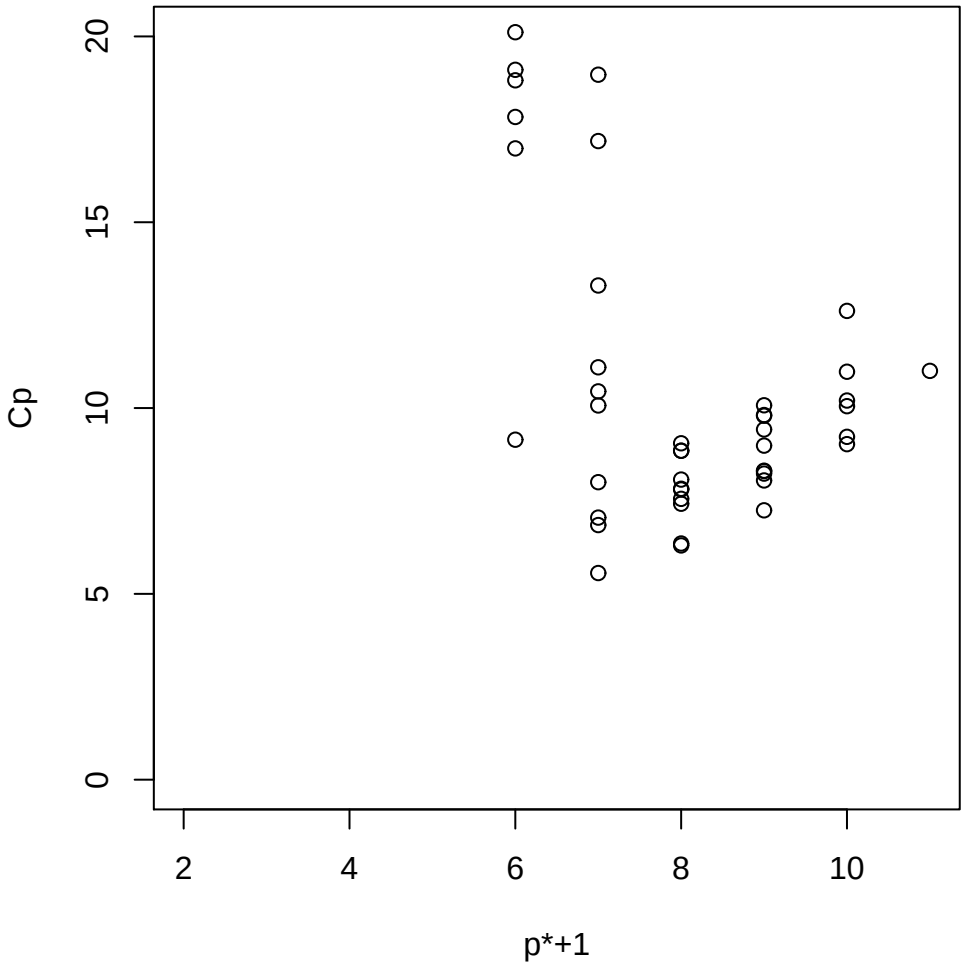
To see the results, plot the size versus C_p:

```
plot(diablp$size,diablp$Cp,xlab="p*+1",ylab="Cp")
```



It looks like $p^* + 1 = 7$ is about right. To focus in a little:

```
plot(diablp$size,diablp$Cp,xlab="p*+1",ylab="Cp",ylim=c(0,20))
```



To figure out which 6 variables are in that best model, find the "which" that has the smallest "Cp":

```
min(diablp$Cp)
diablp$which[diablp$Cp==min(diablp$Cp),]
```

The answers are that the minimum $C_p = 5.56$, and the corresponding model is

```
FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE
```

That means the variables 2, 3, 4, 5, 6, and 9 are in the model. That includes sex, BMI, blood pressure, and three of the blood counts. To fit that model, use `lm` (for "linear model"):

```
diablm <- lm(Y ~ SEX+BMI+BP+S1+S2+S5,data=diab)
summary(diablm)
```

Note: To fit the model with all the variables, you can use

```
diablm <- lm(Y ~ .,data=diab)
```

The "." tells the program to use all variables (except for Y) in the $\mathbf{X}$.

2.6 Regularization: Ridge regression

Subset regression essentially involves setting some of the β_j's to 0, and letting the others be estimated using least squares. A less drastic approach is to allow all estimates to be positive, but to try to constrain them in some way so that they do not become "too big." *Regularization* is a general term that describes methods that impose a penalty on estimates that try to go to far afield. In *ridge regression*, the penalty is the squared norm of the $\underline{\beta}$ vector. The ridge estimate of $\underline{\beta}$ is like the least squares estimate, but seeks to minimize the objective function

$$obj_\lambda(\underline{b}) = \|\underline{y} - \mathbf{X}\underline{b}\|^2 + \lambda\|\underline{b}\|^2. \tag{2.67}$$

The λ is a nonnegative *tuning parameter* that must be chosen in some manner.

If $\lambda = 0$, the objective function is the same as that for least squares. As λ increases, more weight is placed on the b_j's being small. If $\lambda = \infty$ (or thereabouts), then we would take $\underline{b} = \mathbf{0}_{p+1}$. A medium λ yields an estimate somewhere "between" zero and the least squares estimate. Thus a ridge estimate is a type of shrinkage estimate.

A heuristic notion for why one would want to shrink the least squares estimate is that although the least squares estimate is an unbiased estimate of $\underline{\beta}$, its squared length is an overestimate of $\|\underline{\beta}\|^2$:

$$\begin{aligned} E[\|\widehat{\underline{\beta}}_{LS}\|^2] &= \|E[\widehat{\underline{\beta}}_{LS}]\|^2 + trace(Cov[\widehat{\underline{\beta}}_{LS}]) \\ &= \|\underline{\beta}\|^2 + \sigma_e^2 trace((\mathbf{X}'\mathbf{X})^{-1}). \end{aligned} \tag{2.68}$$

If there is a great deal of multicolinearity in the explanatory variables, then $\mathbf{X}'\mathbf{X}$ will be almost non-invertible, which means that $(\mathbf{X}'\mathbf{X})^{-1}$ will be very large.

The ridge estimator of $\underline{\beta}$ for given λ is the $\underline{b}$ that minimizes the resulting $obj_\lambda(\underline{b})$ in (2.67). Finding the ridge estimate is no more difficult than finding the least squares estimate. We use a trick. Write the penalty part as

$$\lambda\|\underline{b}\|^2 = \|\underline{0}_p - \sqrt{\lambda}\ \mathbf{I}_p\underline{b}\|^2 \tag{2.69}$$

Pretend that the zero vector represents some additional y's, and the $\sqrt{\lambda}\mathbf{I}_p$ matrix represents additional x's, so that

$$obj_\lambda(\underline{b}) = \|\underline{y} - \mathbf{X}\underline{b}\|^2 + \|\underline{0}_p - \sqrt{\lambda}\ \mathbf{I}_p\underline{b}\|^2 = \|\underline{y}_\lambda - \mathbf{X}_\lambda\underline{b}\|^2, \tag{2.70}$$

where

$$\underline{y}_\lambda = \begin{pmatrix} \underline{y} \\ \underline{0}_{p+1} \end{pmatrix} \text{ and } \mathbf{X}_\lambda = \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda}\,\mathbf{I}_{p+1} \end{pmatrix}. \tag{2.71}$$

The ridge estimate of $\underline{\beta}$ is then

$$\begin{aligned} \widehat{\underline{\beta}}_\lambda &= (\mathbf{X}'_\lambda \mathbf{X}_\lambda)^{-1} \mathbf{X}'_\lambda \underline{y}_\lambda \\ &= (\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}_{p+1})^{-1} \mathbf{X}' \underline{y}. \end{aligned} \tag{2.72}$$

Originally, ridge regression was introduced to improve estimates when $\mathbf{X}'\mathbf{X}$ is nearly non-invertible. By adding a small λ to the diagonals, the inverse becomes much more stable.

The corresponding ridge predictor of the new individuals is

$$\widehat{\underline{Y}}_\lambda^{New} = \mathbf{X}\widehat{\underline{\beta}}_\lambda = \mathbf{X}(\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}_{p+1})^{-1}\mathbf{X}'\underline{Y} \equiv \mathbf{H}_\lambda \underline{Y}. \tag{2.73}$$

This $\mathbf{H}_\lambda$ is symmetric but not a projection matrix (it is not idempotent), unless $\lambda = 0$. Note that indeed, $\lambda = 0$ leads to the least squares estimate, and letting $\lambda \to \infty$ drives the prediction to zero.

Two questions: How good is the ridge predictor? What should λ be? As in the previous section, we figure out the in-sample error, or at least an estimate of it, for each λ. We start with the means and covariances:

$$E[\widehat{\underline{Y}}_\lambda^{New}] = \mathbf{H}_\lambda \; E[\underline{Y}] = \mathbf{H}_\lambda \mathbf{X}\underline{\beta} \text{ and } Cov[\widehat{\underline{Y}}_\lambda^{New}] = Cov[\mathbf{H}_\lambda \underline{Y} \mathbf{H}_\lambda] = \sigma_e^2 \mathbf{H}_\lambda^2, \tag{2.74}$$

hence

$$E[\underline{Y} - \widehat{\underline{Y}}_\lambda^{New}] = \mathbf{X}\underline{\beta} - \mathbf{H}_\lambda \mathbf{X}\underline{\beta} = (\mathbf{I}_N - \mathbf{H}_\lambda)\mathbf{X}\underline{\beta} \text{ and} \tag{2.75}$$

$$Cov[\underline{Y}^{New} - \widehat{\underline{Y}}_\lambda^{New}] = \sigma_e^2 \mathbf{I}_N + \sigma_e^2 \mathbf{H}_\lambda^2 = \sigma_e^2 (\mathbf{I}_N + \mathbf{H}_\lambda^2). \tag{2.76}$$

The in-sample error is

$$\begin{aligned} ERR_{in,\lambda} &= \frac{1}{N} \; E[\|\underline{Y} - \widehat{\underline{Y}}_\lambda^{New}\|^2] \\ &= \frac{1}{N} \; (\|(\mathbf{I}_N - \mathbf{H}_\lambda)\mathbf{X}\underline{\beta}\|^2 + \sigma_e^2 \; trace(\mathbf{I}_N + \mathbf{H}_\lambda^2)) \\ &= \frac{1}{N} \; \underline{\beta}'\mathbf{X}'(\mathbf{I}_N - \mathbf{H}_\lambda)^2 \mathbf{X}\underline{\beta} + \sigma_e^2 + \sigma_e^2 \; \frac{trace(\mathbf{H}_\lambda^2)}{N}. \end{aligned} \tag{2.77}$$

Repeating part of equation (2.46), we can compare the errors of the subset and ridge predictors:

$$\begin{aligned} ERR_{in,\lambda} &= \tfrac{1}{N} \; \underline{\beta}'\mathbf{X}'(\mathbf{I}_N - \mathbf{H}_\lambda)^2 \mathbf{X}\underline{\beta} \; + \; \sigma_e^2 \; + \; \sigma_e^2 \; \tfrac{trace(\mathbf{H}_\lambda^2)}{N}; \\ ERR_{in}^* &= \tfrac{1}{N} \; \underline{\beta}'\mathbf{X}'(\mathbf{I}_N - \mathbf{H}^*)\mathbf{X}\underline{\beta} \; + \; \sigma_e^2 \; + \; \sigma_e^2 \; \tfrac{p^*+1}{N}; \\ \text{Error} &= \text{Bias}^2 \; + \; \text{Inherent variance} \; + \; \text{Estimation variance}. \end{aligned} \tag{2.78}$$

2.6.1 Estimating the in-sample errors

The larger the λ, the smaller the $\mathbf{H}_\lambda$, which means as λ increases, the bias increases but the estimation variance decreases. In order to decide which λ is best, we have to estimate the $ERR_{in,\lambda}$ for all (or many) λ's. Just as for subset regression, we start by looking at the error for the observed data:

$$\overline{err}_\lambda = \frac{1}{N} \|\underline{Y} - \underline{\widehat{Y}}_\lambda^{New}\|^2 = \frac{1}{N} \|\underline{Y} - \mathbf{H}_\lambda \underline{Y}\|^2. \tag{2.79}$$

We basically repeat the calculations from (2.53):

$$E[\underline{Y} - \mathbf{H}_\lambda \underline{Y}] = (\mathbf{I}_N - \mathbf{H}_\lambda)\mathbf{X}\underline{\beta}, \tag{2.80}$$

and

$$\begin{aligned} Cov[\underline{Y} - \mathbf{H}_\lambda \underline{Y}] &= Cov[(\mathbf{I}_N - \mathbf{H}_\lambda)\underline{Y}] \\ &= \sigma_e^2 \ (\mathbf{I}_N - \mathbf{H}_\lambda)^2 \\ &= \sigma_e^2 (\mathbf{I}_N - 2\mathbf{H}_\lambda + \mathbf{H}_\lambda^2), \end{aligned} \tag{2.81}$$

hence

$$\begin{aligned} E[\overline{err}_\lambda] &= \tfrac{1}{N} \ \underline{\beta}' \mathbf{X}' (\mathbf{I}_N - \mathbf{H}_\lambda)^2 \mathbf{X} \underline{\beta} \ + \ \sigma_e^2 + \sigma_e^2 \ \tfrac{trace(\mathbf{H}_\lambda^2)}{N} \ - \ 2\sigma_e^2 \tfrac{trace(\mathbf{H}_\lambda)}{N}; \\ ERR_{in,\lambda} &= \tfrac{1}{N} \ \underline{\beta}' \mathbf{X}' (\mathbf{I}_N - \mathbf{H}_\lambda)^2 \mathbf{X} \underline{\beta} \ + \ \sigma_e^2 + \sigma_e^2 \ \tfrac{trace(\mathbf{H}_\lambda^2)}{N}. \end{aligned} \tag{2.82}$$

To find an unbiased estimator of the in-sample error, we add $2\widehat{\sigma}_e^2 \, trace(\mathbf{H}_\lambda)/N$ to the observed error. Here, $\widehat{\sigma}_e^2$ is the estimate derived from the regular least squares fit using all the variables, as before. To compare to the subset regression,

$$\begin{aligned} \widehat{ERR}_{in,\lambda} &= \overline{err}_\lambda + 2\widehat{\sigma}_e^2 \ \tfrac{trace(\mathbf{H}_\lambda)}{N}; \\ \widehat{ERR}_{in}^* &= \overline{err}^* + 2\widehat{\sigma}_e^2 \ \tfrac{p^*+1}{N}; \end{aligned} \tag{2.83}$$

$$\text{Prediction error} = \text{Observed error} + \text{Penalty for estimating parameters.}$$

If we put $\mathbf{H}^*$ in for $\mathbf{H}_\lambda$, we end up with the same in-sample error estimates. The main difference is that the number of parameters for subset regression, $p^* + 1$, is replaced by the $trace(\mathbf{H}_\lambda)$. This trace is often called the **effective degrees of freedom**,

$$edf(\lambda) = trace(\mathbf{H}_\lambda). \tag{2.84}$$

If $\lambda = 0$, then $\mathbf{H}_\lambda = \mathbf{H}$, so the $edf(0) = p + 1$. As $\lambda \to \infty$, the $\mathbf{H}_\lambda \to 0$, because λ is in the denominator, hence $edf(\infty) = 0$. Thus the effective degrees of freedom is somewhere between 0 and $p + 1$. By shrinking the parameters, you "decrease" the effective number, in a continuous manner.

2.6.2 Finding the best λ

We make a few modifications in order to preserve some "equivariance." That is, we do not want the predictions to be effected by the units of the variables. That is, it should not matter whether height is in inches or centimeters, or whether temperature is in degrees Fahrenheit or centigrade or Kelvin. To that end, we make the following modifications:

1. Normalize $\underline{y}$ so that it has zero mean (that is, subtract $\overline{y}$ from $\underline{y}$);

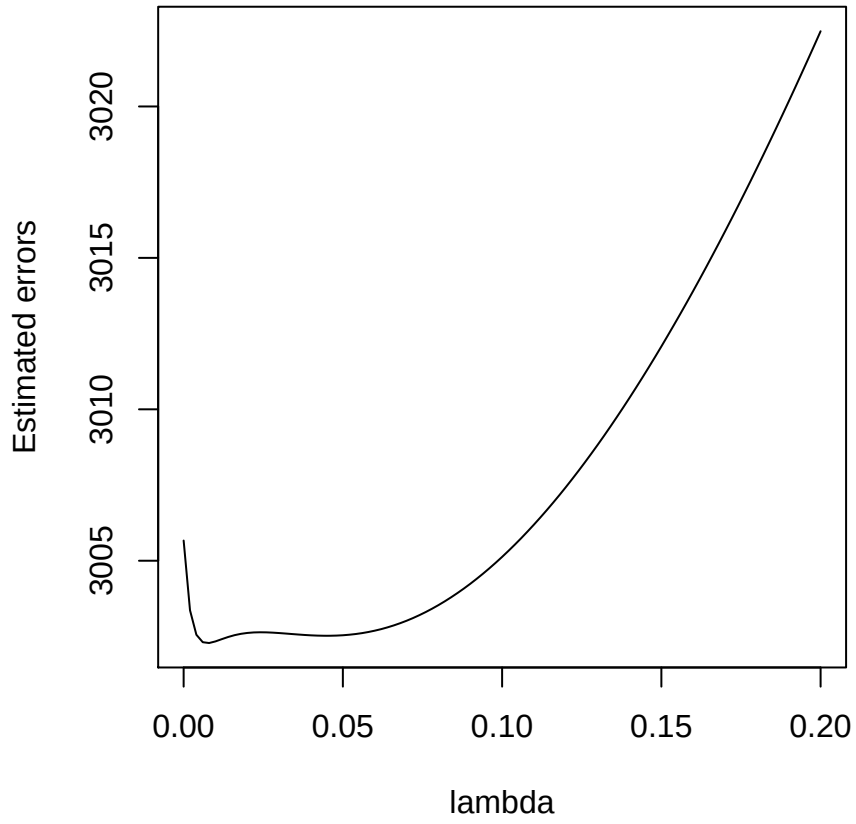
2. Normalize the explanatory variables so that they have zero means and squared norms of 1.

With the variables normalized as in #1 and #2 above, we can eliminate the $\underline{1}_N$ vector from $\mathbf{X}$, and the β_0 parameter. #2 also implies that the diagonals of the $\mathbf{X}'\mathbf{X}$ matrix are all 1's. After finding the best λ and the corresponding estimate, we have to untangle things to get back to the original x's and y's. The one change we need to make is to add "1" to the effective degrees of freedom, because we are surreptitiously estimating β_0.

For any particular λ, the calculations of $\overline{err}_\lambda$ and $edf(\lambda)$ are easy enough using a computer. To find the best λ, one can choose a range of λ's, then calculate the $\widehat{ERR}_{in,\lambda}$ for each one over a grid in that range. With the normalizations we made, the best λ is most likely reasonably small, so starting with a range of $[0, 1]$ is usually fine.

We next tried the ridge predictor on the normalized diabetes data.

Here is a graph of the $\widehat{ERR}_{in,\lambda}$'s versus λ:



Here are the details for some selected λ's including the best:

λ	$edf(\lambda)$	$\overline{err}_\lambda$	$penalty$	$\widehat{ERR}_{in,\lambda}$	
0	11	2859.70	145.97	3005.67	
0.0074	10.38	2864.57	137.70	3002.28	***
0.04	9.46	2877.04	125.49	3002.54	
0.08	8.87	2885.82	117.72	3003.53	
0.12	8.44	2895.47	111.96	3007.43	
0.16	8.08	2906.69	107.22	3013.91	
0.20	7.77	2919.33	103.15	3022.48	

(2.85)

Note the best λ is quite small, 0.0074. The first line is the least squares prediction using all the variables. We can compare the estimates of the coefficients for three of our models: least squares using all the variables, the best subset regression, and the best ridge regression. (These estimates are different than those in the previous section because we have normalized

the x's.)

	Full LS	Subset	Ridge
AGE	-10.01	0	-7.70
SEX	-239.82	-226.51	-235.67
BMI	519.85	529.88	520.97
BP	324.38	327.22	321.31
$S1$	-792.18	-757.93	-438.60
$S2$	476.74	538.58	196.36
$S3$	101.04	0	-53.74
$S4$	177.06	0	136.33
$S5$	751.27	804.19	615.37
$S6$	67.63	0	70.41
$\|\|\underline{\widehat{\beta}}\|\|$	1377.84	1396.56	1032.2
edf	11	7	10.38
$\overline{err}$	2859.696	2876.68	2864.57
$penalty$	145.97	92.89	137.70
$\widehat{ERR}$	3005.67	2969.57	3002.28

(2.86)

The best subset predictor is the best of these. It improves on the full model by about 35 points, whereas the best ridge improves on the full model by very little, about 3 points. Notice also that the best subset predictor is worse than the ridge one in observed error, but quite a bit better in penalty. The ridge estimator does have the smallest $\|\underline{\widehat{\beta}}\|^2$.

Comparing the actual coefficients, we see that all three methods give fairly similar estimates. The main difference between the full model and subset are that the subset model sets the four left-out beta's to zero. The ridge estimates are generally smaller in magnitude than the full model's, but follow s a similar pattern.

Taking everything into consideration, the subset method looks best here. It is the simplest as well as having the lowest prediction error. Of course, one could use ridge regression on just the six variables in the best subset model.

Recovering the unnormalized estimates

To actually use the chosen predictor for new observations, one would want to go back to results in the original units. If $\underline{x}_{[j]}$ is the original j^{th} column vector, then the normalized vector is

$$\underline{x}_{[j]}^{Norm} = \frac{\underline{x}_{[j]} - \overline{x}_{[j]}\underline{1}_N}{\|\underline{x}_{[j]} - \overline{x}_{[j]}\underline{1}_N\|}, \tag{2.87}$$

where $\overline{x}_{[j]}$ is the mean of the elements of $\underline{x}_{[j]}$. Also, the normalized y is $\underline{Y}^{Norm} = \underline{Y} - \overline{Y}\underline{1}_N$. The fit to the normalized data is

$$\underline{Y}^{Norm} = b_1^{Norm}\underline{x}_{[1]}^{Norm} + \cdots + b_p^{Norm}\underline{x}_{[p]}^{Norm}, \tag{2.88}$$

which expands to

$$\begin{aligned} \underline{Y} - \overline{Y}\ \underline{1}_N &= b_1^{Norm}\ \frac{\underline{x}_{[1]} - \overline{x}_{[1]}\underline{1}_N}{\|\underline{x}_{[1]} - \overline{x}_{[1]}\underline{1}_N\|} + \cdots + b_p^{Norm}\ \frac{\underline{x}_{[p]} - \overline{x}_{[p]}\underline{1}_N}{\|\underline{x}_{[p]} - \overline{x}_{[p]}\underline{1}_N\|} \\ &= b_1\ (\underline{x}_{[1]} - \overline{x}_{[1]}\underline{1}_N) + \cdots + b_p\ (\underline{x}_{[p]} - \overline{x}_{[p]}\underline{1}_N), \end{aligned} \tag{2.89}$$

where b_j is the coefficient in the unnormalized fit,

$$b_j = b_j^{Norm}\ \frac{1}{\|\underline{x}_{[j]} - \overline{x}_{[j]}\underline{1}_N\|}. \tag{2.90}$$

Then collecting the terms multiplying the $\underline{1}_N$ vector,

$$\underline{Y} = (\overline{Y} - b_1\overline{x}_{[1]} - \cdots - b_p\overline{x}_{[p]})\ \underline{1}_N + b_1\ \underline{x}_{[1]} + \cdots + b_p\ \underline{x}_{[p]}. \tag{2.91}$$

Thus the intercept for the nonnormalized data is

$$b_0 = \overline{Y} - b_1\overline{x}_{[1]} - \cdots - b_p\overline{x}_{[p]}. \tag{2.92}$$

2.6.3 Using R

The `diab` data is the same as in Section 2.5.3. We first normalize the variables, calling the results `x` and `y`:

```
p <- 10
N <- 442
sigma2 <- sum(lm(Y ~.,data=diab)$resid^2)/(N-p-1)
# sigma2 is the residual variance from the full model
y <- diab[,11]
y <- y-mean(y)
x <- diab[,1:10]
x <- sweep(x,2,apply(x,2,mean),"-")
x <- sweep(x,2,sqrt(apply(x^2,2,sum)),"/")
```

One approach is to perform the matrix manipulations directly. You first have to turn `x` into a matrix. Right now it is a data frame[3].

```
x <- as.matrix(x)
```

For a given `lambda`, ridge regression proceeds as follows:

[3]One thing about *R* that drives me crazy is the distinction between data frames and matrices. For some purposes either will do, for some one needs a matrix, for others one needs a data frame. At least it is easy enough to change from one to the other, using `as.data.frame` or `as.matrix`.

```
beta.lambda <- solve(t(x)%*%x+lambda*diag(p),t(x)%*%y)
# diag(p) is the pxp identity
h.lambda <- x%*%solve(t(x)%*%x+lambda*diag(p),t(x))
y.lambda <- x%*%beta.lambda
rss.lambda <- sum((y-y.lambda)^2)
err.lambda <- rss.lambda/N
edf.lambda <- sum(diag(h.lambda))+1
pen.lambda <- 2*sigma2*(edf.lambda)/N
errhat.lambda <- err.lambda + pen.lambda
```

These calculations work, but are not terrifically efficient.

Another approach that works with the linear model fitter is to make use of the augmented data as in (2.71), where here we are leaving out the $\underline{1}_N$ vector:

```
xx <- rbind(x,sqrt(lambda)*diag(p))
yy <- c(y,rep(0,10))
lm.lambda <- lm(yy~xx-1)
```

The `lm.lambda` will have the correct estimates of the coefficients, but the other output is not correct, since it is using the augmented data as well. The first N of the residuals *are* the correct residuals, hence

```
rss.lambda <- sum(lm.lambda$resid[1:N]^2)
```

The sum of squares of the last p residuals yields the $\lambda\|\underline{\widehat{\beta}}_\lambda\|^2$. I don't know if there is a clever way to get the effective degrees of freedom. (See (2.95) for a method.)

You can skip this subsection

A more efficient procedure if one is going to calculate the regression for many λ's is to use the singular value decomposition of $\mathbf{X}$ ($N \times p$ now), which consists of writing

$$\mathbf{X} = \mathbf{U}\mathbf{\Delta}\mathbf{V}', \tag{2.93}$$

where $\mathbf{U}$ is an $N \times p$ matrix with orthonormal columns, $\mathbf{\Delta}$ is a $p \times p$ diagonal matrix with diagonal elements $\delta_1 \geq \delta_2 \geq \cdots \geq \delta_p \geq 0$, and $\mathbf{V}$ is a $p \times p$ orthogonal matrix. We can then find

$$\begin{aligned}
\mathbf{H}_\lambda &= \mathbf{X}(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}_p)^{-1}\mathbf{X}' \\
&= \mathbf{U}\mathbf{\Delta}\mathbf{V}'(\mathbf{V}\mathbf{\Delta}\mathbf{U}'\mathbf{U}\mathbf{\Delta}\mathbf{V}' + \lambda\mathbf{I}_p)^{-1}\mathbf{V}\mathbf{\Delta}\mathbf{U}' \\
&= \mathbf{U}\mathbf{\Delta}\mathbf{V}'(\mathbf{V}(\mathbf{\Delta}^2 + \lambda\mathbf{I}_p)\mathbf{V}')^{-1}\mathbf{V}\mathbf{\Delta}\mathbf{U}' \quad (\mathbf{U}'\mathbf{U} = \mathbf{I}_p) \\
&= \mathbf{U}\mathbf{\Delta}(\mathbf{\Delta}^2 + \lambda\mathbf{I}_p)^{-1}\mathbf{\Delta}\mathbf{U}' \quad (\mathbf{V}'\mathbf{V} = \mathbf{I}_p) \\
&= \mathbf{U}\left\{\frac{\delta_i^2}{\delta_i^2 + \lambda}\right\}\mathbf{U}',
\end{aligned} \tag{2.94}$$

where the middle matrix is meant to be a diagonal matrix with $\delta_i^2/(\delta_i^2+\lambda)$'s down the diagonal. Then

$$\begin{aligned} edf(\lambda) &= trace(\mathbf{H}_\lambda)+1 \\ &= trace(\mathbf{U}\left\{\frac{\delta_i^2}{\delta_i^2+\lambda}\right\}\mathbf{U}')+1 \\ &= trace(\left\{\frac{\delta_i^2}{\delta_i^2+\lambda}\right\}\mathbf{U}'\mathbf{U})+1 \\ &= \sum_{i=1}^{p}\frac{\delta_i^2}{\delta_i^2+\lambda}+1. \end{aligned} \tag{2.95}$$

To find $\overline{err}_\lambda$, we start by noting that $\mathbf{U}$ in (2.93) has p orthogonal columns. We can find $N-p$ more columns, collecting them into the $N\times(N-p)$ matrix $\mathbf{U}_2$, so that

$$\mathbf{\Gamma} = (\mathbf{U}\ \mathbf{U}_2) \text{ is an } N\times N \text{ orthogonal matrix.} \tag{2.96}$$

The predicted vector can then be rotated,

$$\begin{aligned} \mathbf{\Gamma}'\underline{\widehat{y}}_\lambda^{New} &= \mathbf{\Gamma}'\mathbf{U}\left\{\frac{\delta_i^2}{\delta_i^2+\lambda}\right\}\mathbf{U}'\underline{y} \\ &= \begin{pmatrix}\mathbf{U}'\mathbf{U}\\ \mathbf{U}_2'\mathbf{U}\end{pmatrix}\left\{\frac{\delta_i^2}{\delta_i^2+\lambda}\right\}\mathbf{U}'\underline{y} \\ &= \begin{pmatrix}\mathbf{I}_p\\ \mathbf{0}\end{pmatrix}\left\{\frac{\delta_i^2}{\delta_i^2+\lambda}\right\}\mathbf{U}'\underline{y} \\ &= \begin{pmatrix}\left\{\frac{\delta_i^2}{\delta_i^2+\lambda}\right\}\mathbf{U}'\underline{y}\\ \mathbf{0}\end{pmatrix}. \end{aligned} \tag{2.97}$$

Because the squared norm of a vector is not changed when multiplying by an orthogonal matrix,

$$\begin{aligned} N\ \overline{err}_\lambda &= \|\mathbf{\Gamma}(\underline{y}-\underline{\widehat{y}}_\lambda^{New})\|^2 \\ &= \|\begin{pmatrix}\mathbf{U}'\underline{y}\\ \mathbf{U}_2'\underline{y}\end{pmatrix}-\begin{pmatrix}\left\{\frac{\delta_i^2}{\delta_i^2+\lambda}\right\}\mathbf{U}'\underline{y}\\ \mathbf{0}\end{pmatrix}\|^2 \\ &= \|\begin{pmatrix}\left\{\frac{\lambda}{\delta_i^2+\lambda}\right\}\mathbf{U}'\underline{y}\\ \mathbf{U}_2'\underline{y}\end{pmatrix}\|^2 \\ &= \|\left\{\frac{\lambda}{\delta_i^2+\lambda}\right\}\underline{w}\|^2+\|\mathbf{U}_2\underline{y}\|^2, \quad \text{where } \underline{w}=\mathbf{U}'\underline{y}. \end{aligned} \tag{2.98}$$

When $\lambda=0$, we know we have the usual least squares fit using all the variables, hence $N\overline{err}_0 = RSS$, i.e.,

$$\|\mathbf{U}_2\underline{y}\|^2 = RSS. \tag{2.99}$$

Then from (2.83),

$$\begin{aligned} \widehat{ERR}_{in,\lambda} &= \overline{err}_\lambda + 2\hat{\sigma}_e^2 \, \frac{trace(\mathbf{H}_\lambda)}{N} \\ &= \frac{1}{N} \, (RSS + \sum_{i=1}^{p} \left(\frac{\lambda}{\delta_i^2 + \lambda} \right)^2 w_i^2) + 2\hat{\sigma}_e^2 \, \frac{1}{N} \, (\sum_{i=1}^{p} \frac{\delta_i^2}{\delta_i^2 + \lambda} + 1). \end{aligned} \quad (2.100)$$

Why is equation (2.100) useful? Because the singular value decomposition (2.93) needs to be calculated just once, as do RSS and $\hat{\sigma}_e^2$. Then $\underline{w} = \mathbf{U}'\underline{y}$ is easy to find, and all other elements of the equation are simple functions of these quantities.

To perform these calculations in R, start with

```
N <- 442
p <- 10
s <- svd(x)
w <- t(s$u)%*%y
d2 <- s$d^2
rss <- sum(y^2)-sum(w^2)
s2 <- rss/(N-p-1)
```

Then to find the $\widehat{ERR}_{in,\lambda}$'s for a given set of λ's, and plot the results, use

```
lambdas <- (0:100)/100
errhat <- NULL
for(lambda in lambdas) {
   rssl <- sum((w*lambda/(d2+lambda))^2)+rss
   edfl <- sum(d2/(d2+lambda))+1
   errhat <- c(errhat,(rssl + 2*s2*edfl)/N)
}
plot(lambdas,errhat,type='l',xlab="lambda",ylab="Estimated errors")
```

You might want to repeat, focussing on smaller λ, e.g., `lambdas <- (0:100)/500`.

To find the best, it is easy enough to try a finer grid of values. Or you can use the `optimize` function in R. You need to define the function that, given λ, yields $\widehat{ERR}_{in,\lambda}$:

```
f <- function(lambda) {
   rssl <- sum((w*lambda/(d2+lambda))^2)+rss
   edf <- sum(d2/(d2+lambda))+1
   (rssl + 2*s2*edf)/N
}

optimize(f,c(0,.02))
```

The output gives the best λ (at least the best it found) and the corresponding error:

```
$minimum
[1] 0.007378992

$objective
[1] 3002.279
```

2.7 Lasso

The objective function in ridge regression (2.67) uses sums of squares for both the error term and the regularizing term, i.e., $\|\underline{b}\|^2$. Lasso keeps the sum of squares for the error, but looks at the absolute values of the b_j's, so that the objective function is

$$obj_\lambda^L(\underline{b}) = \|\underline{y} - \mathbf{X}\underline{b}\|^2 + \lambda \sum_{j=1}^{p} |b_j|. \tag{2.101}$$

Notice we are leaving out the intercept in the regularization part. One could leave it in.

Note: Both ridge and lasso can equally well be thought of as constrained estimation problems. Minimizing the ridge objective function for a given λ is equivalent to minimizing

$$\|\underline{y} - \mathbf{X}\underline{b}\|^2 \text{ subject to } \|\underline{b}\|^2 \leq t \tag{2.102}$$

for some t. There is a one-to-one correspondence between λ's and t's (the larger the λ, the smaller the t). Similarly, lasso minimizes

$$\|\underline{y} - \mathbf{X}\underline{b}\|^2 \text{ subject to } \sum_{j=1}^{p} |b_j| \leq t \tag{2.103}$$

for some t (not the same as that for ridge). □

The lasso estimate of $\underline{\beta}$ is the $\underline{b}$ that minimizes obj_λ^L in (2.101). The estimate is not a simple linear function of $\underline{y}$ as before. There are various ways to compute the minimizer. One can use convex programming methods, because we are trying to minimize a convex function subject to linear constraints. In Efron et al. [2004][4], the authors present **least angle regression**, which yields a very efficient method for calculating the lasso estimates for all λ.

Here we discuss an inefficient approach (that is, do not try this at home), but one that may provide some insight. The function $obj_\lambda^L(\underline{b})$ is strictly convex in $\underline{b}$, because the error sum of squares is strictly convex, and each $|b_j|$ is convex (though not strictly). Also, as any b_j goes to $\pm\infty$, the objective function goes to infinity. Thus there does exist a unique minimum. Denote it by $\underline{\widehat{\beta}}_\lambda^L$.There are two possibilities:

1. obj_λ^L is differentiable with respect to each b_j at the minimum.

[4] http://www-stat.stanford.edu/∼hastie/Papers/LARS/LeastAngle_2002.pdf

2. obj_λ^L is not differentiable with respect to at least one b_j at the minimum.

The objective function is differential with respect to b_j for all b_j except $b_j = 0$, because of the $|b_j|$ part of the equation. Which means that if the objective function is not differentiable with respect to b_j at the minimum, $\widehat{\beta}_{\lambda,j}^L = 0$. As in the subset method, let $\underline{b}^*$, $\underline{\widehat{\beta}}_\lambda^{L*}$, and $\mathbf{X}^*$ contain just the elements for the coefficients **not** set to zero at the minimum. It must be that

$$\frac{\partial}{\partial b_j^*} \left[\|\underline{y} - \mathbf{X}^*\underline{b}^*\|^2 + \lambda \sum |b_j^*|\right]\Big|_{\underline{b}^* = \underline{\widehat{\beta}}_\lambda^{L*}} = 0 \tag{2.104}$$

for the b_j^*'s not set to zero. The derivative of the sum of squares part is the same as before, in (2.13), and

$$\frac{d}{dz}\,|z| = Sign(z) \quad \text{for } z \neq 0, \tag{2.105}$$

hence setting the derivatives equal to zero results in the equation

$$-2\mathbf{X}^{*'}(\underline{y} - \mathbf{X}^*\underline{b}^*) + \lambda Sign(\underline{b}^*) = \underline{0}, \tag{2.106}$$

where sign of a vector is the vector of signs. The solution is the estimate, so

$$\begin{aligned} \mathbf{X}^{*'}\mathbf{X}^*\underline{\widehat{\beta}}_\lambda^{L*} &= \mathbf{X}^{*'}\underline{y} - \frac{1}{2}\,\lambda Sign(\underline{\widehat{\beta}}_\lambda^{L*}); \\ \Rightarrow \quad \underline{\widehat{\beta}}_\lambda^{L*} &= (\mathbf{X}^{*'}\mathbf{X}^*)^{-1}(\mathbf{X}^{*'}\underline{y} - \frac{1}{2}\,\lambda Sign(\underline{\widehat{\beta}}_\lambda^{L*})). \end{aligned} \tag{2.107}$$

This equation shows that the lasso estimator is a shrinkage estimator as well. If $\lambda = 0$, we have the usual least squares estimate, and for positive λ, the estimates are decreased if positive and increased if negative. (Also, realize that this equation is really not an explicit formula for the estimate, since it appears on both sides.)

The non-efficient method for finding $\underline{\widehat{\beta}}_\lambda^L$ is for given λ, for each subset, see if (2.107) can be solved. If not, forget that subset. If so, calculate the resulting obj_λ^L. Then choose the estimate with the lowest obj_λ^L.

The important point to note is that lasso incorporates both subset selection and shrinkage.

2.7.1 Estimating the in-sample errors

Figuring out an exact expression for the in-sample error, $ERR_{in,\lambda}^L$, appears difficult, mainly because we do not have an explicit expression for the estimate or prediction. The paper Efron et al. [2004] has some results, including simulations as well as exact calculations in some cases (such as when the x-variables are orthogonal), which leads to the suggestion to use $p^* + 1$ as the effective degrees of freedom in estimating the in-sample errors. Thus the formula is the same as that for subset selection:

$$\widehat{ERR}_{in,\lambda}^L = \overline{err}_\lambda^L + 2\widehat{\sigma}_e^2\,\frac{p^* + 1}{N}, \tag{2.108}$$

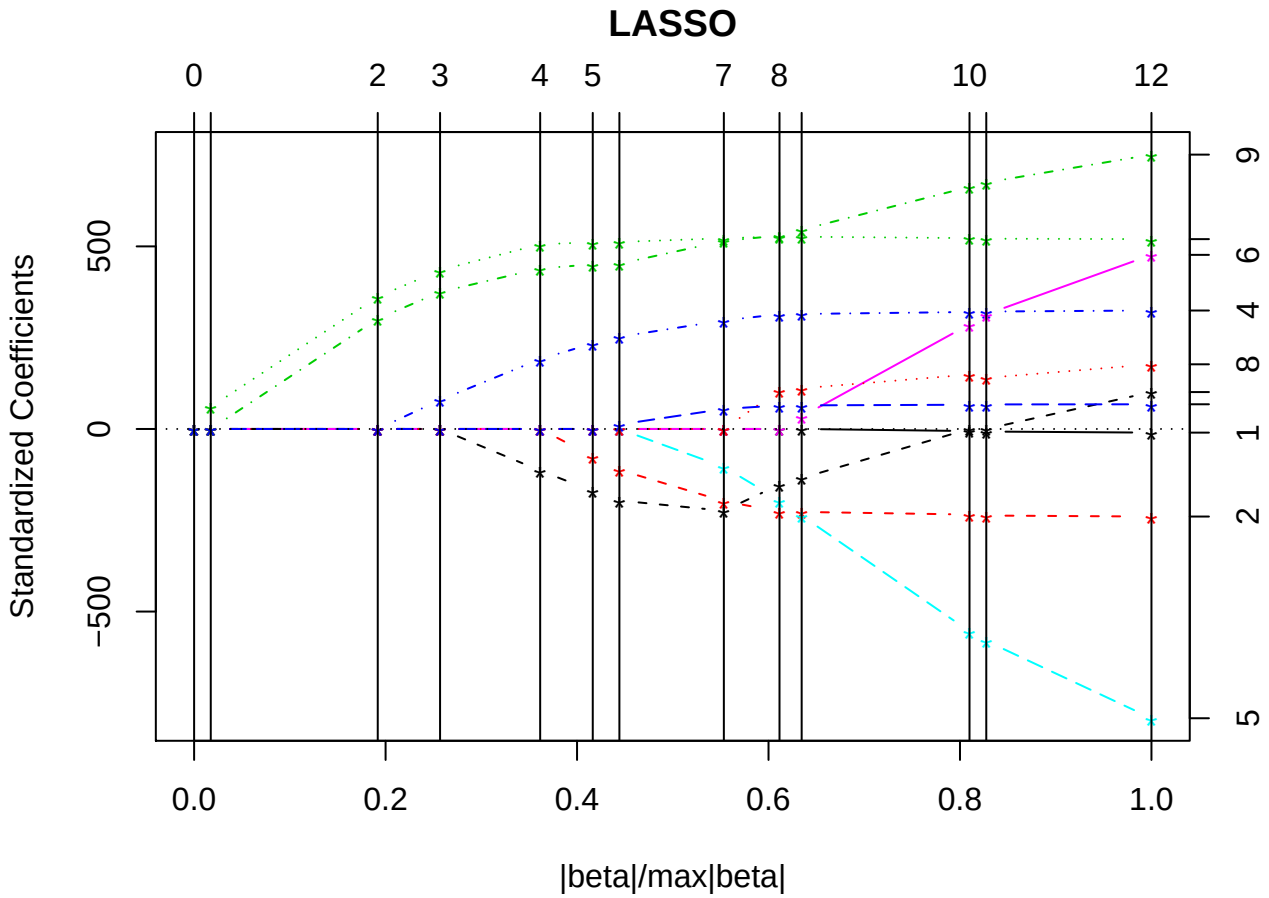
where

$$\overline{err}_\lambda^L = \frac{1}{N} \|\underline{y} - \mathbf{X}\underline{\widehat{\beta}}_\lambda^L\|^2, \tag{2.109}$$

the observed error for the lasso predictor. Looking at (2.107), you can imagine the estimate (2.108) is reasonable if you ignore the $\lambda Sign(\underline{\widehat{\beta}}_\lambda^{L*})$ part when finding the covariance of the prediction errors.

2.7.2 Finding the best λ

We will use the `lars` routine in *R*, which was written by Bradley Efron and Trevor Hastie. As for ridge, the data is normalized. The program calculates the estimates for all possible λ in one fell swoop. The next plot shows all the estimates of the coefficients for the diabetes data:



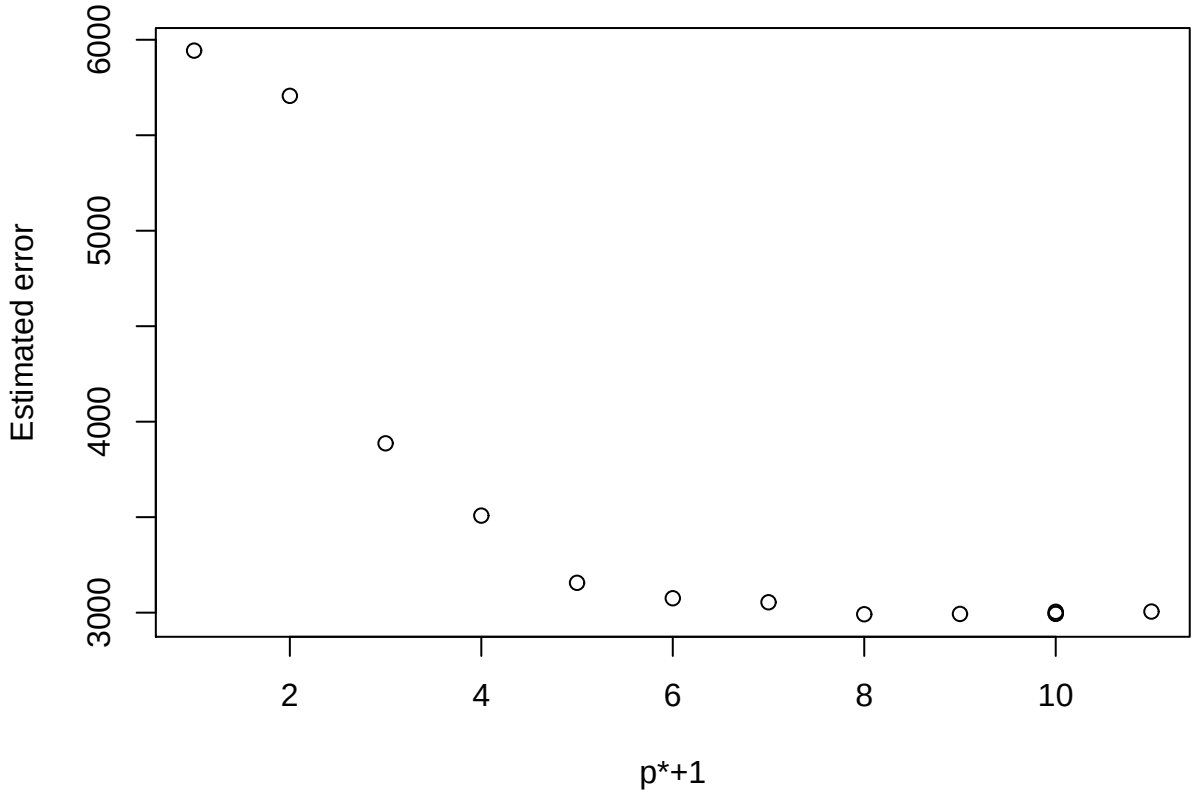
Note that the horizontal axis is not λ, but is

$$\frac{\sum |\widehat{\beta}_{j,\lambda}^L|}{\sum |\widehat{\beta}_{j,LS}|}, \tag{2.110}$$

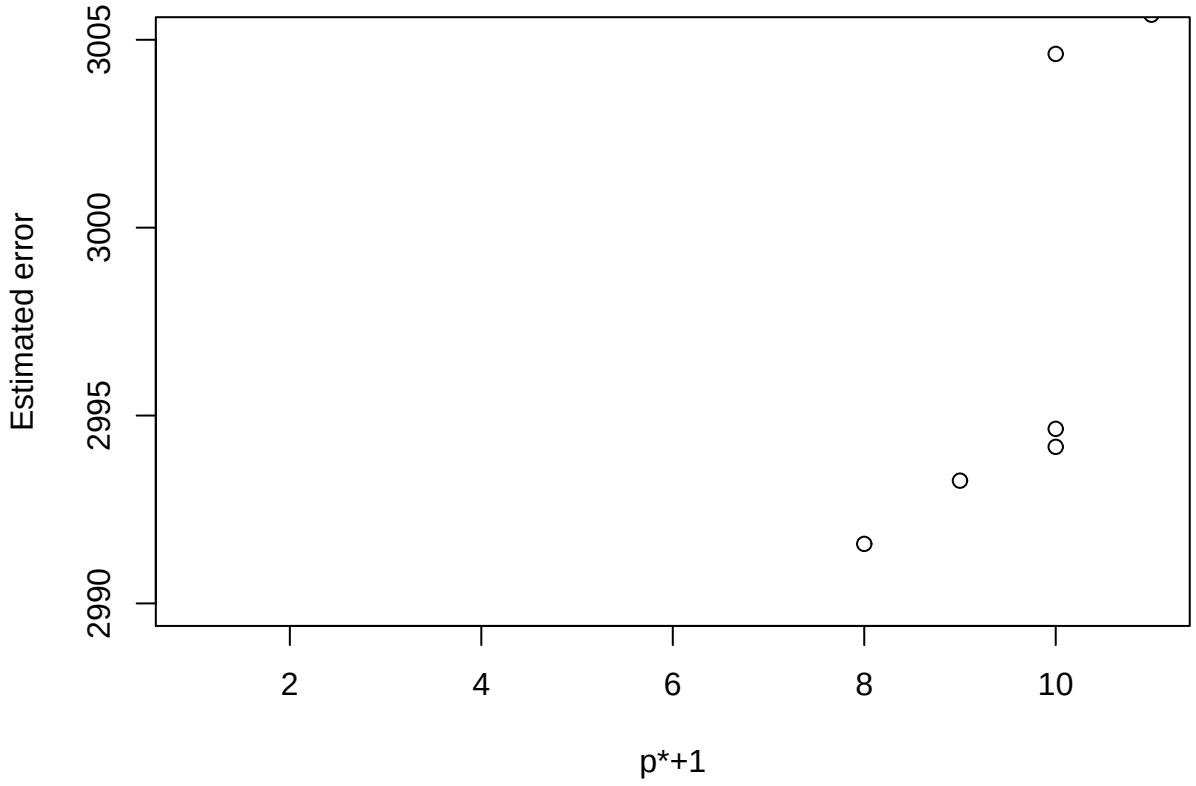
the ratio of the sum of magnitudes of the lasso estimates to that of the full least squares estimates. For $\lambda = 0$, this ratio is 1 (since the lasso = least squares), as λ increases to infinity, the ratio decreases to 0.

Starting at the right, where the ratio is 1, we have the least squares estimates of the coefficients. As λ increases, we move left. The coefficients generally shrink, until at some point one hits 0. That one stays at zero for a while, then goes negative. Continuing, the coefficients shrink, every so often one hits zero and stays there. Finally, at the far left, all the coefficients are 0.

The next plot shows $p^* + 1$ versus the estimated prediction error in (2.108). There are actually thirteen subsets, three with $p^* + 1 = 10$.



It is not easy to see, so we zoom in:



The best has $p^* + 1 = 8$, which means it has seven variables with nonzero coefficients. The corresponding estimated error is 2991.58. The best subset predictor had 6 variables.

The next table adds the lasso to table (2.86):

	Full LS	Subset	Lasso	Ridge
AGE	-10.01	0	0	-7.70
SEX	-239.82	-226.51	-197.75	-235.67
BMI	519.85	529.88	522.27	520.97
BP	324.38	327.22	297.15	321.31
$S1$	-792.18	-757.93	-103.95	-438.60
$S2$	476.74	538.58	0	196.36
$S3$	101.04	0	-223.92	-53.74
$S4$	177.06	0	0	136.33
$S5$	751.27	804.19	514.75	615.37
$S6$	67.63	0	54.77	70.41
$\sum \lvert\widehat{\beta}_j\rvert$	3459.98	3184.30	1914.56	2596.46
$\lVert\underline{\widehat{\beta}}\rVert$	1377.84	1396.56	853.86	1032.2
edf	11	7	8	10.38
$\overline{err}$	2859.70	2876.68	2885.42	2864.57
$penalty$	145.97	92.89	106.16	137.70
$\widehat{ERR}$	3005.67	2969.57	2991.58	3002.28

(2.111)

Again, the estimates of the first four coefficients are similar. Note that among the blood measurements, lasso sets #'s 2 and 4 to zero, while subset sets #'s 3, 4, and 6 to zero. Also, lasso's coefficient for $S3$ is unusually large.

Lasso is meant to keep down the sum of absolute values of the coefficients, which it does, and it also has the smallest norm. Performancewise, lasso is somewhere between ridge and subset. It still appears that the subset predictor is best in this example.

2.7.3 Using R

The `lars` program is in the `lars` package, so you must load it, or maybe install it and then load it. Using the normalized `x` and `y` from Section 2.6.3, fitting all the lasso predictors and plotting the coefficients is accomplished easily

```
diab.lasso <- lars(x,y)
plot(diab.lasso)
```

At each stage, represented by a vertical line on the plot, there is a set of coefficient estimates and the residual sum of squares. This data has 13 stages. The matrix `diab.lasso$beta` is a 13×10 matrix of coefficients, each row corresponding to the estimates for a given stage. To figure out p^*, you have to count the number of nonzero coefficients in that row:

```
pstar <- apply(diab.lasso$beta,1,function(z) sum(z!=0))
```

The $\widehat{\sigma}_e^2$ is the same as before, so

```
errhat <- diab.lasso$RSS/N+2*sigma2*(pstar+1)/N
errhat
       0        1        2        3        4        5        6        7
5943.155 5706.316 3886.784 3508.205 3156.248 3075.372 3054.280 2991.584
       8        9       10       11       12
2993.267 3004.624 2994.646 2994.167 3005.667
```

The smallest occurs at the eighth stage (note the numbering starts at 0). Plotting the $p^* + 1$ versus the estimated errors:

```
plot(pstar+1,errhat,xlab="p*+1",ylab="Estimated error")
 # or, zooming in:
plot(pstar+1,errhat,xlab="p*+1",ylab="Estimated error",ylim=c(2990,3005))
```

The estimates for the best predictor are then in

```
diab.lasso$beta[8,]
```

Chapter 3

Linear Predictors of Non-linear Functions

Chapter 2 assumed that the mean of the dependent variables was a linear function of the explanatory variables. In this chapter we will consider non-linear functions. We start with just one x-variable, and consider the model

$$Y_i = f(x_i) + e_i, \quad i = 1, \ldots, N, \tag{3.1}$$

where the x_i's are fixed, and the e_i's are independent with mean zero and variances σ_e^2. A linear model would have $f(x_i) = \beta_0 + \beta_1 x_i$. Here, we are not constraining f to be linear, or even any parametric function. Basically, f can be any function as long as it is sufficiently "smooth." Exactly what we mean by smooth will be detailed later. Some examples appear in Figure 3.1. It is obvious that these data sets do not show linear relationships between the x's and y's, nor is it particularly obvious what kind of non-linear relationships are exhibited.

From a prediction point of view, the goal is to find an estimated function of f, $\widehat{f}$, so that new y's can be predicted from new x's by $\widehat{f}(x)$. Related but not identical goals include

- Curve-fitting: fit a smooth curve to the data in order to have a good summary of the data; find a curve so that the graph "looks nice";
- Interpolation: Estimate y for values of x not among the observed, but in the same range as the observed;
- Extrapolation: Estimate y for values of x outside the range of the observed x's, a somewhat dangerous activity.

This chapter deals with "nonparametric" functions f, which strictly speaking means that we are not assuming a particular form of the function based on a finite number of parameters. Examples of parametric nonlinear functions:

$$f(x) = \alpha\ e^{\beta x} \quad \text{and} \quad f(x) = \sin(\alpha + \beta\ x + \gamma x^2). \tag{3.2}$$

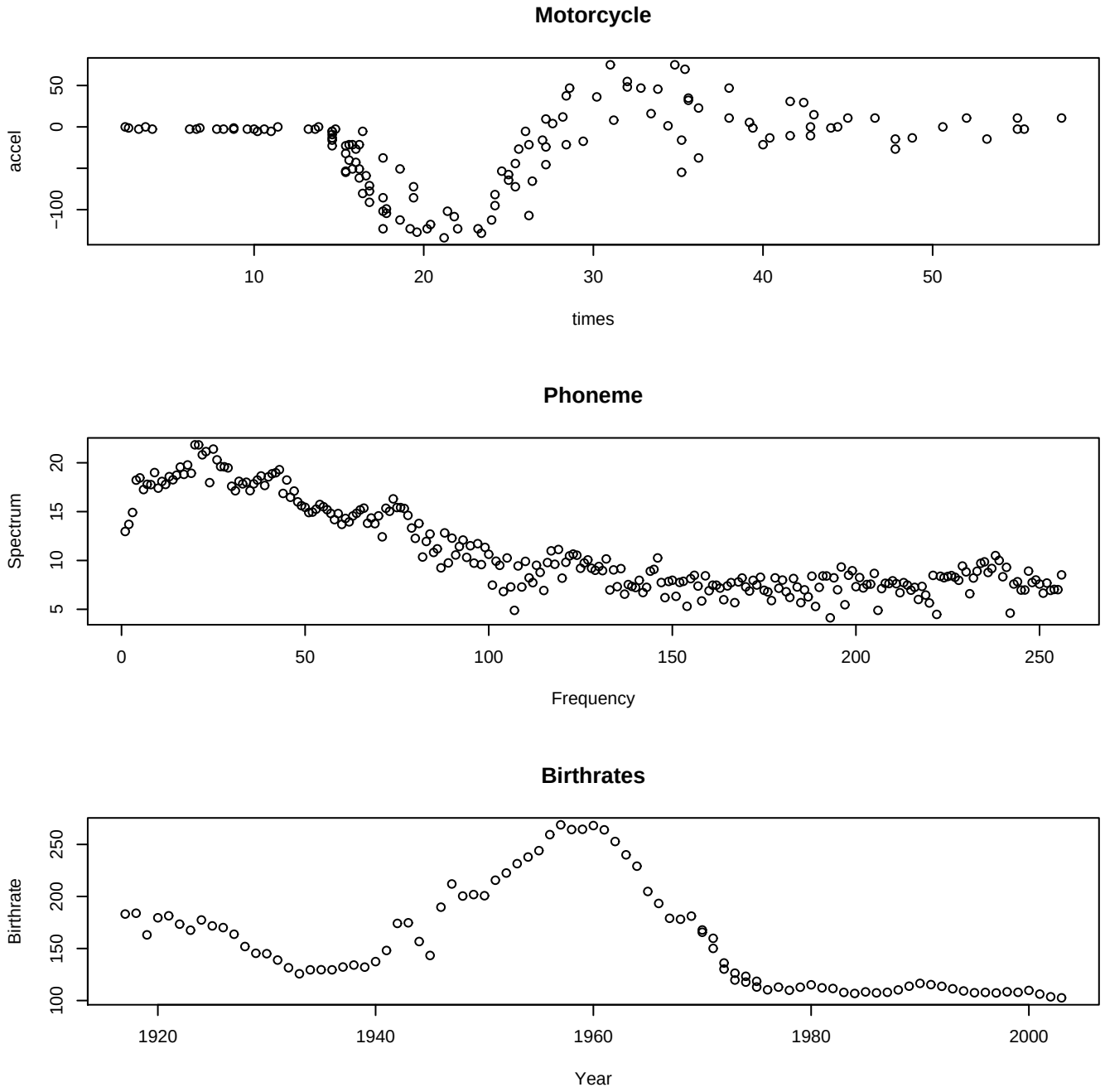


Figure 3.1: Examples of non-linear data

Such models can be fit with least squares much as the linear models, although the derivatives are not simple linear functions of the parameters, and Newton-Raphson or some other numerical method is needed.

The approach we take to estimating f in the nonparametric model is to use some sort of **basis expansion** of the functions on $\mathbb{R}$. That is, we have an infinite set of known functions, $h_1(x), h_2(x), \ldots$, and estimate f using a linear combination of a subset of the functions, e.g.,

$$\widehat{f}(x) = \widehat{\beta}_0 + \widehat{\beta}_1 h_1(x) + \cdots + \widehat{\beta}_m h_m(x). \tag{3.3}$$

We are *not* assuming that f is a finite linear combination of the h_j's, hence will always have a biased estimator of f. Usually we *do* assume that f can be arbitrarily well approximated by such a linear combination, that is, there is a sequence $\beta_0, \beta_1, \beta_2, \ldots$, such that

$$f(x) = \beta_0 + \lim_{m \to \infty} \sum_{i=1}^{m} \beta_j h_j(x) \tag{3.4}$$

uniformly, at least for x in a finite range.

An advantage to using estimates as in (3.3) is that the estimated function is a linear one, linear in the h_j's, though not in the x. But with x_i's fixed, we are in the same estimation bailiwick as the linear models in Chapter 2, hence ideas such as subset selection, ridge, lasso, and estimating prediction errors carry over reasonably easily.

In the next few sections, we consider possible sets of h_j's, including polynomials, sines and cosines, splines, and wavelets.

3.1 Polynomials

The estimate of f is a polynomial in x, where the challenge is to figure out the degree. In raw form, we have

$$h_1(x) = x,\ h_2(x) = x^2,\ h_3(x) = x^3, \ldots. \tag{3.5}$$

(The Weierstrass Approximation Theorem guarantees that (3.4) holds.) The m^{th} degree polynomial fit is then

$$\widehat{f}(x) = \widehat{\beta}_0 + \widehat{\beta}_1 x + \widehat{\beta}_2 x^2 + \cdots + \widehat{\beta}_m x^m. \tag{3.6}$$

It is straightforward to find the estimates $\widehat{\beta}_j$ using the techniques from the previous chapter, where here

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^m \end{pmatrix}. \tag{3.7}$$

Technically, one could perform a regular subset regression procedure, but generally one considers only fits allowing the first m coefficients to be nonzero, and requiring the rest to

be zero. Thus the only fits considered are

$$\begin{array}{rcll} \widehat{f}(x_i) & = & \widehat{\beta}_0 & \text{(constant)} \\ \widehat{f}(x_i) & = & \widehat{\beta}_0 + \widehat{\beta}_1 x_i & \text{(linear)} \\ \widehat{f}(x_i) & = & \widehat{\beta}_0 + \widehat{\beta}_1 x_i + \widehat{\beta}_2 x_i^2 & \text{(quadratic)} \\ & \vdots & & \\ \widehat{f}(x_i) & = & \widehat{\beta}_0 + \widehat{\beta}_1 x_i + \widehat{\beta}_2 x_i^2 + \cdots + \widehat{\beta}_m x_i^m & (m^{th} \text{ degree}) \\ & \vdots & & \end{array} \tag{3.8}$$

We will use the birthrate data to illustrate polynomial fits. The x's are the years from 1917 to 2003, and the y's are the births per 10,000 women aged twenty-three in the U.S.[1]

Figure 3.2 contains the fits of several polynomials, from a cubic to an 80^{th}-degree polynomial. It looks like the $m = 3$ and $m = 5$ fits are poor, $m = 20$ to 40 are reasonable, and $m = 80$ is overfitting, i.e., the curve is too jagged.

If you are mainly interested in a good summary, you would choose your favorite fit visually. For prediction, we proceed as before by trying to estimate the prediction error. As for subset regression in (2.59), the estimated in-sample prediction error for the m^{th}-degree polynomial fit is, since $p^* = m$,

$$\widehat{ERR}_{in}^m = \overline{err}^m + 2\widehat{\sigma}_e^2 \ \frac{m+1}{N}. \tag{3.9}$$

The catch is that $\widehat{\sigma}_e^2$ is the residual variance from the "full" model, where here the full model has $m = \infty$ (or at least $m = N-1$). Such a model fits perfectly, so the residuals and residual degrees of freedom will all be zero. There are several ways around this problem:

Specify an upper bound M for m. You want the residual degrees of freedom, $N-M-1$, to be sufficiently large to estimate the variance reasonably well, but M large enough to fit the data. For the birthrate data, you might take $M = 20$ or 30 or 40 or 50. It is good to take an M larger than you think is best, because you can count on the subset procedure to pick a smaller m as best. For $M = 50$, the $\widehat{\sigma}_e^2 = 59.37$.

Find the value at which the residual variances level off. For each m, find the residual variance. When the m in the fit is larger than or equal to the true polynomial degree, then the residual variance will be an unbiased estimator of σ_e^2. Thus as a function of m, the

[1]The data up through 1975 can be found in the Data and Story Library at `http://lib.stat.cmu.edu/DASL/Datafiles/Birthrates.html`. See Velleman, P. F. and Hoaglin, D. C. (1981). *Applications, Basics, and Computing of Exploratory Data Analysis*. Belmont. CA: Wadsworth, Inc. The original data is from P.K. Whelpton and A. A. Campbell, "Fertility Tables for Birth Charts of American Women," Vital Statistics Special Reports 51, no. 1. (Washington D.C.:Government Printing Office, 1960, years 1917-1957) and National Center for Health Statistics, Vital Statistics of the United States Vol. 1, Natality (Washington D.C.:Government Printing Office, yearly, 1958-1975). The data from 1976 to 2003 are actually rates for women aged 20-24, found in the National Vital Statistics Reports Volume 54, Number 2 September 8, 2005, *Births: Final Data for 2003*; `http://www.cdc.gov/nchs/data/nvsr/nvsr54/nvsr54_02.pdf`.

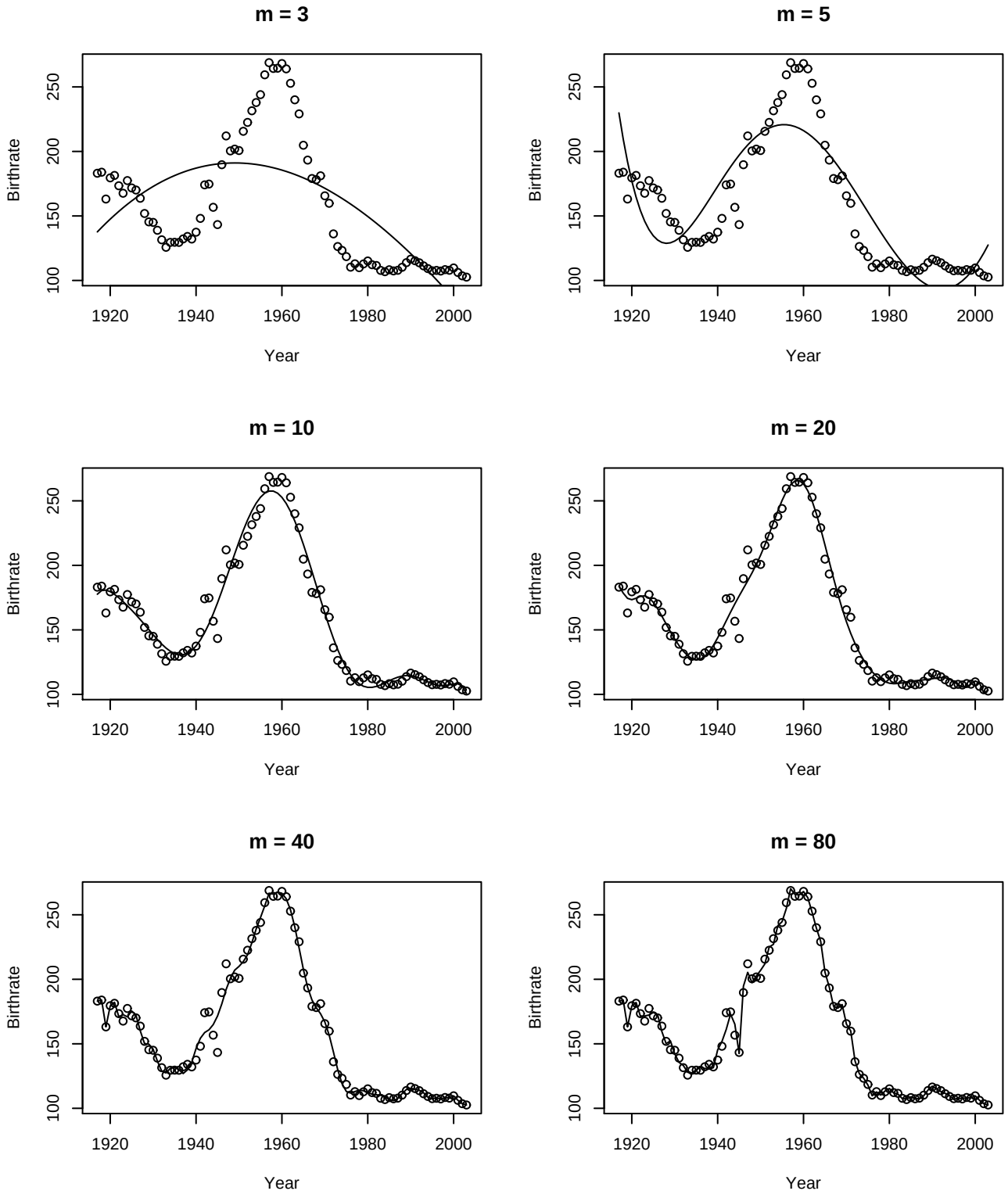


Figure 3.2: Polynomial fits to the Birthrate data

residual variance should settle to the right estimate. Figure 3.3 show the graph for the birthrate data. The top plot show that the variance is fairly constant from about $m = 15$ to $m = 70$. The bottom plot zooms in, and we see that from about $m = 20$ to 70, the variance is bouncing around 60.

Use the local variance. Assuming that Y_1 and Y_2 are independent with the same mean and same variance σ_e^2,

$$\frac{1}{2}\, E[(Y_1 - Y_2)^2] = \frac{1}{2}\, Var[Y_1 - Y_2] = \sigma_e^2. \tag{3.10}$$

If the f is not changing too quickly, then consecutive Y_i's will have approximately the same means, hence an estimate of the variance is

$$\widehat{\sigma}_e^2 = \frac{1}{2}\, \frac{1}{N-1}\, \sum_{i=1}^{N-1} (y_i - y_{i+1})^2. \tag{3.11}$$

For the birthrate data, this estimate is 50.74.

Whichever approach we take, the estimate of the residual variance is around 50 to 60. We will use 60 as the estimate in what follows, but you are welcome to try some other values. Figure 3.4 has the estimated prediction errors. The minimum 72.92 occurs at $m = 26$, but we can see that the error at $m = 16$ is almost as low at 73.18, and is much simpler, so either of those models is reasonable (as is the $m = 14$ fit). Figure 3.5 has the fits. The $m = 16$ fit is smoother, and the $m = 26$ fit is closer to some of the points.

Unfortunately, polynomials are not very good for extrapolation. Using the two polynomial fits, we have the following extrapolations.

Year	$m = 16$	$m = 26$	Observed
1911	3793.81	−2554538.00	
1912	1954.05	−841567.70	
1913	993.80	−246340.00	
1914	521.25	−61084.75	
1915	305.53	−11613.17	
1916	216.50	−1195.05	
1917	184.82	183.14	183.1
2003	102.72	102.55	102.6
2004	123.85	−374.62	
2005	209.15	−4503.11	
2006	446.58	−26035.92	
2007	1001.89	−112625.80	
2008	2168.26	−407197.50	

(3.12)

The ends of the data are 1917 and 2003, for which both predictors are quite good. As we move away from those dates, the 16^{th}-degree polynomial deteriorates after getting a few years away from the data, while the 26^{th}-degree polynomial gets ridiculous right away. The actual (preliminary) birthrate for 2004 is 101.8. The two fits did not predict this value well.

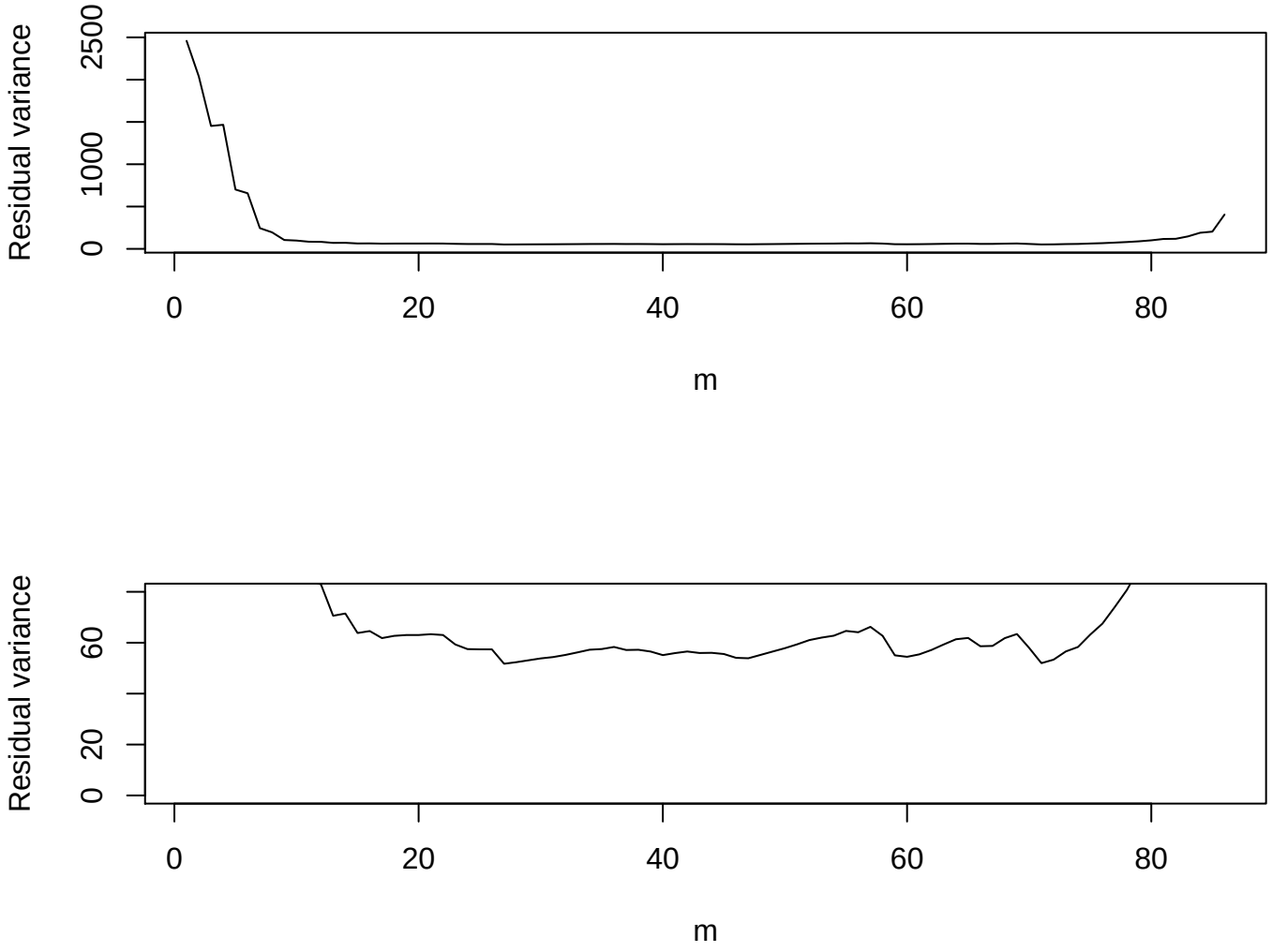


Figure 3.3: Residual variance as a function of $m + 1$

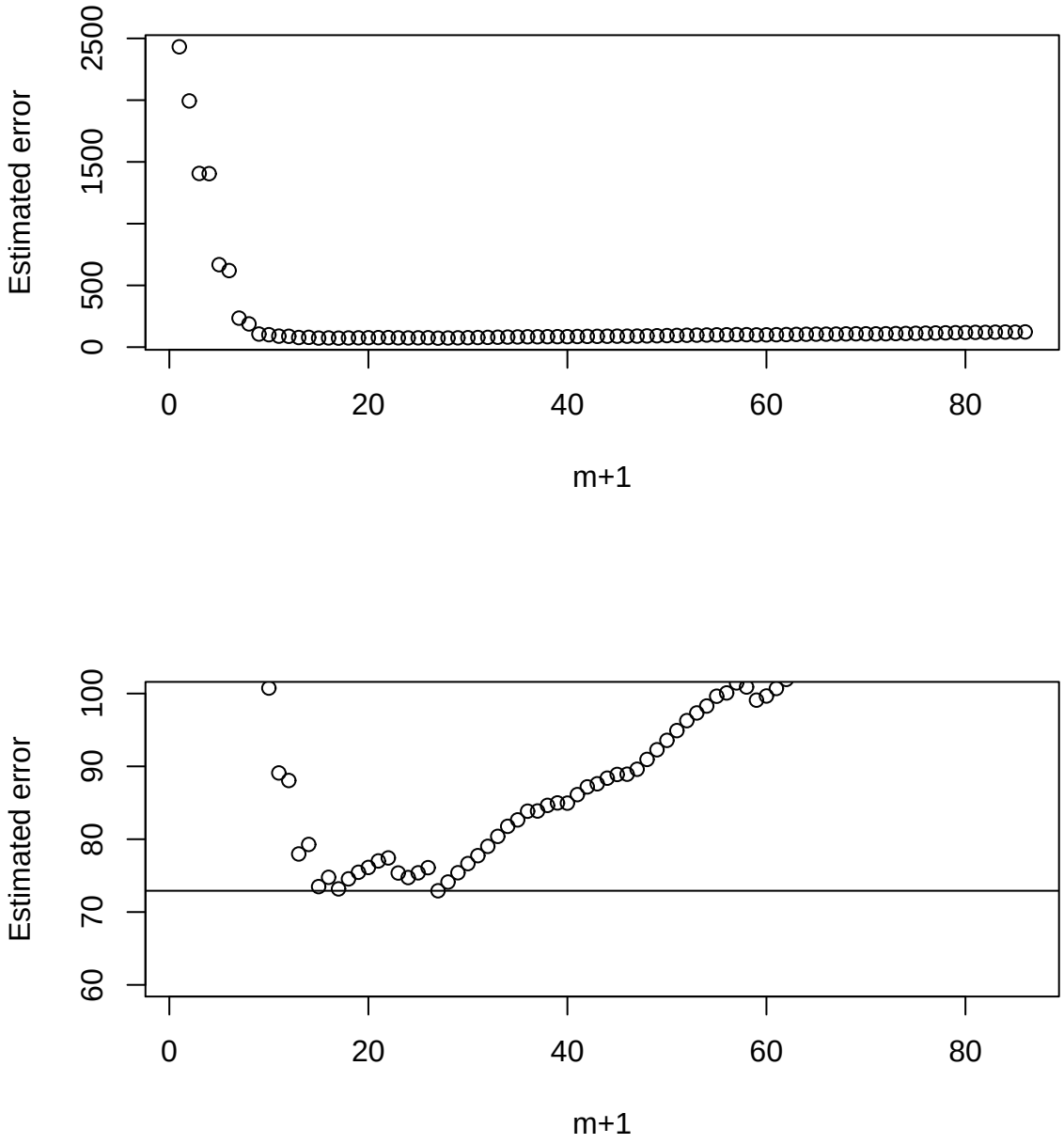


Figure 3.4: Estimated prediction error as a function of $m + 1$

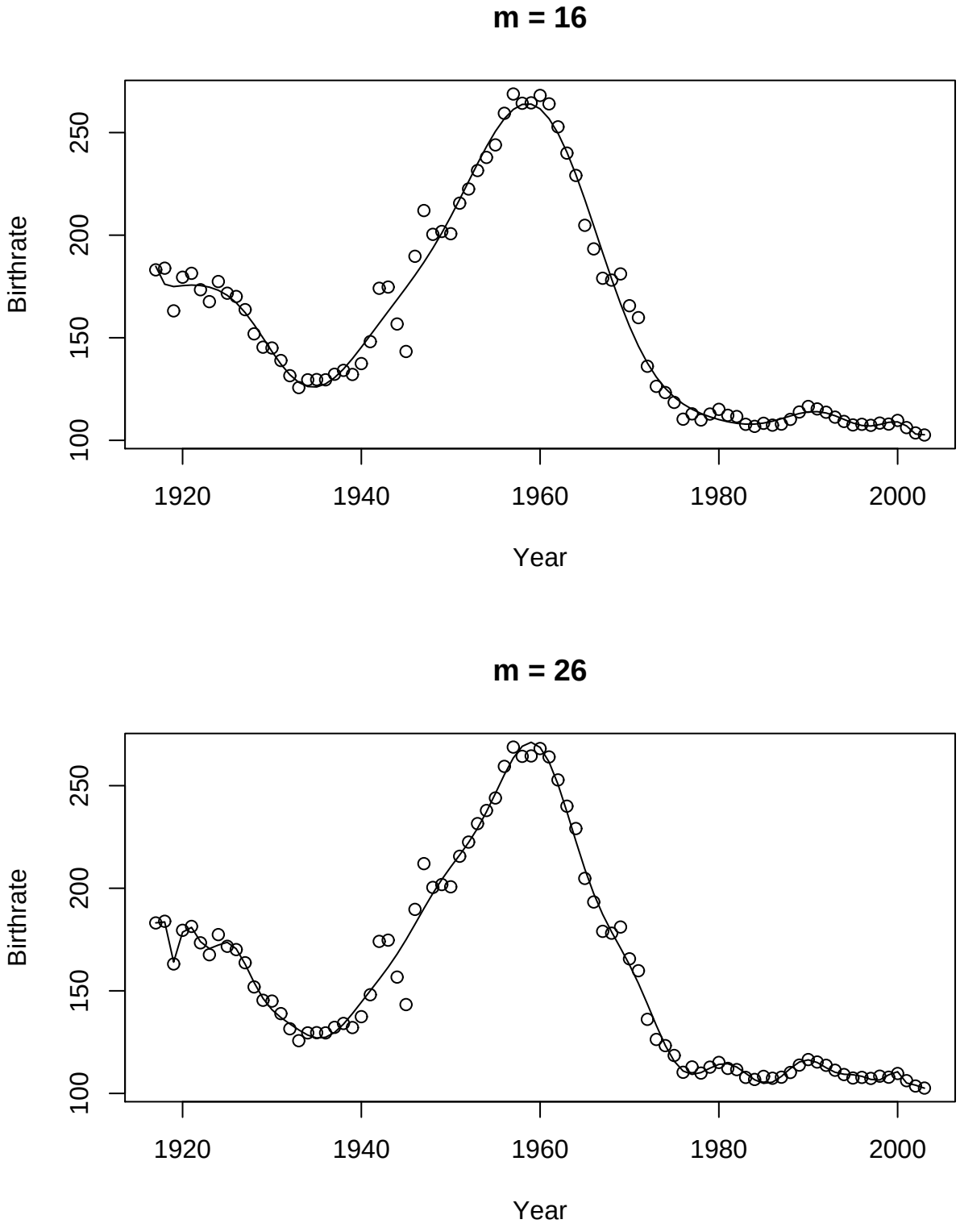


Figure 3.5: Estimated predictions

3.1.1 Leave-one-out cross-validation

Another method for estimating prediction error is **cross-validation**. It is more widely applicable than the method we have used so far, but can be computationally intensive. The problem with using the observed error for estimating the prediction error is that the observed values are used for creating the prediction, hence are not independent. One way to get around that problem is to set aside the first observation, say, then try to predict it using the other $N-1$ observations. We are thus using this first observation as a "new" observation, which is indeed independent of its prediction. We repeat leaving out just the second observation, then just the third, etc.

Notationally, let $[-i]$ denote calculations leaving out the i^{th} observation, so that

$$\widehat{Y}_i^{[-i]} \text{ is the prediction of } Y_i \text{ based on the data without } (x_i, Y_i). \tag{3.13}$$

The **leave-one-out cross-validation** estimate of the prediction error is then

$$\widehat{ERR}_{in,cv} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \widehat{y}_i^{[-i]})^2. \tag{3.14}$$

It has the advantage of not needing a preliminary estimate of the residual error.

For least squares estimates, the cross-validation prediction errors can be calculated quite simply by the dividing the regular residuals by a factor,

$$y_i - \widehat{y}_i^{[-i]} = \frac{y_i - \widehat{y}_i}{1 - h_{ii}}, \tag{3.15}$$

where $\widehat{y}$ is from the fit to all the observations, and h_{ii} is i^{th} the diagonal of the $H = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ matrix. (See Section 3.1.3.) Note that for each m, there is a different $\underline{\widehat{y}}$ and H.

To choose a polynomial fit using cross-validation, one must find the $\widehat{ERR}_{in,cv}^{m}$ in (3.14) for each m. Figure 3.6 contains the results. It looks like $m = 14$ is best here. The estimate of the prediction error is 85.31.

Looking more closely at the standardized residuals, a funny detail appears. Figure 3.7 plots the standardized residuals for the $m = 26$ fit. Note that the second one is a huge negative outlier. It starts to appear for $m = 20$ and gets worse. Figure 3.8 recalculates the cross-validation error without that residual. Now $m = 27$ is the best, with an estimated prediction error of 58.61, compared to the estimate for $m = 26$ of 71.64 using (3.9). These two estimates are similar, suggesting $m = 26$ or 27. The fact that the cross-validation error is smaller may be due in part to having left out the outlier.

3.1.2 Using R

The $\mathbf{X}$ matrix in (3.7) is not the one to use for computations. For any high-degree polynomial, we will end up with huge numbers (e.g., $87^{16} \approx 10^{31}$) and small numbers in the matrix,

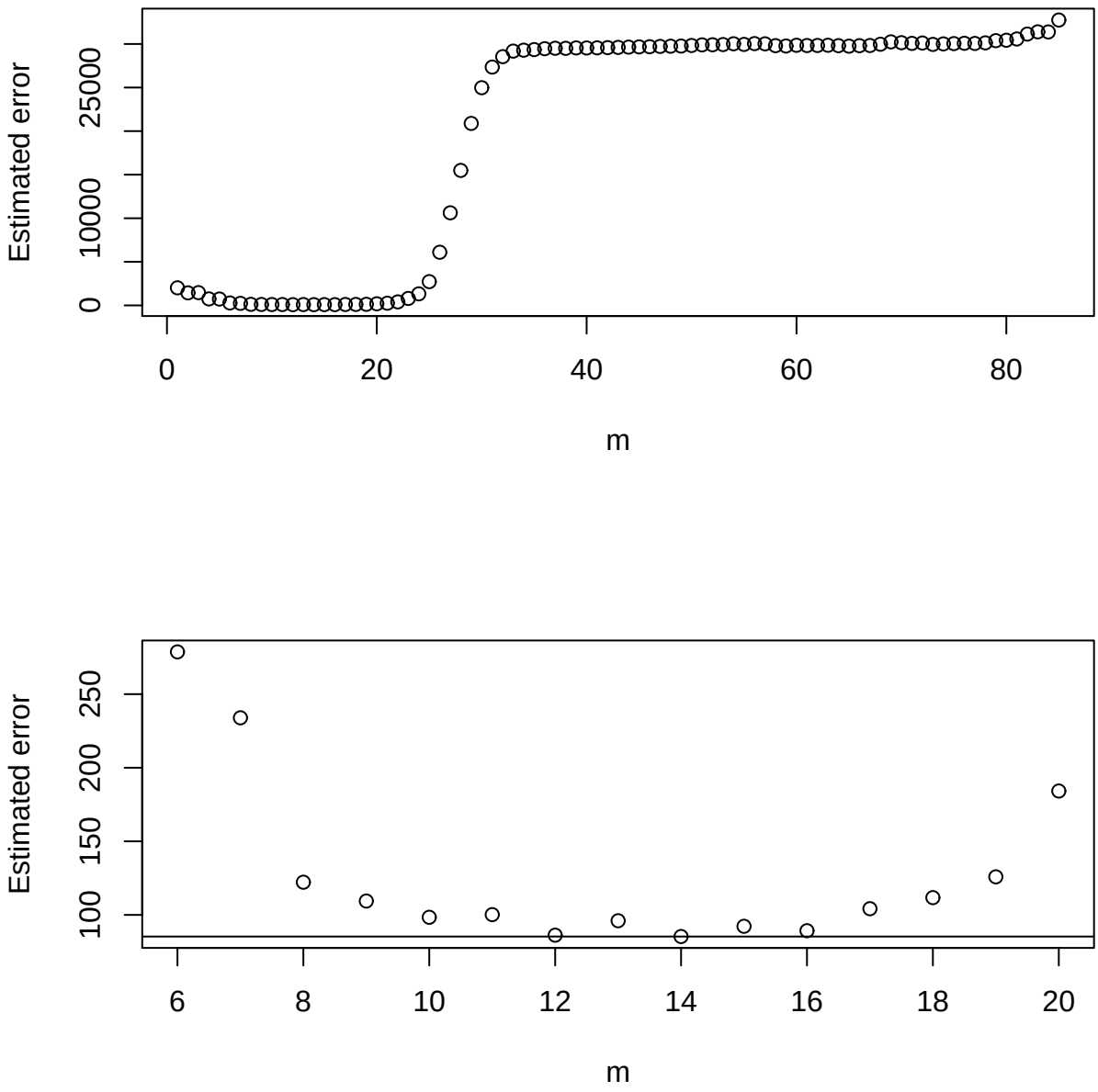


Figure 3.6: Cross-validation estimates of prediction error

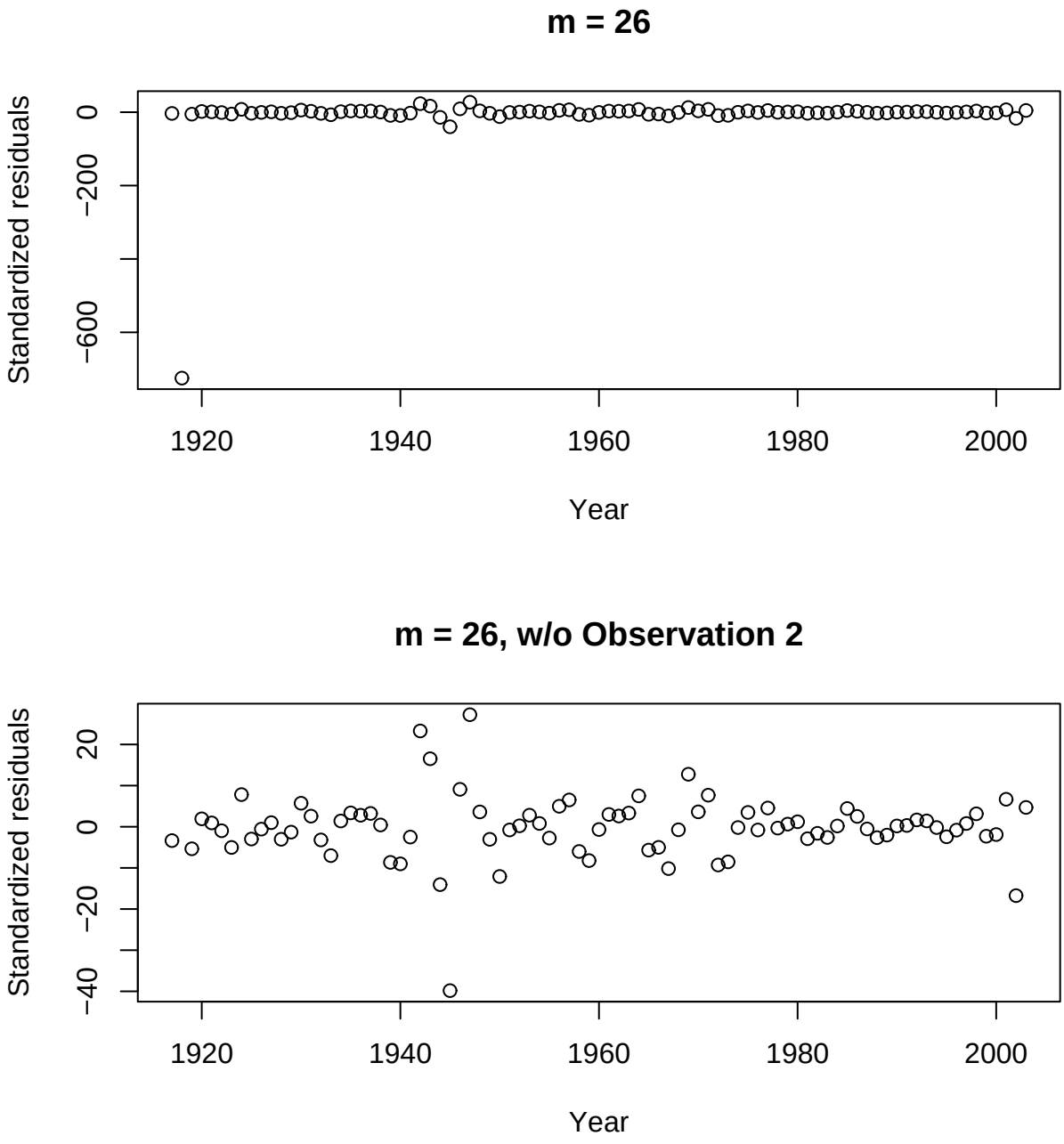


Figure 3.7: Standardized residuals for $m = 26$

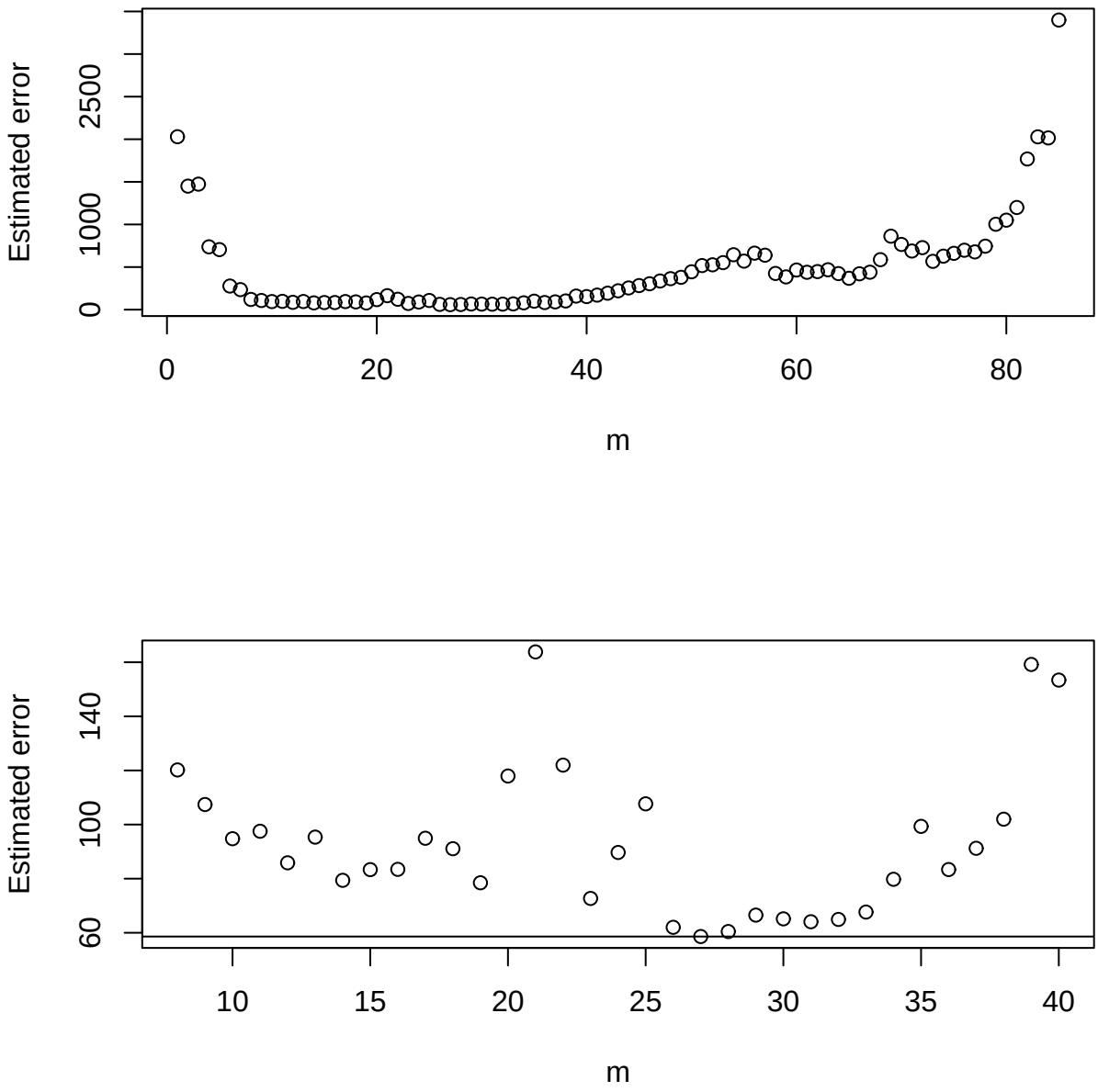


Figure 3.8: Cross-validation estimates of prediction error without observation #2

which leads to numerical inaccuracies. A better matrix uses *orthogonal polynomials*, or in fact orthonormal ones. Thus we want the columns of the $\mathbf{X}$ matrix, except for the $\underline{1}_N$, to have mean 0 and norm 1, but also to have them orthogonal to each other in such a way that the first m columns still yield the m^{th}-degree polynomials. To illustrate, without going into much detail, we use the Gram-Schmidt algorithm for $\underline{x} = (1, 2, 3, 4, 5)'$ and $m = 2$. Start with the raw columns,

$$\underline{1}_5 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \underline{x}_{[1]} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}, \quad \text{and} \quad \underline{x}_{[2]} = \begin{pmatrix} 1 \\ 4 \\ 9 \\ 16 \\ 25 \end{pmatrix}. \tag{3.16}$$

Let the first one as is, but subtract the means (3 and 11) from each of the other two:

$$\underline{x}^{(2)}_{[1]} = \begin{pmatrix} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{pmatrix}, \quad \text{and} \quad \underline{x}^{(2)}_{[2]} = \begin{pmatrix} -10 \\ -7 \\ -2 \\ 5 \\ 14 \end{pmatrix}. \tag{3.17}$$

Now leave the first two alone, and make the third orthogonal to the second by applying the main Gram-Schmidt step,

$$\underline{u} \to \underline{u} - \frac{\underline{u}'\underline{v}}{\underline{v}'\underline{v}}\,\underline{v}, \tag{3.18}$$

with $\underline{v} = \underline{x}^{(2)}_{[1]}$ and $\underline{u} = \underline{x}^{(2)}_{[2]}$:

$$\underline{x}^{[3]}_{[2]} = \begin{pmatrix} -10 \\ -7 \\ -2 \\ 5 \\ 14 \end{pmatrix} - \frac{60}{10} \begin{pmatrix} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ -2 \\ -1 \\ 2 \end{pmatrix}. \tag{3.19}$$

To complete the picture, divide the last two x's by their respective norms, to get

$$\underline{1}_5 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \underline{x}^{Norm}_{[1]} = \frac{1}{\sqrt{10}} \begin{pmatrix} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{pmatrix}, \quad \text{and} \quad \underline{x}^{Norm}_{[2]} = \frac{1}{\sqrt{14}} \begin{pmatrix} -10 \\ -7 \\ -2 \\ 5 \\ 14 \end{pmatrix}. \tag{3.20}$$

You can check that indeed these three vectors are orthogonal, and the last two orthonormal.

For a large N and m, you would continue, at step k orthogonalizing the current $(k+1)^{st}, \ldots, (m+1)^{st}$ vector to the current k^{th} vector. Once you have these vectors, then the

fitting is easy, because the $\mathbf{X}_m$ for the m^{th} degree polynomial (leaving out the $\underline{1}_N$) just uses the first m vectors, and $\mathbf{X}_m'\mathbf{X}_m = \mathbf{I}_m$, so that the estimates of beta are just $\mathbf{X}_m'\underline{y}$, and the $\mathbf{H}_m = \mathbf{X}_m\mathbf{X}_m'$. Using the saturated model, i.e., $(N-1)^{st}$-degree, we can get all the coefficients at once,

$$\underline{\widehat{\beta}} = \mathbf{X}_{N-1}'\underline{y}, \quad (\widehat{\beta}_0 = \overline{y}). \tag{3.21}$$

Then the coefficients for the m^{th} order fit are the first m elements of $\underline{\widehat{\beta}}$. Also, the residual sum of squares equals the sum of squares of the left-out coefficients:

$$RSS^m = \sum_{j=m+1}^{N-1} \widehat{\beta}_j^2, \tag{3.22}$$

from which it is easy to find the residual variances, $\overline{err}^m$'s, and estimated prediction errors.

The following commands in R will read in the data and find the estimated coefficients and predicted y's.

```
source("http://www.stat.uiuc.edu/~jimarden/birthrates.txt")

N <- 87
x <- 1:87
y <- birthrates[,2]

xx <- poly(x,86)

betah <- t(xx)%*%y

yhats <- sweep(xx,2,betah,"*")
yhats <- t(apply(yhats,1,cumsum)) + mean(y)
# yhats[,m] has the fitted y's for the m-th degree polynomial.
```

Plot the 16^{th}-degree fit:

```
m <- 16
plot(birthrates,main=paste("m =",m))
lines(x+1916,yhats[,m])
```

You can plot all the fits sequentially using

```
for(m in 1:86) {
   plot(birthrates,main=paste("m =",m))
   lines(x+1916,yhats[,m])
   readline()
}
```

where you hit the enter key to see the next one.

Next, use (3.22) to find the residual sums of squares, and plot the sequence of residual variances.

```
rss <- cumsum(betah[86:1]^2)[86:1]
sigma2hat <- rss/(86:1)
par(mfrow=c(2,1))
plot(1:86,sigma2hat,xlab="m+1",ylab="Residual variance",type='l')
plot(1:86,sigma2hat,xlab="m+1",ylab="Residual variance",type='l',ylim=c(0,70))
```

Using $\widehat{\sigma}_e^2 = 60$, plot the $\widehat{ERR}_{in}^m$'s:

```
sigma2hat <- 60
errhat <- rss/N + 2*sigma2hat*(1:86)/N
plot(1:86,errhat,xlab="m+1",ylab="Estimated error")
plot(1:86,errhat,xlab="m+1",ylab="Estimated error",ylim=c(60,100))
abline(h=min(errhat))
```

The prediction of y for x's outside the range of the data is somewhat difficult when using orthogonal polynomials since one does not know what they are for new x's. Fortunately, the function `predict` can help. To find the $\mathbf{X}$ matrices for values $-5, -1, \ldots, 0, 1, 87, 88, \ldots 92$, use

```
z<-c((-5):1,87:92)
x16 <- predict(poly(x,16),z)
p16 <- x16%*%betah[1:16]+mean(y)

x26 <- predict(poly(x,26),z)
p26 <- x26%*%betah[1:26]+mean(y)
```

The `p16` and `p26` then contain the predictions for the fits of degree 16 and 26, as in (3.12).

For the cross-validation estimates, we first obtain the h_{ii}'s, N of them for each m. The following creates an $N \times 85$ matrix `hii`, where the m^{th} column has the h_{ii}'s for the m^{th}-degree fit.

```
hii <- NULL

for(m in 1:85) {
h <- xx[,1:m]%*%t(xx[,1:m])
hii <- cbind(hii,diag(h))
}
```

Then find the regular residuals, (3.15)'s, called `sresids`, and the cross-validation error estimates (one for each m):

```
resids <- - sweep(yhats[,-86],1,y,"-")
sresids <- resids/(1-hii)
errcv <- apply(sresids^2,2,mean)
```

```
plot(1:85,errcv,xlab="m",ylab="Estimated error")
plot(6:20,errcv[6:20],xlab="m",ylab="Estimated error")
abline(h=min(errcv))
```

The errors look too big for m's over 25 or so. For example,

```
plot(x+1916,sresids[,26],xlab="Year",ylab="Standardized residuals",main= "m = 26")
plot((x+1916)[-2],sresids[-2,26],xlab="Year",ylab="Standardized residuals",
                main= "m = 26, w/o Observation 2")
```

Recalculating the error estimate leaving out the second residual yields

```
errcv2 <- apply(sresids[-2,]^2,2,mean)
plot(1:85,errcv2,xlab="m",ylab="Estimated error")
plot(8:40,errcv2[8:40],xlab="m",ylab="Estimated error")
abline(h=min(errcv2))
```

3.1.3 The cross-validation estimate

We will consider the leave-one-out prediction error for the first observation $(y_1, \underline{x}_1)$. The others works similarly. Indicate by "$[-1]$" the quantities without the first observations, so that

$$\underline{y}^{[-1]} = \begin{pmatrix} y_2 \\ y_3 \\ \vdots \\ y_N \end{pmatrix}, \ \mathbf{X}^{[-1]} = \begin{pmatrix} \underline{x}_2' \\ \underline{x}_3' \\ \vdots \\ \underline{x}_N' \end{pmatrix}, \widehat{\underline{\beta}}^{[-1]} = (\mathbf{X}^{[-1]'}\mathbf{X}^{[-1]})^{-1}\mathbf{X}^{[-1]'}\underline{y}^{[-1]}, \ \text{ and } \ \widehat{y}_1^{[-1]} = \underline{x}_1'\widehat{\underline{\beta}}^{[-1]}. \tag{3.23}$$

We know that $\widehat{\underline{\beta}}^{[-1]}$ is the $\underline{b}$ that minimizes

$$\|\underline{y}^{[-1]} - \mathbf{X}^{[-1]}\underline{b}\|^2, \tag{3.24}$$

but it is also the $\underline{b}$ that minimizes

$$(\widehat{y}_1^{[-1]} - \underline{x}_1'\underline{b})^2 + \|\underline{y}^{[-1]} - \mathbf{X}^{[-1]}\underline{b}\|^2 \tag{3.25}$$

because it minimizes the first term (since it is zero when $\underline{b} = \widehat{\underline{\beta}}^{[-1]}$) as well as the second term. Adding the terms, we have that $\widehat{\underline{\beta}}^{[-1]}$ is the $\underline{b}$ that minimizes

$$\left\| \begin{pmatrix} \widehat{y}_1^{[-1]} \\ y_2 \\ \vdots \\ y_N \end{pmatrix} - \mathbf{X}\underline{b} \right\|^2. \tag{3.26}$$

Thus it must be that

$$\underline{\widehat{\beta}}^{[-1]} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' \begin{pmatrix} \widehat{y}_1^{[-1]} \\ y_2 \\ \vdots \\ y_N \end{pmatrix}, \quad \text{hence} \quad \begin{pmatrix} \widehat{y}_1^{[-1]} \\ \widehat{y}_2 \\ \vdots \\ \widehat{y}_N \end{pmatrix} = \mathbf{H} \begin{pmatrix} \widehat{y}_1^{[-1]} \\ y_2 \\ \vdots \\ y_N \end{pmatrix}. \tag{3.27}$$

Using the first row of $\mathbf{H}$, we have

$$\widehat{y}_1^{[-1]} = h_{11}\widehat{y}_1^{[-1]} + h_{12}y_2 + \cdots + h_{1N}y_N. \tag{3.28}$$

Solving,

$$\widehat{y}_1^{[-1]} = \frac{h_{12}y_2 + \cdots + h_{1N}y_N}{1 - h_{11}}. \tag{3.29}$$

The cross-validation error estimate for the first observation is then

$$\begin{aligned} y_1 - \widehat{y}_1^{[-1]} &= y_1 - \frac{h_{12}y_2 + \cdots + h_{1N}y_N}{1 - h_{11}} \\ &= \frac{y_1 - (h_{11}y_1 + h_{12}y_2 + \cdots + h_{1N}y_N)}{1 - h_{11}} \\ &= \frac{y_1 - \widehat{y}_1}{1 - h_{11}}, \end{aligned} \tag{3.30}$$

where $\widehat{y}_1$ is the regular fit using all the data, which is (3.15).

The equation (3.27) is an example of the *missing information principle* for imputing missing data. Supposing y_1 is missing, we find a value for y_1 such that the value and its fit are the same. In this case, that value is the $\widehat{y}_1^{[-1]}$.

3.2 Sines and cosines

Data collected over time often exhibits cyclical behavior, such as the outside temperature over the course of a year. To these one may consider fitting sine waves. Figure 3.9 shows sine waves of various frequencies covering 133 time points. The frequency is the number of complete cycles of the sine curve, so that this figure shows frequencies of 1 (at the top), 2, 3, 4, and 5. More than one frequency can be fit. In the temperature example, if one has hourly temperature readings over the course of a year, it would be reasonable to expect a large sine wave of frequency 1, representing the temperature cycle over the course of the year, and a smaller sine wave of frequency 365, representing the temperature cycle over the course of a day.

In this section we will focus on the motorcycle acceleration data[2], exhibited in the top panel of Figure 3.1. It can be found in the MASS package for *R*. Here is the description:

[2]See Silverman, B. W. (1985) Some aspects of the spline smoothing approach to non-parametric curve fitting. *Journal of the Royal Statistical Society Series B* **47**, 1-52.

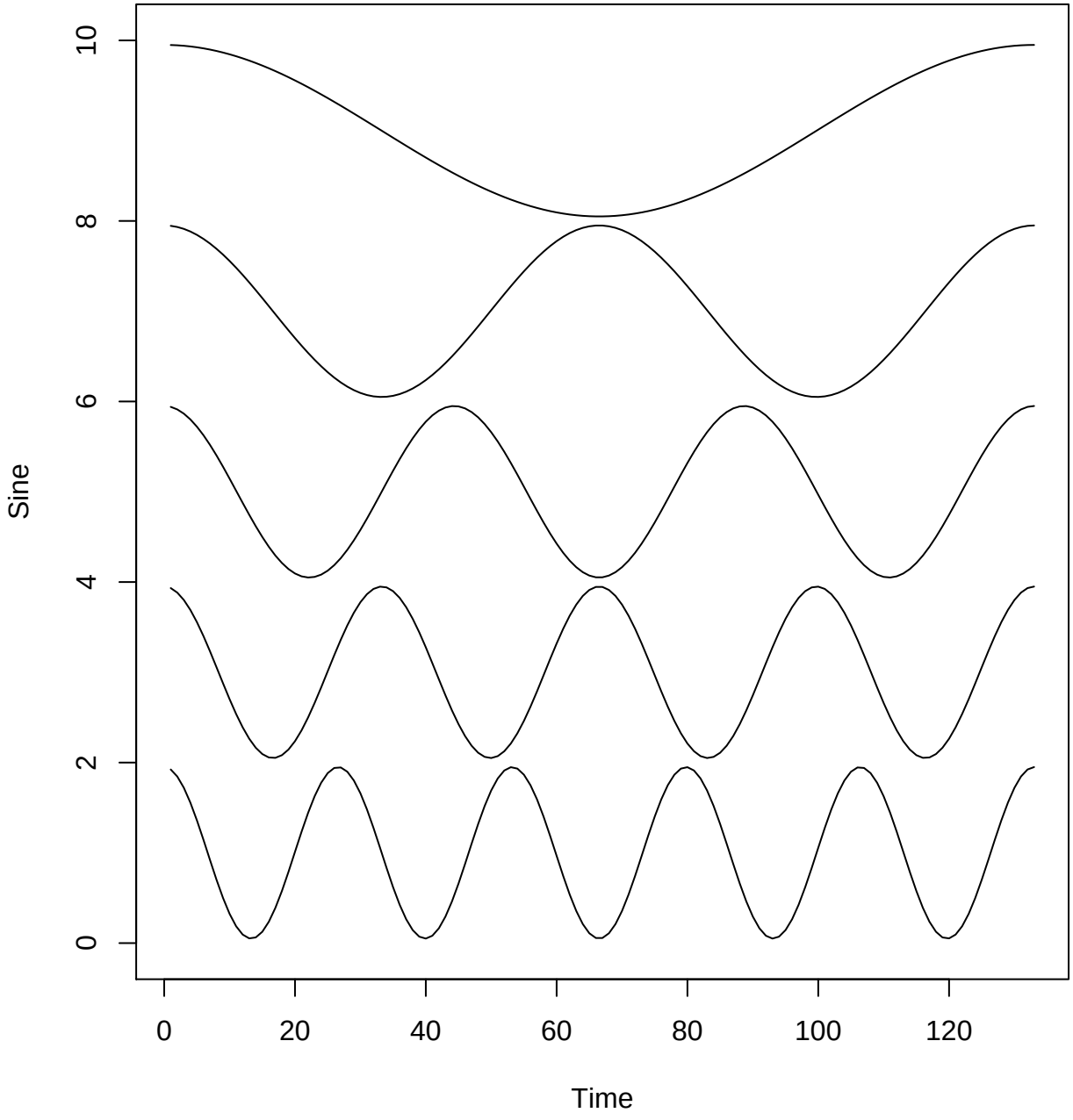


Figure 3.9: Some sine waves

```
Description:

    A data frame giving a series of measurements of head acceleration
    in a simulated motorcycle accident, used to test crash helmets.

Format:

    'times' in milliseconds after impact

    'accel' in g
```

The x-variable, time, is not exactly equally spaced, but we will use equally spaced time points as an approximation: $1, 2, \ldots, N$, where $N = 133$ time points. We will fit a number of sine waves; deciding on which frequencies to choose is the challenge.

A sine wave not only has a frequency, but also an amplitude α (the maximum height) and a phase ϕ. That is, for frequency k, the values of the sign curve at the data points are

$$\alpha \sin(\frac{2\pi i}{N}\ k + \phi), \quad i = 1, 2, \ldots, N. \tag{3.31}$$

The equation as written is not linear in the parameters, α and ϕ. But we can rewrite it so that it is linear:

$$\begin{aligned} \alpha \sin(\frac{2\pi i}{N}\ k + \phi) &= \alpha \sin(\frac{2\pi i}{N}\ k) \cos(\phi) + \alpha \cos(\frac{2\pi i}{N}\ k) \sin(\phi) \\ &= \beta_{k1} \sin(\frac{2\pi i}{N}\ k) + \beta_{k2} \cos(\frac{2\pi i}{N}\ k), \end{aligned} \tag{3.32}$$

where the new parameters are the inverse polar coordinates of the old ones,

$$\beta_{k1} = \alpha \cos(\phi) \ \text{ and } \ \beta_{k2} = \alpha \sin(\phi). \tag{3.33}$$

Now (3.32) is linear in (β_{k1}, β_{k2}).

For a particular fit, let $\mathcal{K}$ be the set of frequencies used in the fit. Then we fit the data via

$$\widehat{y}_i = \widehat{\beta}_0 + \sum_{k \in \mathcal{K}} \left(\widehat{\beta}_{k1} \sin(\frac{2\pi i}{N}\ k) + \widehat{\beta}_{k2} \cos(\frac{2\pi i}{N}\ k) \right). \tag{3.34}$$

Note that for each frequency k, we either have both sine and cosine in the fit, or both out. Only integer frequencies $k \leq (N-1)/2$ are going to be used. When N is odd, using all those frequencies will fit the data exactly.

Suppose there are just two frequencies in the fit, k and l. Then the matrix form of the equation (3.34) is

$$\underline{\widehat{y}} = \begin{pmatrix} 1 & \sin(\frac{2\pi 1}{N}\ k) & \cos(\frac{2\pi 1}{N}\ k) & \sin(\frac{2\pi 1}{N}\ l) & \cos(\frac{2\pi 1}{N}\ l) \\ 1 & \sin(\frac{2\pi 2}{N}\ k) & \cos(\frac{2\pi 2}{N}\ k) & \sin(\frac{2\pi 2}{N}\ l) & \cos(\frac{2\pi 2}{N}\ l) \\ & & \vdots & & \\ 1 & \sin(\frac{2\pi N}{N}\ k) & \cos(\frac{2\pi N}{N}\ k) & \sin(\frac{2\pi N}{N}\ l) & \cos(\frac{2\pi N}{N}\ l) \end{pmatrix} \begin{pmatrix} \widehat{\beta}_0 \\ \widehat{\beta}_{k1} \\ \widehat{\beta}_{k2} \\ \widehat{\beta}_{l1} \\ \widehat{\beta}_{l2} \end{pmatrix}. \tag{3.35}$$

As long as the frequencies in the model are between 1 and $(N-1)/2$, the columns in the $\mathbf{X}$ matrix are orthogonal. In addition, each column (except the $\underline{1}$) has a squared norm of $N/2$. We divide the sine and cosines by $\sqrt{N/2}$, so that the $\mathbf{X}$ matrix we use will be

$$\mathbf{X} = \begin{pmatrix} \underline{1}_N & \mathbf{X}^* \end{pmatrix}, \quad \mathbf{X}^* = \frac{1}{\sqrt{N/2}} \begin{pmatrix} \sin(\frac{2\pi 1}{N}\, k) & \cos(\frac{2\pi 1}{N}\, k) & \sin(\frac{2\pi 1}{N}\, l) & \cos(\frac{2\pi 1}{N}\, l) \\ \sin(\frac{2\pi 2}{N}\, k) & \cos(\frac{2\pi 2}{N}\, k) & \sin(\frac{2\pi 2}{N}\, l) & \cos(\frac{2\pi 2}{N}\, l) \\ & & \vdots & \\ \sin(\frac{2\pi N}{N}\, k) & \cos(\frac{2\pi N}{N}\, k) & \sin(\frac{2\pi N}{N}\, l) & \cos(\frac{2\pi N}{N}\, l) \end{pmatrix}. \tag{3.36}$$

In general, there will be one sine and one cosine vector for each frequency in $\mathcal{K}$. Let $K = \#\mathcal{K}$, so that $\mathbf{X}$ has $2K+1$ columns.

Choosing the set of frequencies to use in the fit is the same as the subset selection from Section 2.5, with the caveat that the columns come in pairs. Because of the orthonormality of the vectors, the estimates of the parameters are the same no matter which frequencies are in the fit. For the full model, with all $\lfloor (N-1)/2 \rfloor$ frequencies[3], the estimates of the coefficients are

$$\widehat{\beta}_0 = \overline{y}, \ \underline{\widehat{\beta}}^* = \mathbf{X}^{*'}\underline{y}, \tag{3.37}$$

since $\mathbf{X}^{*'}\mathbf{X}^* = \mathbf{I}$. These $\widehat{\beta}^*_{kj}$'s are the coefficients in the Fourier Transform of the y's. Figure 3.10 has the fits for several sets of frequencies. Those with 3 and 10 frequencies fit look the most reasonable.

If N is odd, the residual sum of squares for the fit using the frequencies in $\mathcal{K}$ is the sum of squares of the coefficients for the frequencies that are left out:

$$N \text{ odd:} \qquad RSS_{\mathcal{K}} = \sum_{k \notin \mathcal{K}} SS_k, \quad \text{where} \quad SS_k = (\widehat{\beta}^2_{k1} + \widehat{\beta}^2_{k2}). \tag{3.38}$$

If N is even, then there is one degree of freedom for the residuals of the full model. The space for the residuals is spanned by the vector $(+1, -1, +1, -1, \ldots, +1, -1)'$, hence the residual sums of squares is

$$RSS_{Full} = \frac{(y_1 - y_2 + y_3 - y_4 \pm \cdots + y_{N-1} - y_N)^2}{N}. \tag{3.39}$$

Then

$$N \text{ even:} \qquad RSS_{\mathcal{K}} = \sum_{k \notin \mathcal{K}} SS_k + RSS_{Full}. \tag{3.40}$$

We can now proceed as for subset selection, where for each subset of frequencies $\mathcal{K}$ we estimate the prediction error using either the direct estimate or some cross-validation scheme. We need not search over all possible subsets of the frequencies, because by orthogonality we automatically know that the best fit with K frequencies will use the K frequencies with the best SS_k's. For the motorcycle data, $N = 133$, so there are 66 frequencies to

[3] $\lfloor z \rfloor$ is the largest integer less than or equal to z.

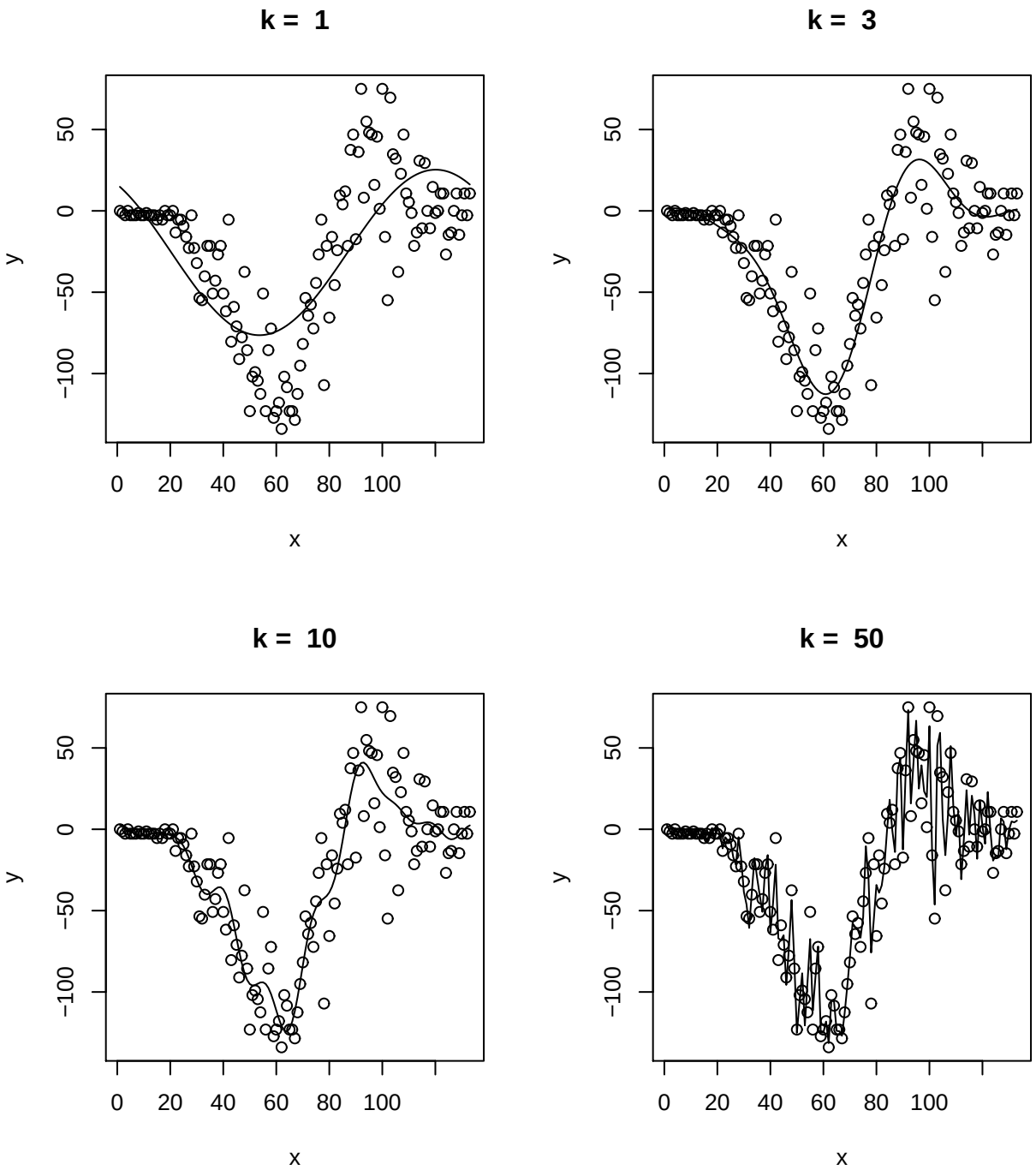


Figure 3.10: Some fits

consider. Ranking the frequencies based on their corresponding sums of squares, from largest to smallest, produces the following table:

Rank	Frequency k	SS_k
1	1	172069.40
2	2	67337.03
3	3	7481.31
4	43	4159.51
5	57	2955.86
	$\vdots$	
65	52	8.77
66	31	4.39

(3.41)

3.2.1 Estimating σ_e^2

In order to estimate the ERR_{in}, we need the estimate of σ_e^2. The covariance matrix of the $\widehat{\beta}_{kj}$'s is $\sigma_e^2\mathbf{I}$. If we further assume the e_i's are normal, then these coefficient estimates are also normal and independent, and all with variance σ_e^2. Thus if a $\beta_{jk} = 0$, $\widehat{\beta}_{kj}^2$ is $\sigma_e^2\chi_1^2$, and

$$(\beta_{k1}, \beta_{k2}) = (0, 0) \Rightarrow SS_k = \widehat{\beta}_{k1}^2 + \widehat{\beta}_{k2}^2 \sim \sigma_e^2\chi_2^2. \tag{3.42}$$

To estimate σ_e^2, we want to use the SS_k's for the frequencies whose coefficients are 0. It is natural to use the smallest ones, but which ones? QQ-plots are helpful in this task.

A QQ-plot plots the quantiles of one distribution versus the quantiles of another. If the plot is close to the line $y = x$, then the two distributions are deemed to be similar. In our case we have a set of SS_k's, and wish to compare them to the χ_2^2 distribution. Let $SS_{(k)}$ be the k^{th} smallest of the SS_k's. (Which is the opposite order of what is in table (3.41).) Suppose we wish to take the smallest m of these, where m will be presumably close to 66. Then among the sample

$$SS_{(1)}, \ldots, SS_{(m)}, \tag{3.43}$$

the $(i/m)^{th}$ quantile is just $SS_{(i)}$. We match this quantile with the $(i/m)^{th}$ quantile of the χ_2^2 distribution, although to prevent (i/m) from being 1, and the quantile 0, we instead look at $(i - 3/8)/(m + 1/4)$ (if $m \leq 10$) or $(i - 1/2)/m$ instead. For a given distribution function F, this quantile η_i satisfies

$$F(\eta_i) = \frac{i - \frac{1}{2}}{m}. \tag{3.44}$$

The $F(z) = 1 - e^{-x/2}$ in the χ_2^2 case.

Figure 3.11 shows the QQ-Plots for $m = 66, 65, 64$, and 63, where the η_i's are on the horizontal axis, and the $SS_{(i)}$'s are on the vertical axis.

We can see that the first two plots are not linear; clearly the largest two SS_k's should not be used for estimating σ_e^2. The other two plots look reasonable, but we will take the fourth as the best. That is, we can consider the SS_k's, leaving out the three largest, as a

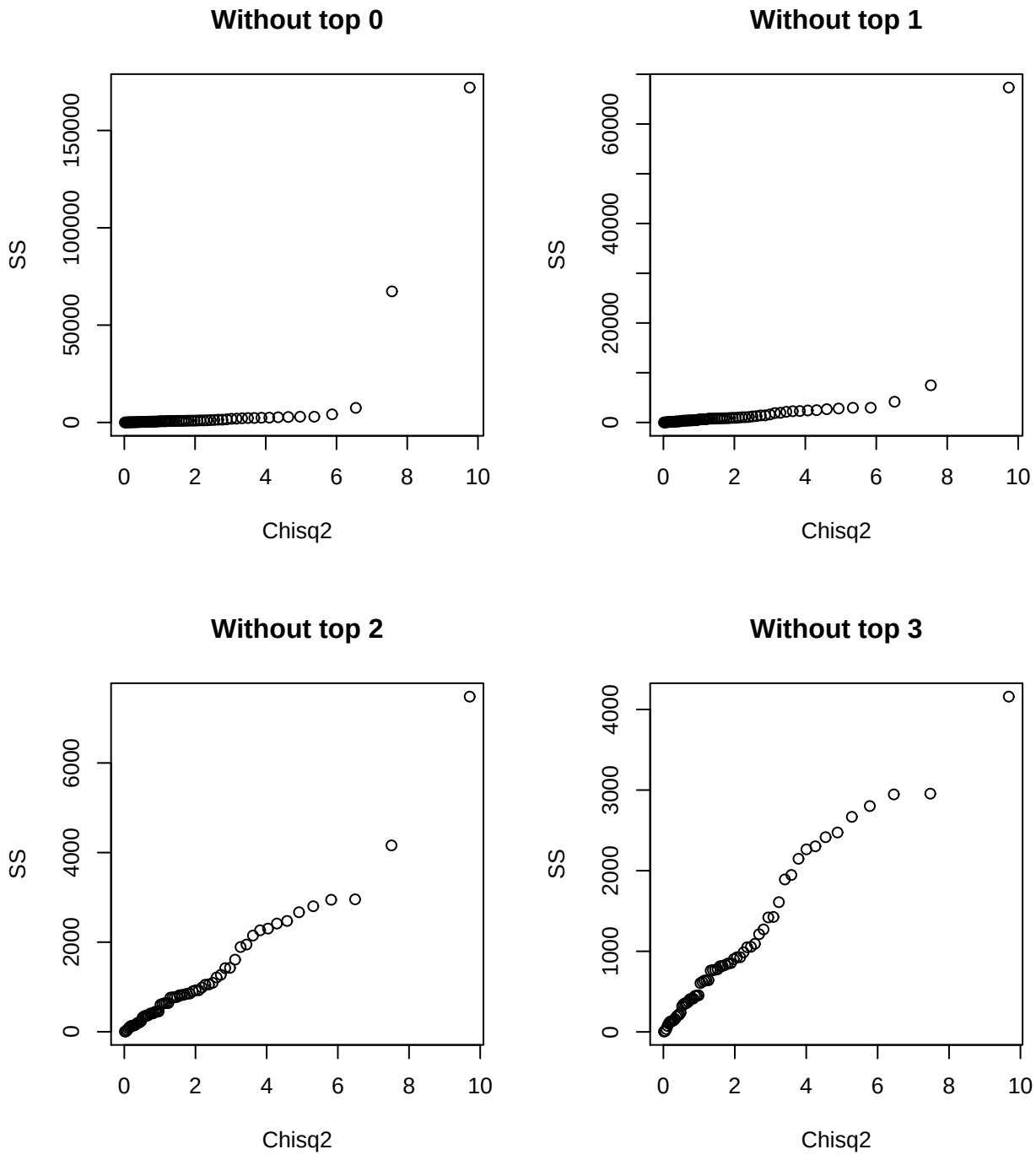


Figure 3.11: QQ-Plots

sample from a $\sigma_e^2\chi_2^2$. The slope of the line should be approximately σ_e^2. We take the slope (fitting the least-squares line) to be our estimate of σ_e^2, which in this case turns out to be $\widehat{\sigma}_e^2 = 473.8$.

Letting K be the number of frequencies used for a given fit, the prediction error can be estimated by

$$\widehat{ERR}_{in,K} = \overline{err}_K + 2\ \widehat{\sigma}_e^2\ \frac{2K+1}{N}, \tag{3.45}$$

where $\overline{err}_K = RSS_K/N$ is found in (3.38) with $\mathcal{K}$ containing the frequencies with the K largest sums of squares. The next table has these estimates for the first twenty fits:

K	edf	$\widehat{ERR}_{in,K}$	
0	1	2324.59	
1	3	1045.08	
2	5	553.04	
3	7	511.04	
4	9	494.01	
5	11	486.04	
6	13	478.14	
7	15	471.32	
8	17	465.51	
9	19	461.16	
10	21	457.25	
11	23	454.19	
12	25	451.41	
13	27	449.51	
14	29	449.12	***
15	31	449.15	
16	33	451.30	
17	35	454.82	
18	37	458.40	
19	39	463.09	

(3.46)

The fit using 14 frequencies has the lowest estimated error, although 12 and 13 are not much different. See Figure 3.34. It is very wiggly.

3.2.2 Cross-validation

Next we try cross-validation. The leave-one-out cross-validation estimate of error for y_i for a given fit is, from (3.15),

$$y_i - y_i^{[-i]} = \frac{y_i - \widehat{y}_i}{1 - h_{ii}}. \tag{3.47}$$

The $\mathbf{H}$ matrix is

$$\mathbf{H} = (\underline{1}_N\mathbf{X}^*)\begin{pmatrix} N & \underline{0}' \\ \underline{0} & \mathbf{I} \end{pmatrix}(\underline{1}_N X^*)' = \frac{1}{N}\ \underline{1}_N\underline{1}_N' + \mathbf{X}^*\mathbf{X}^{*'}, \tag{3.48}$$

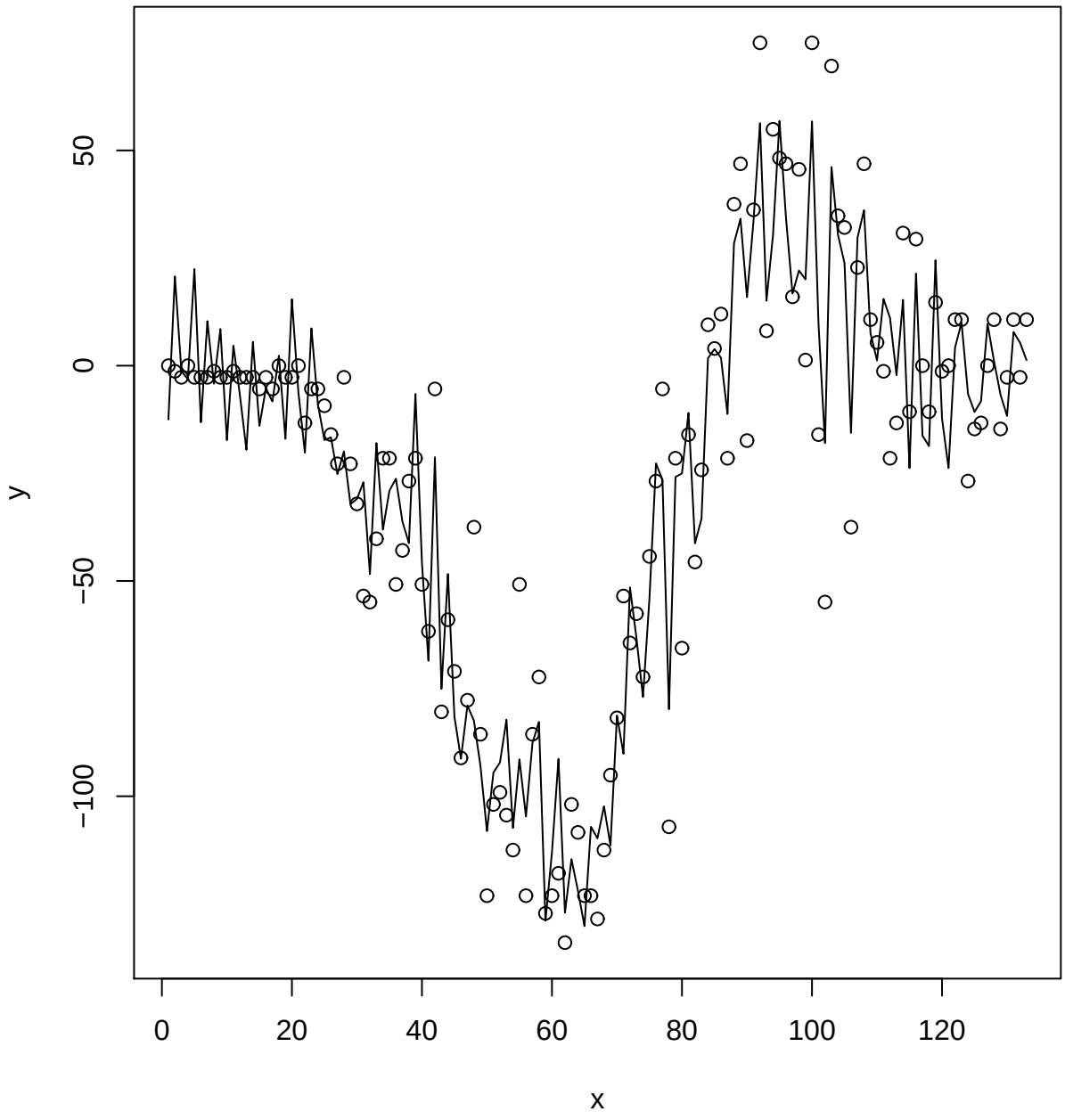


Figure 3.12: The fit with $K = 14$ frequencies

where $\mathbf{X}^*$ has the columns for whatever frequencies are being entertained. The diagonals are thus

$$h_{ii} = \frac{1}{N} + \|\underline{x}_i^*\|^2, \tag{3.49}$$

where $\underline{x}_i^*$ has the sines and cosines for observation i, divided by $\sqrt{N/2}$ (as in (3.36)). Then

$$\|\underline{x}_i^*\|^2 = \frac{1}{N/2} \sum_{k \in \mathcal{K}} [\sin\left(\frac{2\pi i}{N}\ k\right)^2 + \cos\left(\frac{2\pi i}{N}\ k\right)^2] = \frac{2}{N}\ K, \tag{3.50}$$

because $\sin^2 + \cos^2 = 1$. Hence the h_{ii}'s are the same for each i,

$$h_{ii} = \frac{2K+1}{N}. \tag{3.51}$$

That makes it easy to find the leave-one-out estimate of the prediction error for the model with K frequencies:

$$\widehat{ERR}_{in,K,cv} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \widehat{y}_i}{1-h_{ii}}\right)^2 = \frac{1}{N}\ \frac{RSS_K}{(1-(2K+1)/N)^2} = \frac{N}{(N-2K-1)^2}\ RSS_K. \tag{3.52}$$

Figure 3.13 has the plot of K versus this prediction error estimate.

For some reason, the best fit using this criterion has $K = 64$, which is ridiculous. But you can see that the error estimates level off somewhere between 10 and 20, and even 3 is much better than 0, 1, or 2. It is only beyond 60 that there is a distinct fall-off. I believe the reason for this phenomenon is that even though it seems like we search over 66 fits, we are implicitly searching over all $2^{66} \approx 10^{20}$ fits, and leaving just one out at a time does not fairly address so many fits. (?)

I tried again using a leave-thirteen-out cross-validation, thirteen being about ten percent of N. This I did directly, randomly choosing thirteen observations to leave out, finding the estimated coefficients using the remaining 120 observations, then finding the prediction errors of the thirteen. I repeated the process 1000 times for each K, resulting in Figure 3.14. Now $K = 4$ is best, with $K = 3$ very close.

The table (3.53) has the first few frequencies, plus the standard error of the estimate. It is the standard error derived from repeating the leave-thirteen-out 1000 times. What this error estimate is estimating is the estimate we would have after trying all possible $\binom{133}{13}$ subsamples. Thus the standard error estimates how far from this actual estimate we are. In any case, you can see that the standard error is large enough that one has no reason to suspect that $K = 4$ is better than $K = 3$, so we will stick with that one, resulting in Figure 3.15.

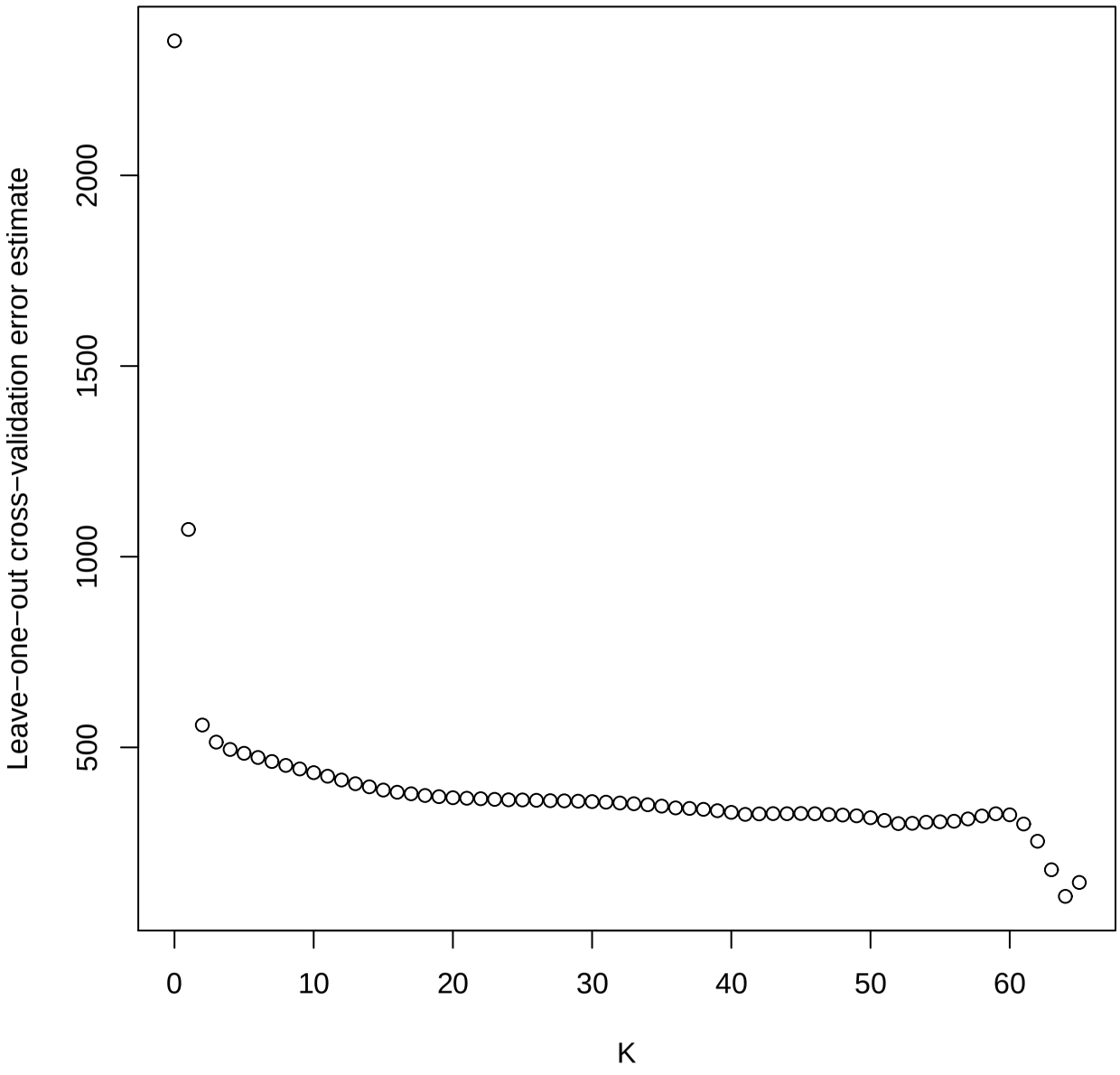


Figure 3.13: Leave-one-out error estimates

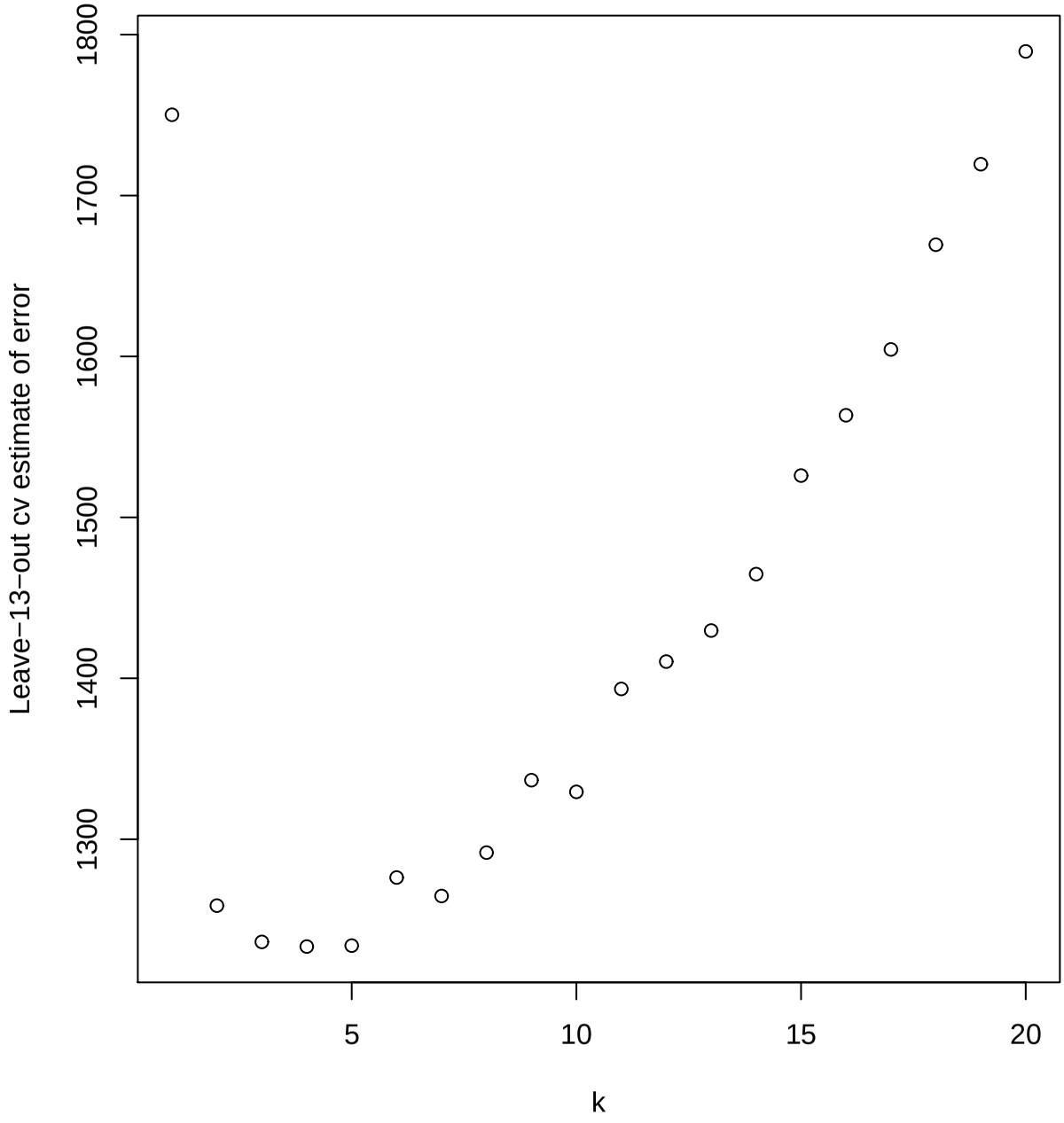


Figure 3.14: Leave-thirteen-out error estimates

K	$\widehat{ERR}_{in,K,cv}$	SE
1	1750.17	17.02
2	1258.75	14.59
3	1236.19	14.83
4	1233.30	13.38
5	1233.89	13.15
6	1276.21	13.35

(3.53)

To summarize, various methods chose 3, around 14, and around 64 as the yielding the best prediction. Visually, $K = 3$ seems fine.

3.2.3 Using R

The Motorcycle Acceleration Data is in the `MASS` package, which is used in the book *Modern Applied Statistics with S*, by Venables and Ripley. It is a very good book for learning S and R, and modern applied statistics. You need to load the package. The data set is in `mcycle`.

First, create the big $\mathbf{X}$ matrix as in (3.36), using all the frequencies:

```
N <- 133
x <- 1:N
y <- mcycle[,2]
theta <- x*2*pi/N
xx <- NULL
for(k in 1:66) xx<-cbind(xx,cos(k*theta),sin(k*theta))
xx <- xx/sqrt(N/2)
xx<-cbind(1,xx)
```

The $\widehat{\beta}_{kj}$'s (not including $\widehat{\beta}_0$, which is $\overline{y}$) are in

```
bhats <- t(xx[,-1])%*%y
```

To find the corresponding sums of squares, SS_k of (3.38), we need to sum the squares of consecutive pairs. The following first puts the coefficients into a $K \times 2$ matrix, then gets the sum of squares of each row:

```
b2 <- matrix(bhats,ncol=2,byrow=T)
b2 <- apply(b2^2,1,sum)
```

For the QQ-plots, we plot the ordered SS_k's versus the quantiles of a χ^2_2. To get the m points $(i - 1/2)/m$ as in (3.44), use `ppoints(m)`.

```
ss <- sort(b2) # The ordered SS_k's
m <- 66
plot(qchisq(ppoints(m),2),ss[1:m])
m <- 63
plot(qchisq(ppoints(m),2),ss[1:m])
```

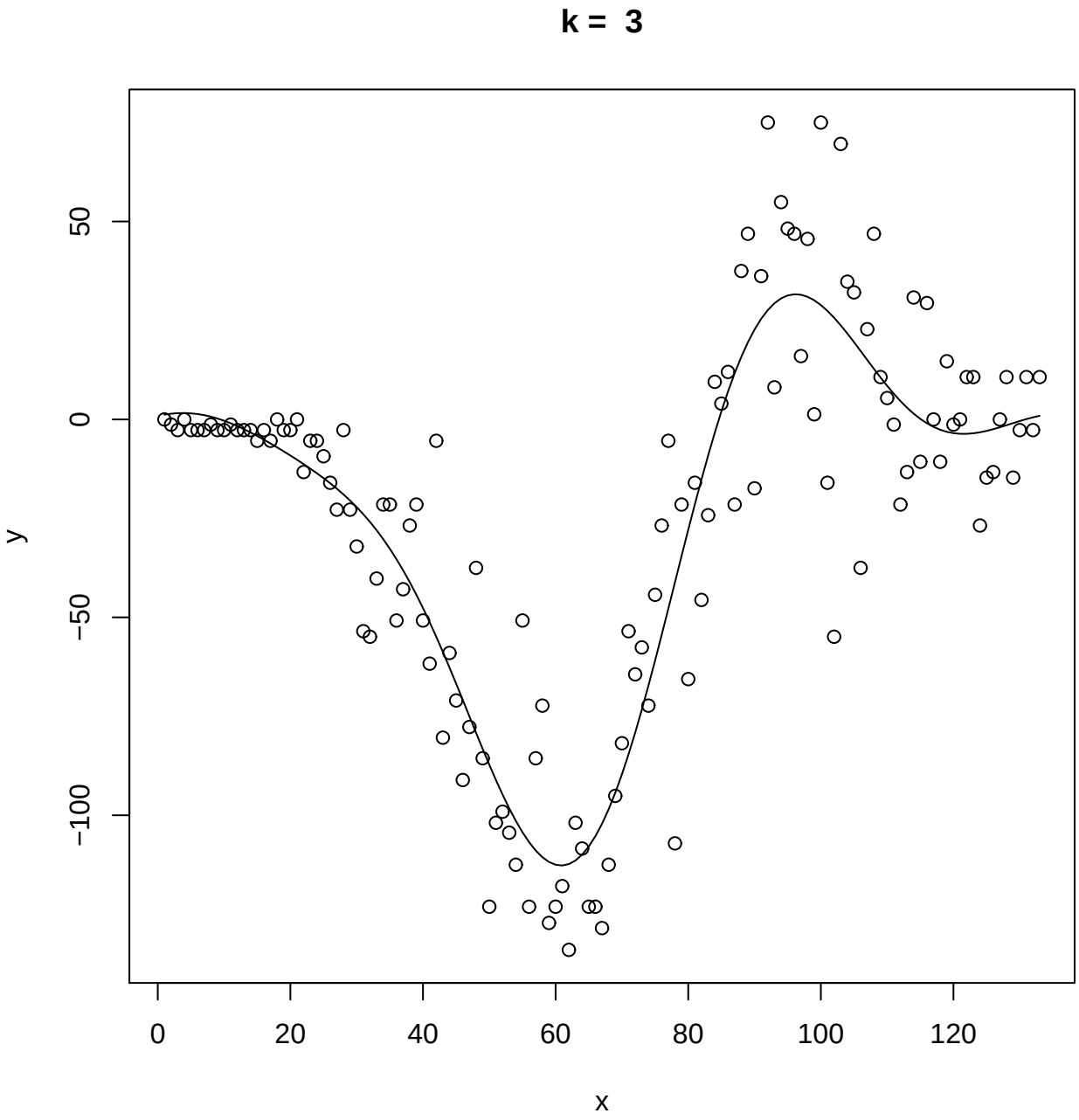


Figure 3.15: The fit for $K = 3$

To find $\widehat{ERR}_{in,K}$ in (3.45), we try the $K = 0, ..., 65$. The `rss` sums up the smallest m sums of squares for each m, then reverses order so that the components are $RSS_0, RSS_1, \ldots, RSS_65$. If you wish to include $K = 66$, it fits exactly, so that the prediction error is just $2\widehat{\sigma}_e^2$ since $edf = N$.

```
edf <- 1+2*(0:65)  # The vector of (2K+1)'s
rss <- cumsum(ss)[66:1]
errhat <- rss/N+2*473.8*edf/N
plot(0:65,errhat)
```

For leave-one-out cross-validation, we just multiply the residual sums of squares by the appropriate factor from (3.52):

```
errcv<-N*rss/(N-dd)^2
plot(0:65,errcv)
```

Finally, for leave-13-out cv,

```
ii <- order(b2)[66:1] # Frequencies in order
cv <- NULL
cvv <- NULL
for(k in 1:40) {
   jj <- c(1,2*ii[1:k],2*ii[1:k]+1)
   # Indices of beta's for chosen frequencies
   s <- NULL
   for(i in 1:1000) {
      b <- sample(133,13) # Picks 13 from 1 to 133
      xb <- xx[-b,jj]
      rb<-y[b] - xx[b,jj]%*%solve(t(xb)%*%xb,t(xb))%*%y[-b]
      s <- c(s,mean(rb^2))
   }
   cv <- c(cv,mean(s))
   cvv <- c(cvv,var(s))
}
```

The `cv` has the cross-validation estimates, and `cvv` has the variances, so that the standard errors are `sqrt(cvv/1000)`. Because of the randomness, each time you run this routine you obtain a different answer. Other times I have done it the best was much higher than 3, like $K = 20$.

3.3 Local fitting: Regression splines

The fits uses so far have been "global" in the sense that the basis functions $h_k(x_i)$ cover the entire range of the x's. Thus, for example, in the birthrates example, the rates in the 1920's

affect the fits in the 1990's, and *vice versa*. Such behavior is fine if the trends stays basically the same throughout, but as one can see in the plots (Figure 3.1), different regions of the x values could use different fits. Thus "local" fits have been developed, wherein the fits for any x depends primarily on the nearby observations.

The simplest such fit is the **regressogram** (named by John Tukey), which divides the x-axis into a number of regions, then draws a horizontal line above each region at the average of the corresponding y's. Figure 3.16 shows the regressogram for the birthrate data, where there are (usually) five observations in each region. The plot is very jagged, but does follow the data well, and is extremely simple to implement. One can use methods from this chapter to decide on how many regions, and which ones, to use.

The regressogram fits the simplest polynomial to each region, that is, the constant. Natural extensions would be to fit higher-degree polynomials to each region (or sines and cosines). Figure 3.17 fits a linear regression to each of four regions (A *lineogram*[4].). The lines follow the data fairly well, except at the right-hand area of the third region.

One drawback to the lineogram, or higher-order analogs, is that the fits in the separate regions do not meet at the boundaries. One solution is to use a moving window, so for any x, the x_i's within a certain distance of x are used for the fit. That route leads to kernel fits, which are very nice. We will look more carefully at **splines**, which fit polynomials to the regions but require them to be connected smoothly at the boundaries. It is as if one ties knots to connect the ends of the splines, so the x-values demarking the boundaries are called **knots**. Figure 3.18 shows the **linear spline**, where the knots are at 1937.5, 1959.5, and 1981.5. The plot leaves out the actual connections, but one can imagine that the appropriate lines will intersect at the boundaries of the regions.

The plot has sharp points. By fitting higher-order polynomials, one can require more smoothness at the knots. The typical requirement for degree m polynomials is having $m-1$ continuous derivatives:

Name of spline	Degree	Smoothness
Constant	0	None
Linear	1	Continuous
Quadratic	2	Continuous first derivatives
Cubic	3	Continuous second derivatives
	$\vdots$	
m-ic	m	Continuous $(m-1)^{th}$ derivatives

(3.54)

Figure 3.19 shows the cubic spline fit for the data. It is very smooth, so that one would not be able to guess by eye where the knots are. Practitioners generally like cubic splines. They provide a balance between smoothness and simplicity.

For these data, the fit here is not great, as it is too low around 1960, and varies too much after 1980. The fit can be improved by increasing either the degree or the number of knots, or both. Because we like the cubics, we will be satisfied with cubic splines, but

[4]Not a real statistical term. It is a real word, being some kind of motion X-ray.

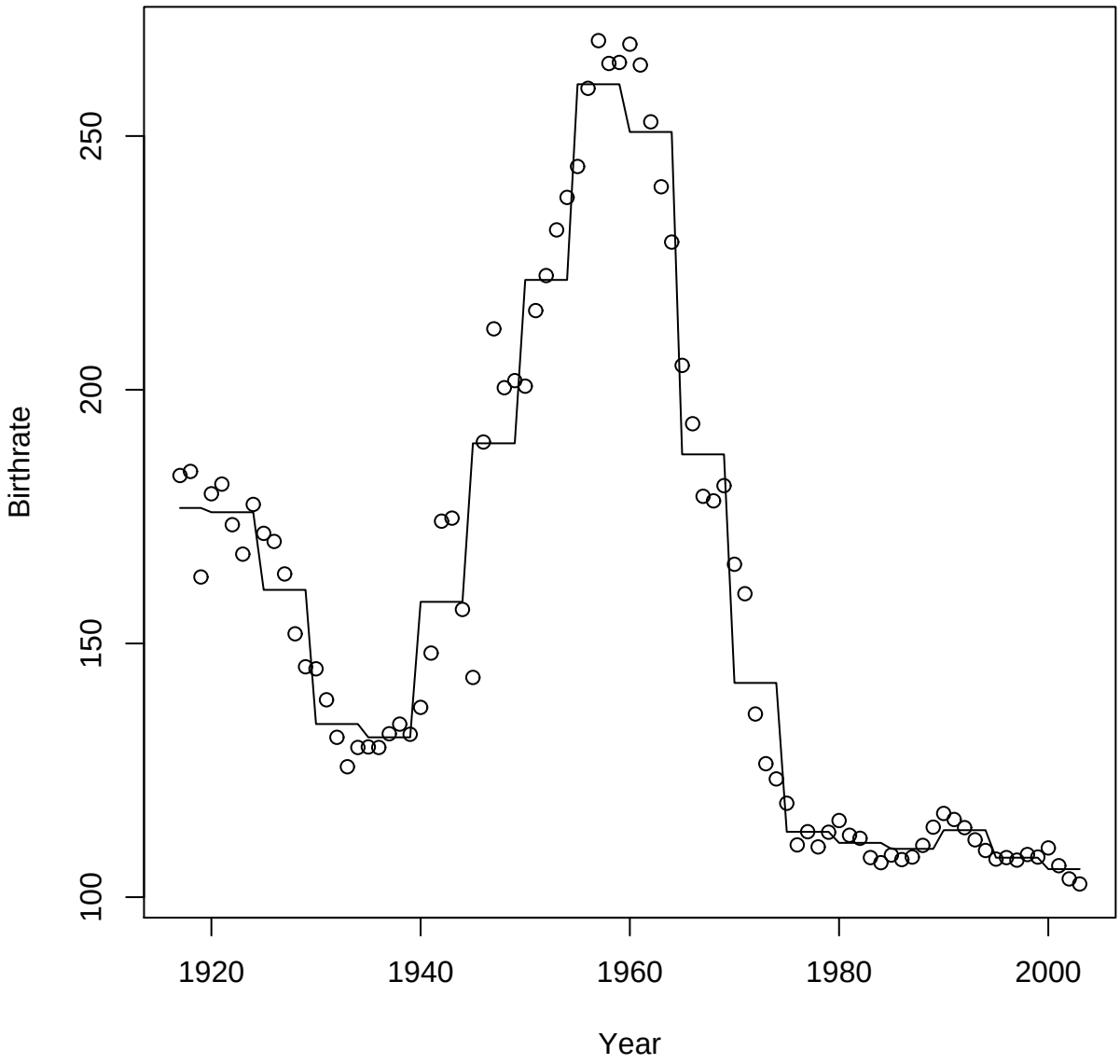


Figure 3.16: Regressogram for birthrate data

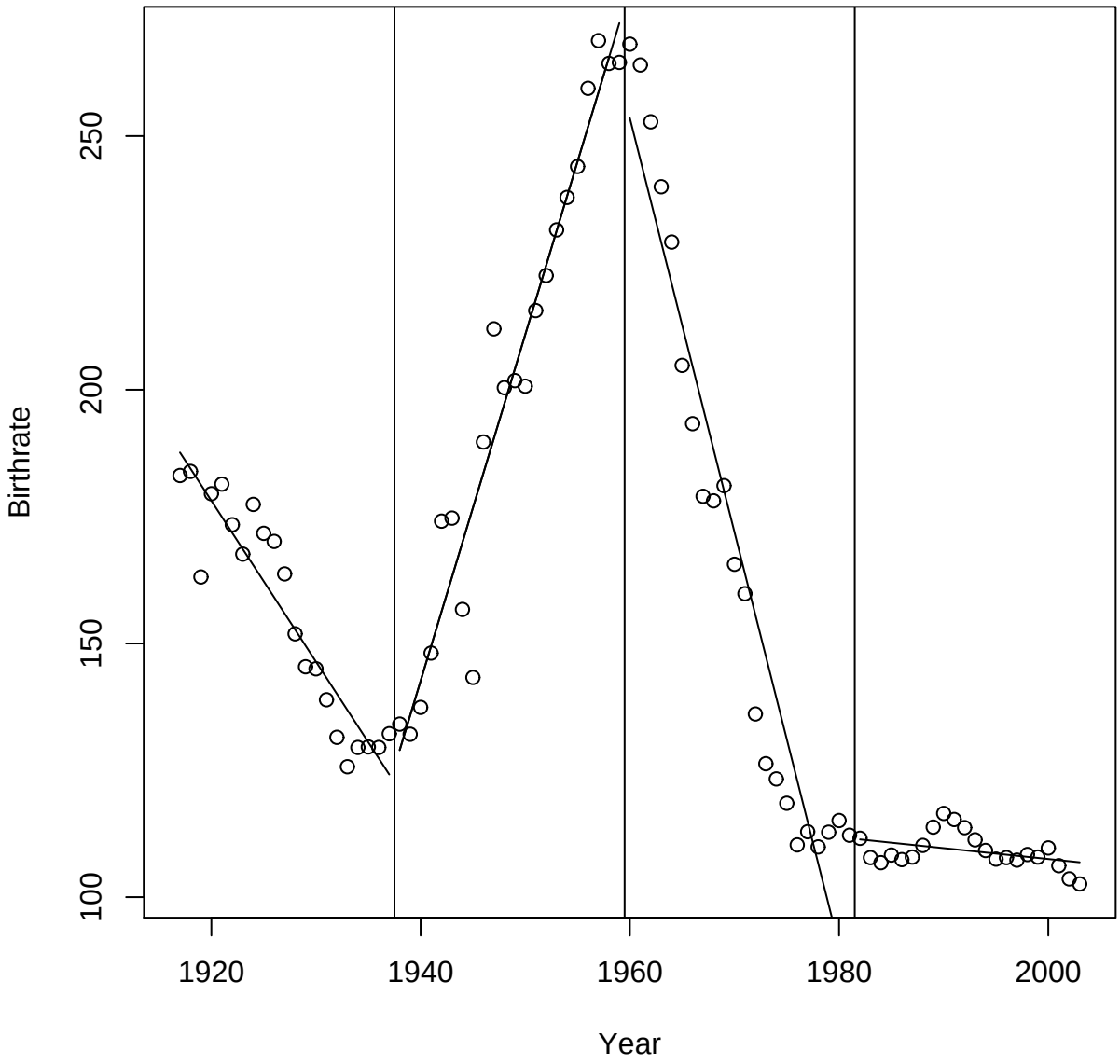


Figure 3.17: Lineogram for birthrate data

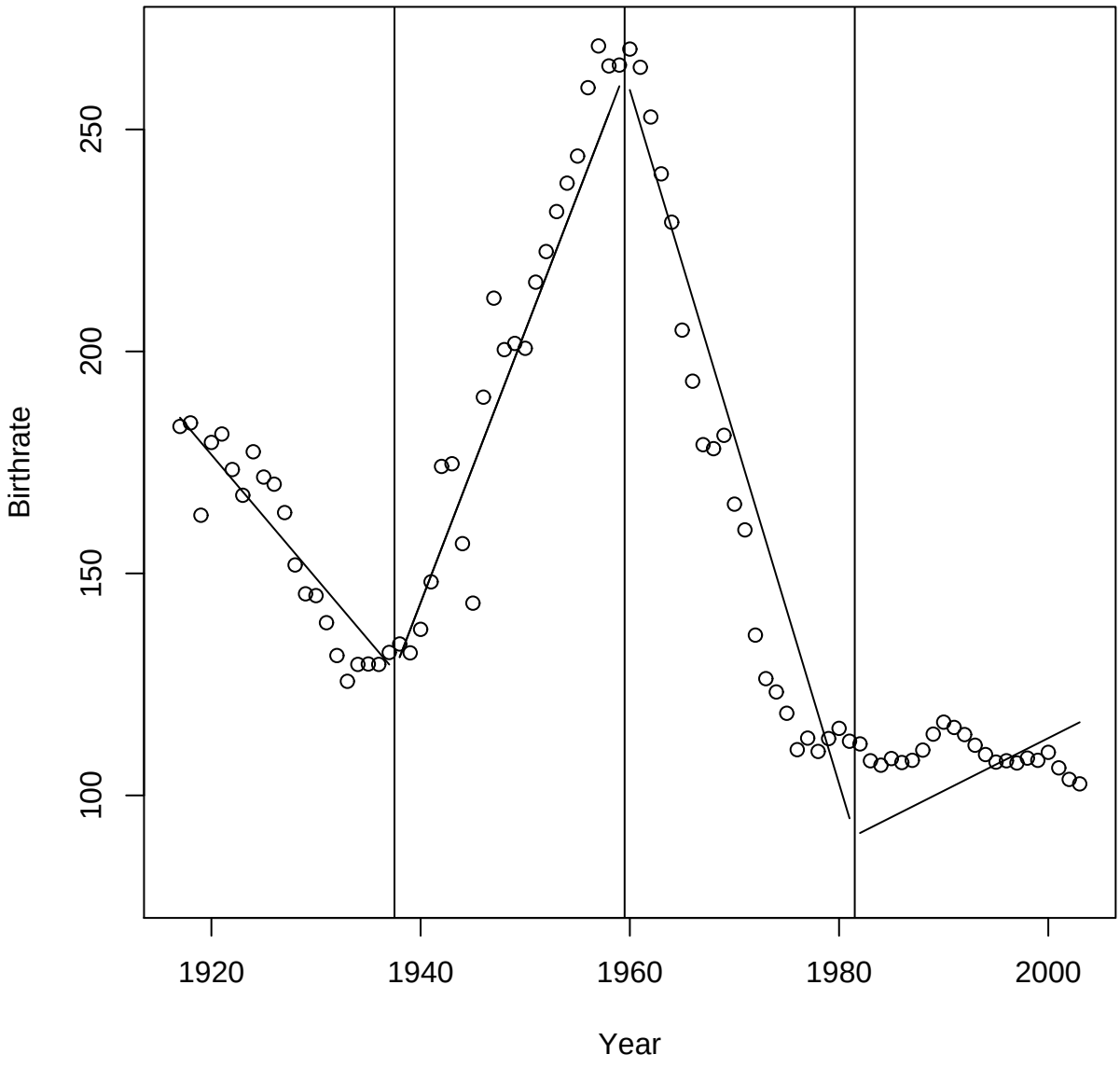


Figure 3.18: Linear spline for birthrate data

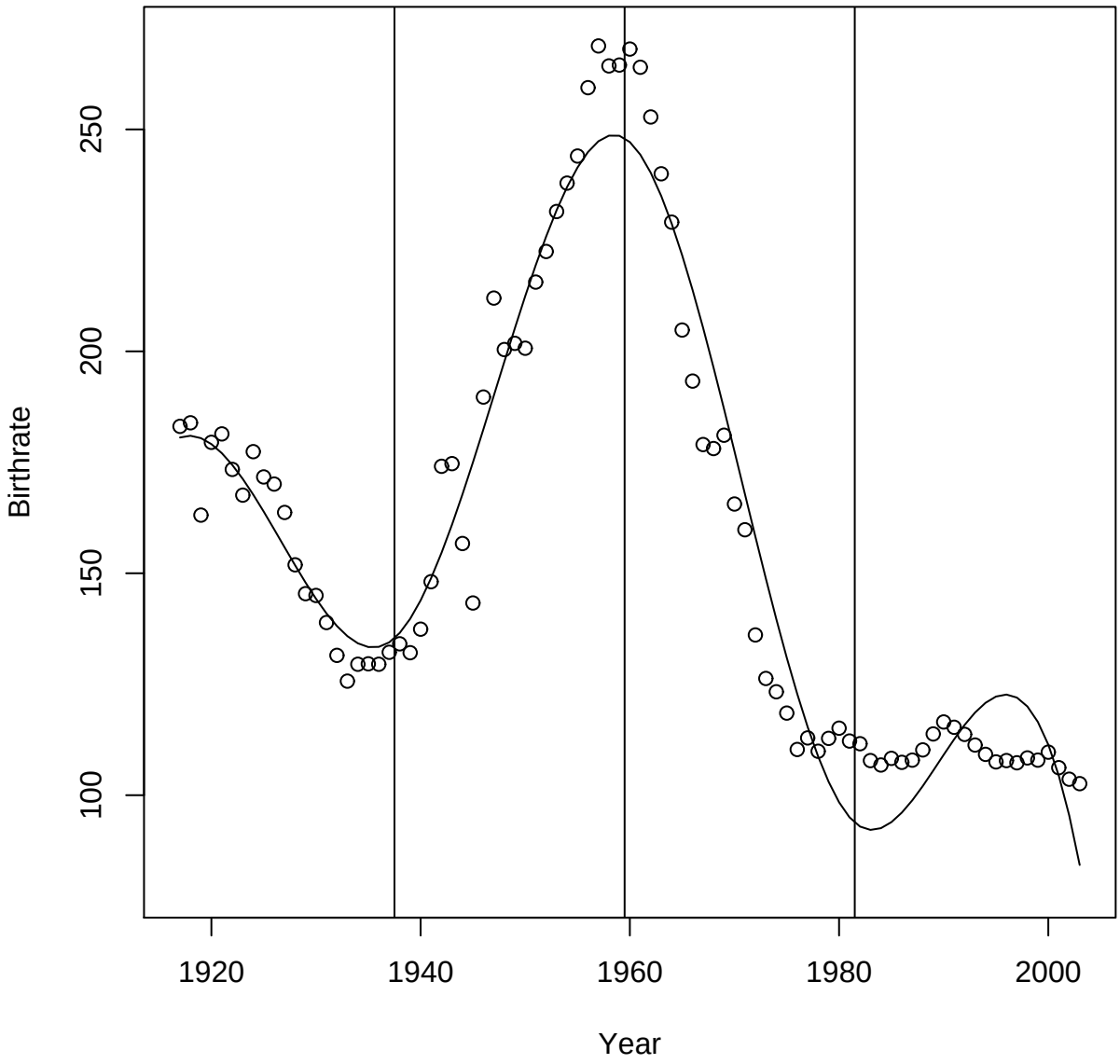


Figure 3.19: Cubic spline for birthrate data

consider increasing the number of knots. The question then is to decide on how many knots. For simplicity, we will use equally-spaced knots, although there is no reason to avoid other spacings. E.g., for the birthrates, knots at least at 1938, 1960, and 1976 would be reasonable.

The effective degrees of freedom is the number of free parameters to estimate. With K knots, there are $K+1$ regions. Let $k_1 < k_2 < \cdots < k_K$ be the values of the knots. Consider the cubic splines, so that the polynomial for the l^{th} region is

$$a_l + b_l x + c_l x^2 + d_l x^3. \tag{3.55}$$

At knot l, the $(l-1)^{st}$ and l^{th} regions' cubic polynomials have to match the value (so that the curve is continuous), the first derivative, and the second derivative:

$$\begin{aligned} a_{l-1} + b_{l-1}k_l + c_{l-1}k_l^2 + d_{l-1}k_l^3 &= a_l + b_l k_l + c_l k_l^2 + d_l k_l^3, \\ b_{l-1} + 2c_{l-1}k_l + 3d_{l-1}k_l^2 &= b_l + 2c_l k_l + 3d_l k_l^2, \quad \text{and} \\ 2c_{l-1} + 6d_{l-1}k_l &= 2c_l + 6d_l k_l. \end{aligned} \tag{3.56}$$

Thus each knot contributes three linear constraints, meaning the effective degrees of freedom are $edf(K \text{ knots, degree} = 3) = 4(K+1) - 3K = K + 4$. For general degree m of the polynomials,

$$edf(K \text{ knots, degree} = m) = (m+1)(K+1) - mK = K + m + 1. \tag{3.57}$$

An intuitive basis for the cubic splines is given by the following, where there are knots at $k_1, k_2, \ldots, k_K$:

$$\begin{aligned} x < k_1 \quad &: \quad \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \\ k_1 < x < k_2 \quad &: \quad \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-k_1)^3 \\ k_2 < x < k_3 \quad &: \quad \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-k_1)^3 + \beta_5 (x-k_2)^3 \\ &\vdots \\ k_K < x \quad &: \quad \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-k_1)^3 + \beta_5 (x-k_2)^3 + \cdots + \beta_{K+3}(x-k_K)^3 \end{aligned} \tag{3.58}$$

First, within each region, we have a cubic polynomial. Next, it is easy to see that at k_1, the first two equations are equal; at k_2 the second and third are equal, etc. Thus the entire curve is continuous. The difference between the first derivative l^{th} and $(l+1)^{st}$ regions' curves is

$$\frac{\partial}{\partial x}(x-k_l)^3 = 3(x-k_l)^2, \tag{3.59}$$

which is 0 at the boundary of those regions, k_l. Thus the first derivative of the curve is continuous. Similarly for the second derivative. The span of the functions

$$1, x, x^2, x^3, (x-k_1)^3_+, (x-k_2)^3_+, \ldots, (x-k_K)^3_+, \tag{3.60}$$

where $z^3_+ = 0$ if $z < 0$ and z^3 if $z \geq 0$, is indeed within the set of cubic splines, and the number of such functions is $K+4$, the dimension of the space. As long as there are enough

distinct x's for these functions to be linearly independent, then, they must constitute a basis for the cubic splines. Translating to the $\mathbf{X}$ matrix for given data, for $K = 2$ we have

$$\begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & 0 & 0 \\ 1 & x_2 & x_2^2 & x_2^3 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_a & x_a^2 & x_a^3 & 0 & 0 \\ 1 & x_{a+1} & x_{a+1}^2 & x_{a+1}^3 & (x_{a+1}-k_1)^3 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{a+b} & x_{a+b}^2 & x_{a+b}^3 & (x_{a+b}-k_1)^3 & 0 \\ 1 & x_{a+b+1} & x_{a+b+1}^2 & x_{a+b+1}^3 & (x_{a+b+1}-k_1)^3 & (x_{a+b+1}-k_2)^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & x_N^3 & (x_N-k_1)^3 & (x_N-k_2)^3 \end{pmatrix}, \tag{3.61}$$

where there are a observations in the first region and b in the second. In practice, the so-called **B-spline basis** is used, which is an equivalent basis to the one in (3.61), but has some computational advantages. Whichever basis one uses, the fit for a given sets of knots is the same.

Using the usual least-squares fit, we have the estimated prediction error for the cubic spline using K knots to be

$$\widehat{ERR}_{in,K} = \overline{err}_K + 2\widehat{\sigma}_e^2 \, \frac{K+4}{N}. \tag{3.62}$$

Or, it is easy to use find the leave-one-out cross-validation estimate. The results of the two methods are in Figure 3.20. For the C_p-type estimate, $K = 36$ knots minimizes the prediction error estimate at 66.27, but we can see there are many other smaller K's with similar error estimates. The $K = 9$ has estimate 66.66, so it is reasonable to take $K = 9$. Using cross-validation the best is $K = 9$, although many values up to 20 are similar. Figure 3.21 has the two fits $K = 9$ and $K = 36$. Visually, the smaller K looks best, though the $K = 36$ fit works better after 1980.

As we saw in (3.12), high-degree polynomials are not good for extrapolation. The cubic splines should be better than higher degree polynomials like 16 or 26, but can still have problems. **Natural splines** are cubic splines that try to alleviate some of the concerns with extrapolation by also requiring that outside the two extreme knots, the curve is linear. Thus we have four more constraints, two at each end (the quadratic and cubic coefficients being 0). The effective degrees of freedom for a natural spline for with K knots is simply K.

The next table looks at some predictions beyond 2003 (the last date in the data set) using the same effective degrees of freedom of 13 for the polynomial, cubic spline, and natural spline

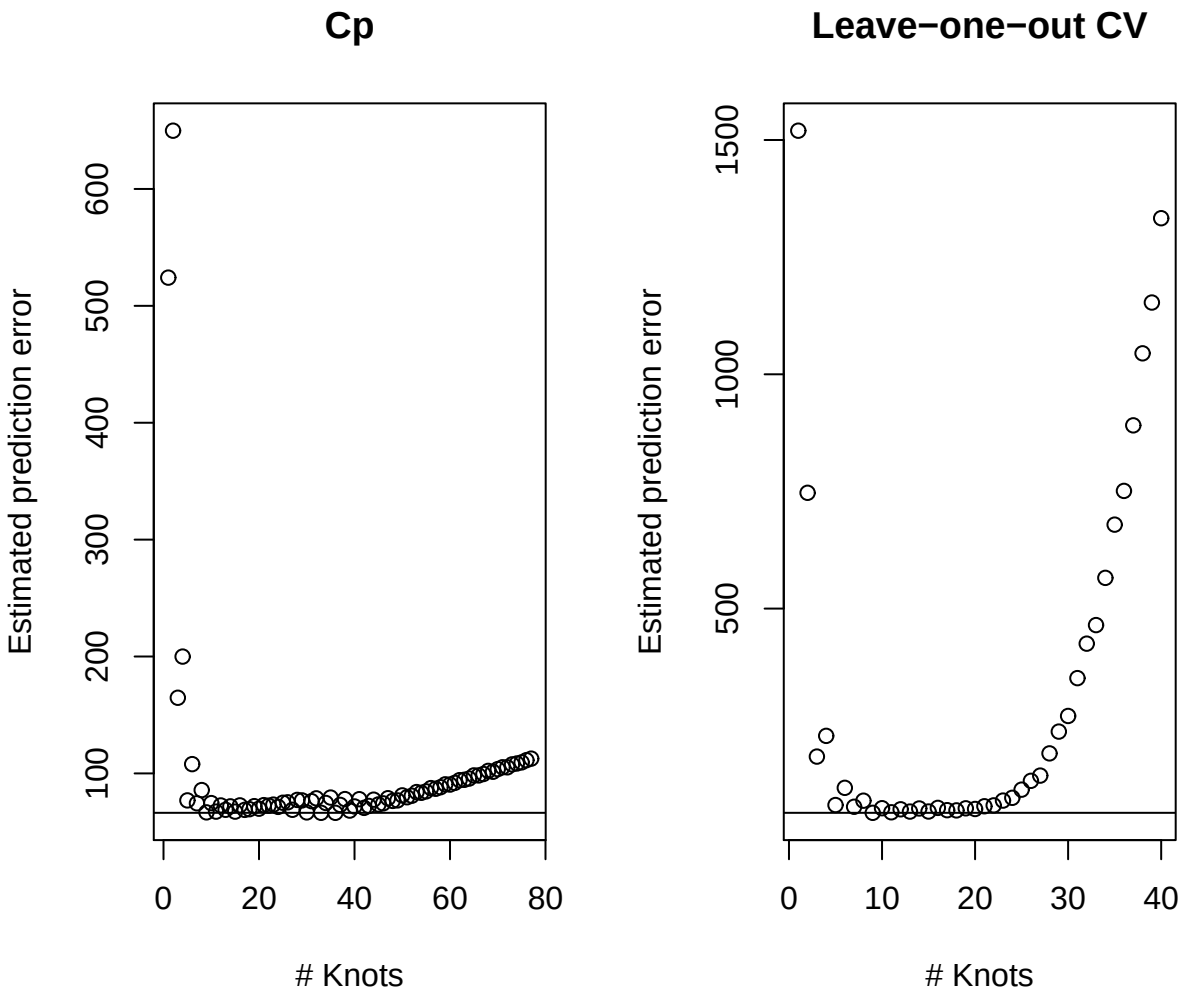


Figure 3.20: Estimates of prediction error for cubic splines

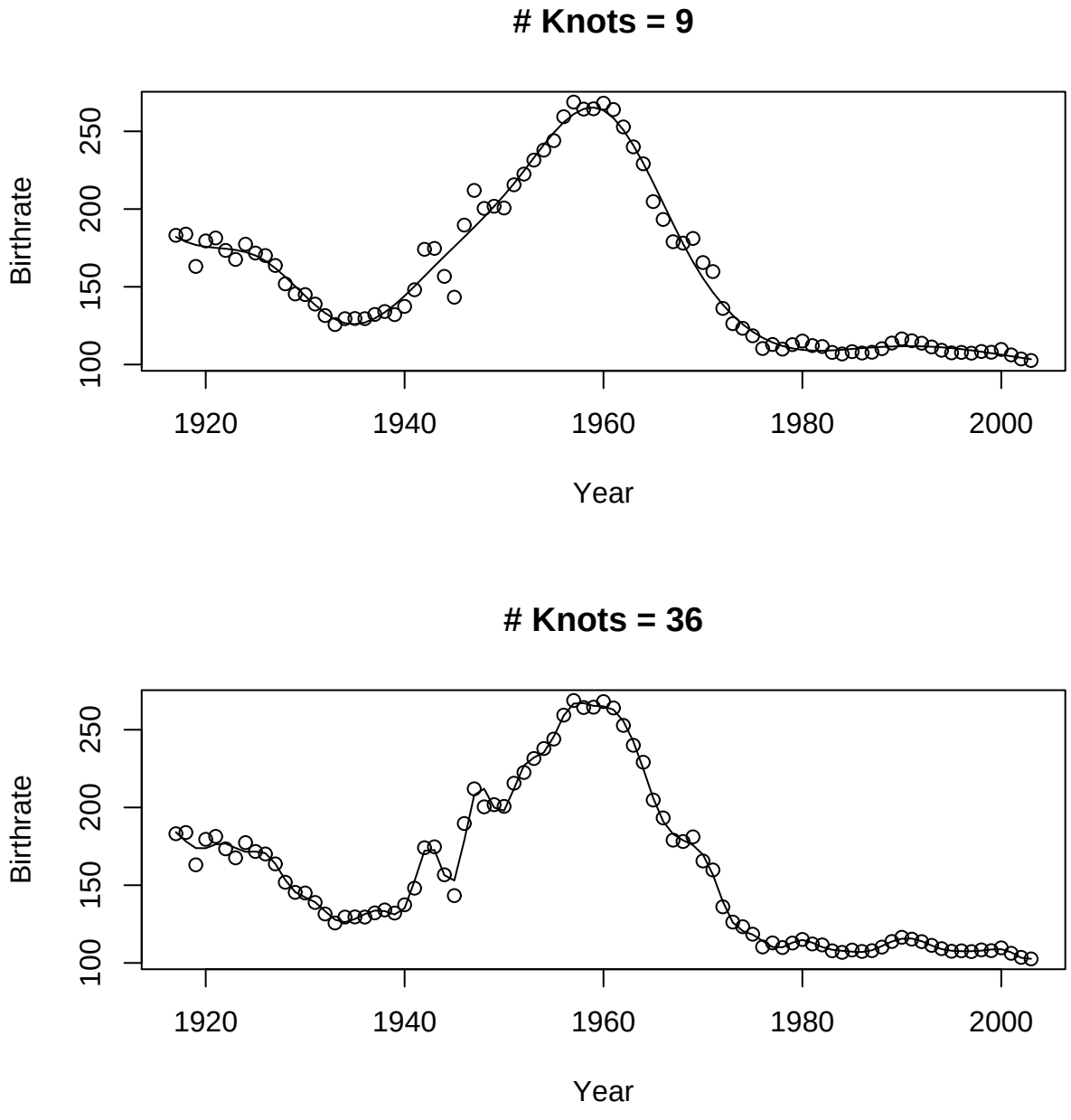


Figure 3.21: The cubic spline fits for 9 and 36 knots

fits:

Year	Polynomial $degree = 12$	Spline $K = 9$	Natural spline $K = 13$	Observed
2003	104.03	103.43	103.79	102.6
2004	113.57	102.56	102.95	(101.8)
2005	139.56	101.75	102.11	
2006	195.07	101.01	101.27	
2007	299.80	100.33	100.43	
2008	482.53	99.72	99.59	
2009	784.04	99.17	98.75	
2010	1260.96	98.69	97.91	

(3.63)

From the plot in Figure 3.21, we see that the birthrates from about 1976 on are not changing much, declining a bit at the end. Thus the predictions for 2004 and on should be somewhat smaller than that for around 2003. The polynomial fit does a very poor job, as usual, but both the regular cubic spline and the natural spline look quite reasonable, though the future is hard to predict.

3.3.1 Using R

Once you obtain the $\mathbf{X}$ matrix for a given fit, you use whatever linear model methods you wish. First, load the `splines` package. Letting `x` be your $\underline{x}$, and `y` be the $\underline{y}$, to obtain the cubic B-spline basis for K knots, use

```
xx <- bs(x,df=K+3)
```

The effective degrees of freedom are $K + 4$, but `bs` dos not return the $\underline{1}_N$ vector, and it calls the number of columns it returns `df`. To fit the model, just use

```
lm(y~xx)
```

For natural splines, use `ns`, where for K knots, you use `df = K+1`:

```
xx <- ns(x,df=K+1)
lm(y~xx)
```

The calls above pick the knots so that there are approximately the same numbers of observations in each region. If you wish to control where the knots are placed, then use the `knots` keyword. For example, for knots at 1936, 1960, and 1976, use

```
xx <- bs(x,knots=c(1936,1960,1976))
```

3.4 Smoothing splines

In the previous section, we fit splines using regular least squares. An alternative approach uses a penalty for non-smoothness, which has the effect of decreasing some of the regression coefficients, much as in ridge regression. The main task of this section will be to fit cubic splines, but we start with a simple cubic (or spline with zero knots). That is, the fit we are after is of the form

$$\widehat{y}_i = f(x_i) = b_0 + b_1 x_i + b_2 x_i^2 + b_3 x_i^3. \tag{3.64}$$

We know how to find the usual least squares estimates of the b_j's, but now we want to add a regularization penalty on the non-smoothness. There are many ways to measure non-smoothness. Here we will try to control the second derivative of f, so that we are happy with any slope, we do not want the slope to change too quickly. One common penalty is the integrated square of the second derivative:

$$\int (f''(x))^2 dx. \tag{3.65}$$

Suppose that the range of the x_i's is $(0, 1)$. Then the objective function for choosing the $\underline{b}$ is

$$obj_\lambda(\underline{b}) = \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \int_0^1 (f''(x))^2 dx \tag{3.66}$$

It is similar to ridge regression, except that ridge tries to control the b_j's directly, i.e., the slopes. Here,

$$f''(x) = 2b_2 + 6b_3 x, \tag{3.67}$$

hence

$$\begin{aligned} \int_0^1 (f''(x))^2 dx &= \int_0^1 (2b_2 + 6b_3 x)^2 dx \\ &= \int_0^1 (4b_2^2 + 24 b_2 b_3 x + 36 b_3 x^2) dx \\ &= 4b_2^2 + 12 b_2 b_3 + 12 b_3^2. \end{aligned} \tag{3.68}$$

Thus, letting $\mathbf{X}$ be the matrix with the cubic polynomial arguments,

$$obj_\lambda(\underline{b}) = \|\underline{y} - \mathbf{X}\underline{b}\|^2 + \lambda(4b_2^2 + 12 b_2 b_3 + 12 b_3^2). \tag{3.69}$$

Minimizing this objective function is a least squares task as in Section 2.2. The vector of derivatives with respect to the b_j's is

$$\begin{pmatrix} \frac{\partial}{\partial b_0} obj_\lambda(\underline{b}) \\ \frac{\partial}{\partial b_1} obj_\lambda(\underline{b}) \\ \frac{\partial}{\partial b_2} obj_\lambda(\underline{b}) \\ \frac{\partial}{\partial b_3} obj_\lambda(\underline{b}) \end{pmatrix} = -2\mathbf{X}'(\underline{y} - \mathbf{X}\underline{b}) + \lambda \begin{pmatrix} 0 \\ 0 \\ 8b_2 + 12 b_3 \\ 12 b_2 + 24 b_3 \end{pmatrix} = -2\mathbf{X}'(\underline{y} - \mathbf{X}\underline{b}) + \lambda \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 12 \\ 0 & 0 & 12 & 24 \end{pmatrix} \underline{b}. \tag{3.70}$$

Letting

$$\Omega = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 12 \\ 0 & 0 & 12 & 24 \end{pmatrix}. \tag{3.71}$$

setting the derivative vector in (3.70) to zero leads to

$$-2\mathbf{X}'\underline{y} + 2\mathbf{X}'\mathbf{X}\underline{b} + \lambda\Omega\underline{b} = -2\mathbf{X}'\underline{y} + (2\mathbf{X}'\mathbf{X} + \lambda\Omega)\underline{b} = \underline{0}_4, \tag{3.72}$$

or

$$\underline{\widehat{\beta}}_\lambda = (\mathbf{X}'\mathbf{X} + \frac{\lambda}{2}\,\Omega)^{-1}\mathbf{X}'\underline{y}. \tag{3.73}$$

Compare this estimator to the ridge estimate in (2.72). It is the same, but with the identity instead of $\Omega/2$. Note that with $\lambda = 0$, we have the usual least squares estimate of the cubic equation, but as $\lambda \to \infty$, the estimates for β_2 and β_3 go to zero, meaning the f approaches a straight line, which indeed has zero second derivative.

To choose λ, we can use the C_p for estimating the prediction error. The theory is exactly the same as for ridge, in Section 2.6.1, but with the obvious change. That is,

$$\widehat{ERR}_{in,\lambda} = \overline{err}_\lambda + 2\widehat{\sigma}_e^2\,\frac{edf(\lambda)}{N}, \tag{3.74}$$

where

$$edf(\lambda) = trace(\mathbf{X}(\mathbf{X}'\mathbf{X} + \frac{\lambda}{2}\,\Omega)^{-1}\mathbf{X}'). \tag{3.75}$$

Next, consider a cubic spline with K knots. Using the basis in (3.58) and (3.61), the penalty term contains the second derivatives within each region, squared and integrated:

$$\begin{aligned} \int_0^1 (f''(x))^2 dx &= \int_0^{k_1} (2b_2 + 6b_3 x)^2 dx + \int_{k_1}^{k_2} (2b_2 + 6b_3 x + 6b_4(x-k_1))^2 dx + \\ &\cdots + \int_{k_K}^1 (2b_2 + 6b_3 x + 6b_4(x-k_1) + \cdots + 6b_{K+3}(x-k_K))^2 dx. \end{aligned} \tag{3.76}$$

It is a tedious but straightforward calculus exercise to find the penalty, but one can see that it will be quadratic in $b_2, \ldots, b_{K+3}$. More calculus will yield the derivatives of the penalty with respect to the b_j's, which will lead as in (3.70) to a matrix Ω. The estimate of $\underline{\beta}$ is then (3.73), with the appropriate $\mathbf{X}$ and Ω. The resulting $\mathbf{X}\underline{\widehat{\beta}}_\lambda$ is the **smoothing spline**. Choosing λ using C_p proceeds as in (3.74).

A common choice for the smoothing spline is to place a knot at each distinct value of x_i. This approach avoids having to try various K's, although if N is very large one may wish to pick a smaller K. We use all the knots for the birthrates data. Figure 3.22 exhibits the estimated prediction errors (3.74) for various λ's (or edf's):

The best has $edf = 26$, with an estimated error of 63.60. There are other smaller edf's with almost as small errors. We will take $edf = 18$, which has an error estimate of 64.10. Figure 3.23 shows the fit. It is quite nice, somewhere between the fits in Figure 3.21.

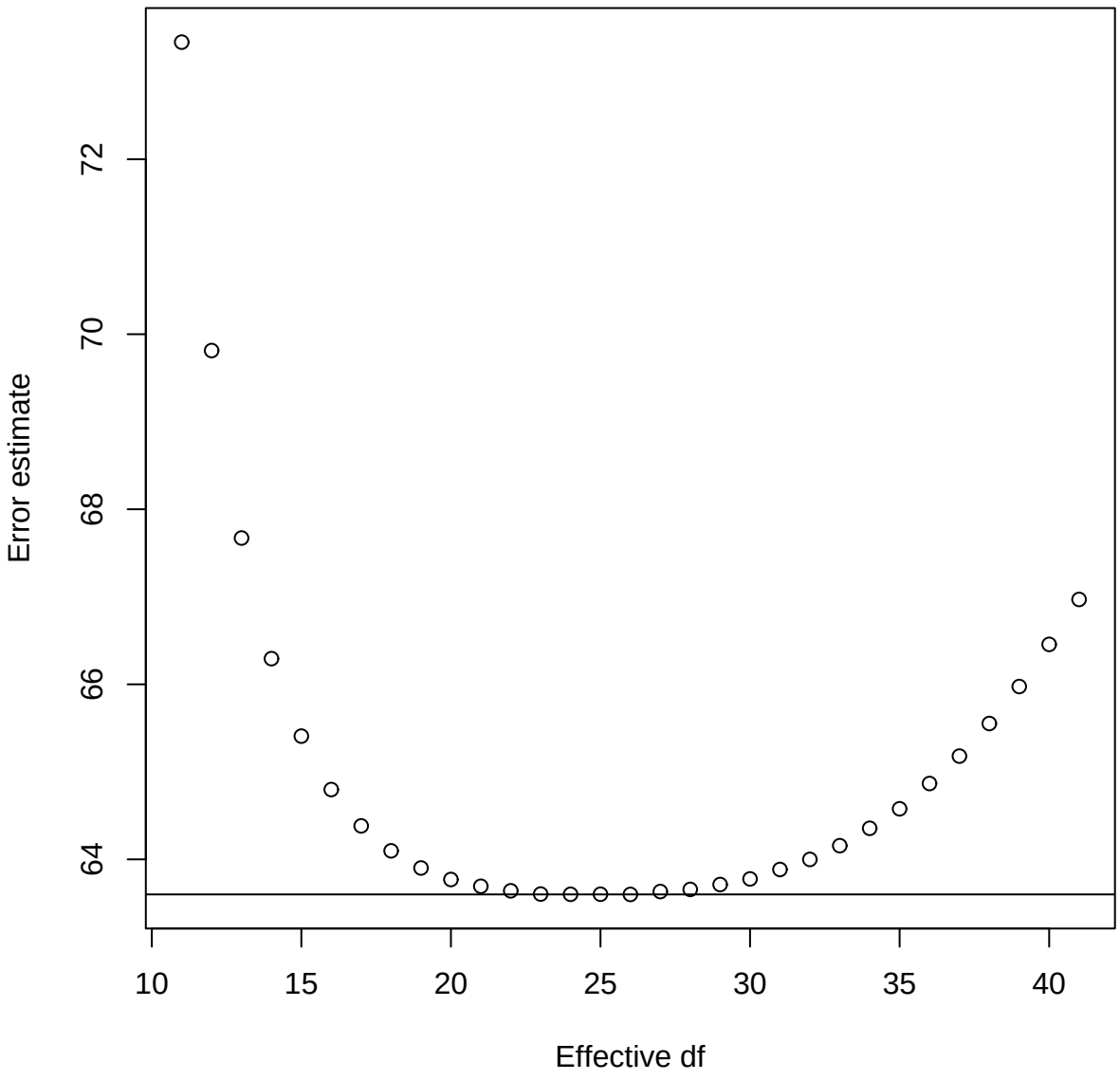


Figure 3.22: Estimates of prediction error for smoothing spline

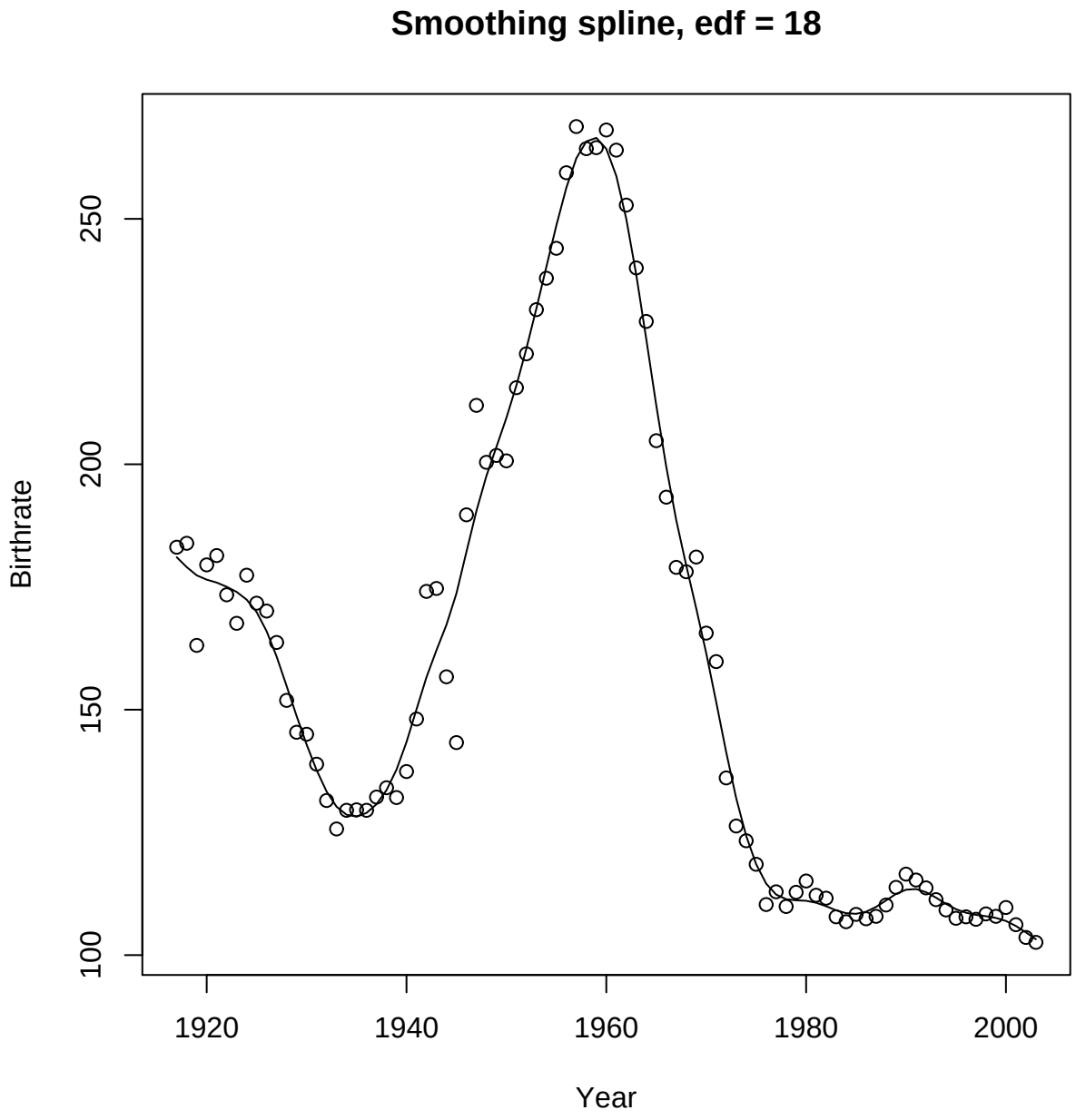


Figure 3.23: The smoothing spline with edf=18

Figure 3.24 compares four fitting procedures for the birthrates by looking at the estimated errors as functions of the effective degrees of freedom. Generally, the smoothing splines are best, the polynomials worst, and the regular and natural splines somewhere in between. The latter two procedures have jagged plots, because as we change the number of knots, their placements change, too, so that the estimated errors are not particularly smooth as a function of edf.

Among these polynomial-type fits, the smoothing spline appears to be the best one to use, at least for these data.

3.4.1 Using R

It is easy to use the `smooth.spline` function in *R*. As a default, it uses knots at the x_i's, or at least approximately so. You can also give the number of knots via the keyword `nknots`. The effective degrees of freedom are indicated by `df`. Thus to find the smoothing spline fit for $edf = 18$, use

```
x <- birthrates[,1]
y <- birthrates[,2]
ss <- smooth.spline(x,y,df=18)
```

Then `ss$y` contains the fitted y_i's. To plot the fit:

```
plot(x,y);lines(x,ss$y)
```

3.4.2 An interesting result

Suppose one does not wish to be restricted to cubic splines. Rather, one wishes to find the f among *all* functions with continuous second derivatives that minimizes the objective function (3.66). It turns out that the minimizer is a natural cubic spline with knots at the distinct values of the x_i's, which is what we have calculated above.

3.5 A glimpse of wavelets

The models in the previous sections assumed a fairly smooth relation between x and y, without spikes or jumps. In some situations, jumps and spikes are exactly what one is looking for, e.g., in earthquake data, or images where there are sharp boundaries between colors of pixels. Wavelets provide alternative bases to the polynomial- and trigometric-based ones we have used so far. Truly understanding wavelets is challenging. Rather than making that attempt, we will look at the implementation of a particularly simple type of wavelet, then wave our hands in order to try more useful wavelets.

Figure 3.25 is a motivating plot of Bush's approval rating, found at *Professor Pollkatz's Pool Of Polls*[5]

[5] `http://www.pollkatz.homestead.com/`, the *Bush Index*.

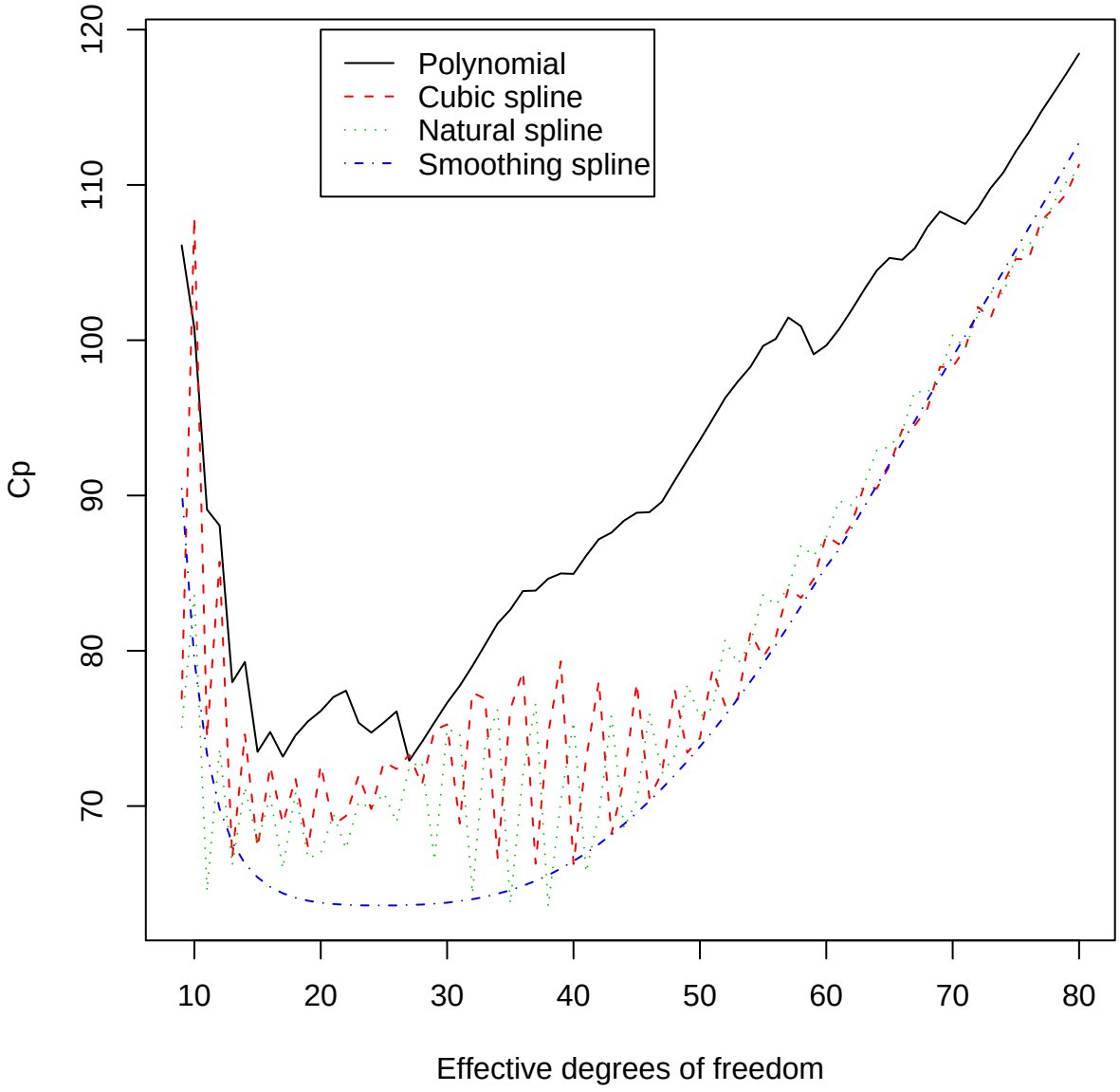


Figure 3.24: Estimated errors for various fits

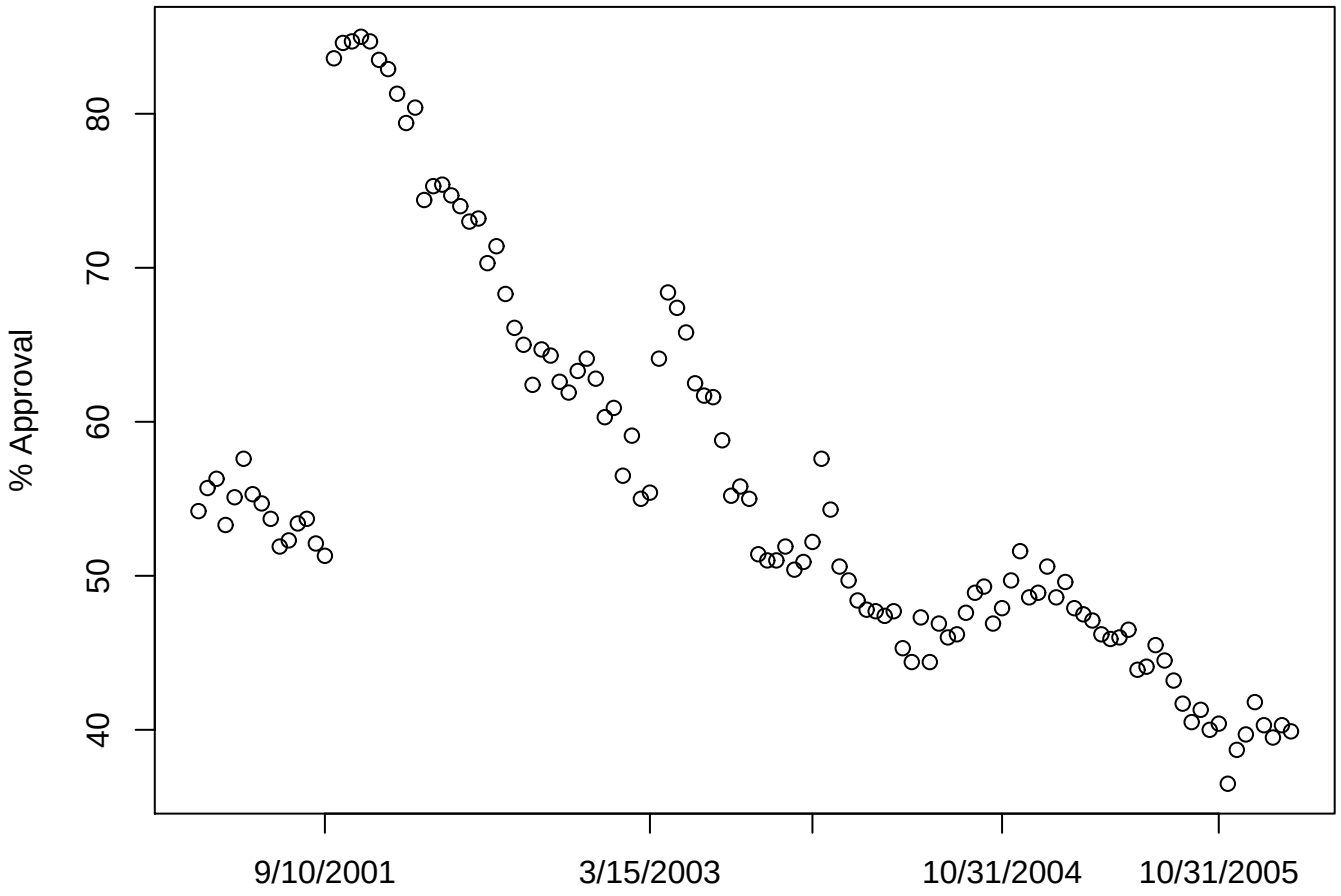


Figure 3.25: Bush's approval ratings

Though most of the curve is smooth, often trending down gently, there are notable spikes: Right after 9/11, when the Iraq War began, a smaller one when Saddam was captured, and a dip when I. Lewis "Scooter" Libby was indicted. There is also a mild rise in Fall 2004, peaking at the election[6]. Splines may not be particularly successful at capturing these jumps, as they are concerned with low second derivatives.

3.5.1 Haar wavelets

The simplest wavelets are **Haar** wavelets, which generalize the bases used in regressograms. A regressogram (as in Figure 3.16) divides the x-variable into several regions, then places a horizontal line above each region, the height being the average of the corresponding y's. Haar wavelets do the same, but are flexible in deciding how wide each region should be. If the curve is relatively flat over a wide area, the fit will use just one or a few regions in that area. If the curve is jumpy in another area, there will be many regions used in that area. The wavelets adapt to the local behavior, being smooth were possible, jumpy where necessary. Figure 3.26 shows four fits to the phoneme data[7] in Figure 3.1. This graph is a periodogram based on one person saying " ". Notice, especially in the bottom two plots, that the flat regions have different widths.

The book (Section 5.9) has more details on wavelets than will be presented here. To give the idea, we will look at the $\mathbf{X}$ matrix for Haar wavelets when $N = 8$, but it should be clear how to extend the matrix to any N that is a power of 2:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & -1 \\ \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{4}} & \frac{1}{\sqrt{4}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \tag{3.77}$$

The unusual notation indicates that all elements in each column are divided by the square root in the last row, so that the columns all have norm 1. Thus $\mathbf{X}$ is an orthogonal matrix. Note that the third and fourth columns are basically like the second column, but with the range cut in half. Similarly, the last four columns are like the third and fourth, but with the range cut in half once more. This $\mathbf{X}$ then allows both global contrasts (i.e., the first half of the data versus the last half) and very local contrasts (y_1 versus y_2, y_3 versus y_4, etc.), and in-between contrasts.

[6] It looks like Bush's numbers go down unless he has someone to demonize: Bin Laden, Saddam, Saddam, Kerry.

[7] `http://www-stat.stanford.edu/~Etibs/ElemStatLearn/datasets/phoneme.info`

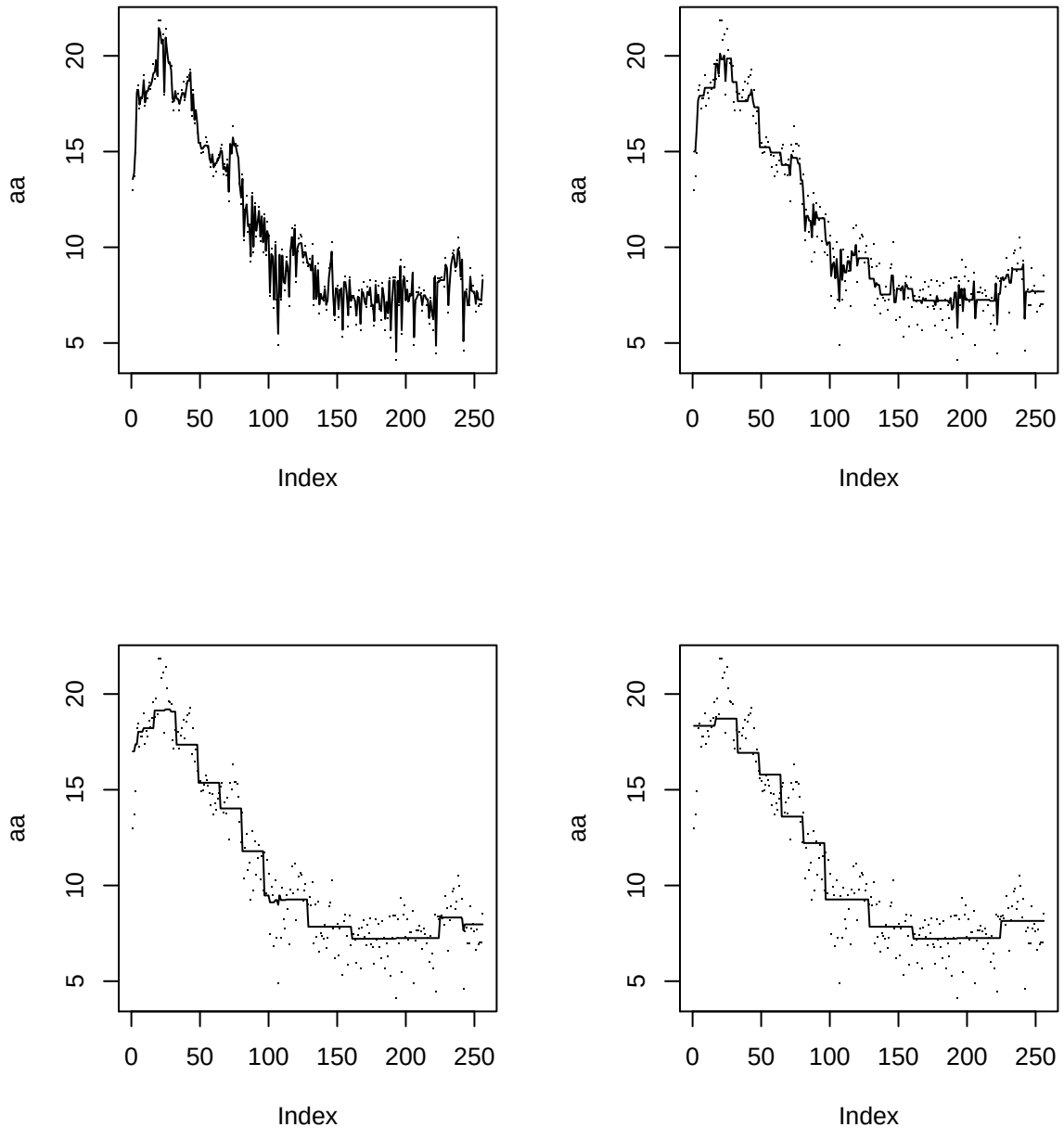


Figure 3.26: Haar wavelet fits to the phoneme data

The orthogonality of $\mathbf{X}$ means that the usual least squares estimate of the $\underline{\beta}$ is

$$\underline{\widehat{\beta}}_{LS} = \mathbf{X}'\underline{y}. \tag{3.78}$$

The two most popular choices for deciding on the estimate of $\underline{\beta}$ to use in prediction with wavelets are subset regression and lasso regression. Subset regression is easy enough because of the orthonormality of the columns of $\mathbf{X}$. Thus, assuming that β_0 is kept in the fit, the best fit using $p^* + 1$ of the coefficients takes $\widehat{\beta}_{LS,0}$ (which is $\overline{y}$), and the p^* coefficients with the largest absolute values. Thus, as when using the sines and cosines, one can find the best fits for each p^* directly once one orders the coefficients by absolute value.

The lasso fits are almost as easy to obtain. The objective function is

$$obj_\lambda(\underline{b}) = \|\underline{y} - \mathbf{X}\underline{b}\|^2 + \lambda \sum_{j=1}^{N-1} |b_j|. \tag{3.79}$$

Because $\mathbf{X}$ is orthogonal,

$$\|\underline{y} - \mathbf{X}\underline{b}\|^2 = \|\mathbf{X}'\underline{y} - \underline{b}\|^2 = \|\underline{\widehat{\beta}}_{LS} - \underline{b}\|^2 = \sum_{j=0}^{N-1} (\widehat{\beta}_{LS,j} - b_j)^2, \tag{3.80}$$

hence

$$obj_\lambda(\underline{b}) = (\widehat{\beta}_{LS,0} - b_0)^2 + \sum_{j=1}^{N-1} \left((\widehat{\beta}_{LS,j} - b_j)^2 + \lambda |b_j| \right). \tag{3.81}$$

The objective function decomposes into N components, each depending on just on b_j, so that the global minimizer $\underline{b}$ can be found by minimizing each component

$$(\widehat{\beta}_{LS,j} - b_j)^2 + \lambda |b_j| \tag{3.82}$$

over b_j. To minimize such a function, note that the function is strictly convex in b_j, and differentiable everywhere except at $b_j = 0$. Thus the minimum is where the derivative is 0, if there is such a point, or at $b_j = 0$ if not. Now

$$\frac{\partial}{\partial b} \left((\widehat{\beta} - b)^2 + \lambda |b| \right) = -2(\widehat{\beta} - b) + \lambda \; Sign(b) \quad \text{if } b \neq 0. \tag{3.83}$$

Setting that derivative to 0 yields

$$b = \widehat{\beta} - \frac{\lambda}{2} \; Sign(b). \tag{3.84}$$

There is no b as in (3.84) if $|\widehat{\beta}| < \lambda/2$, because then if $b > 0$, it must be $b < 0$, and if $b < 0$, it must be $b > 0$. Otherwise, b is $\widehat{\beta} \pm \lambda/2$, where the $\pm$ is chosen to move the coefficient closer to zero. That is,

$$b = \begin{cases} \widehat{\beta} - \frac{\lambda}{2} & \text{if} \quad \widehat{\beta} > \frac{\lambda}{2} \\ 0 & \text{if} \quad |\widehat{\beta}| \leq \frac{\lambda}{2} \\ \widehat{\beta} + \frac{\lambda}{2} & \text{if} \quad \widehat{\beta} < \frac{\lambda}{2}. \end{cases} \tag{3.85}$$

Wavelet people like to call the above methods for obtaining coefficients **thresholding**. In our notation, a threshold level $\lambda/2$ is chosen, then one performs either *hard* or *soft* thresholding, being subset selection or lasso, respectively.

- **Subset selection ≡ Hard thresholding**. Choose the coefficients for the fit to be the least squares estimates whose absolute values are greater than $\lambda/2$;
- **Lasso ≡ Soft thresholding**. Choose the coefficients for the fit to be the lasso estimates as in (3.85).

Thus either method sets to zero any coefficient that does not meet the threshold. Soft thresholding also shrinks the remaining towards zero.

3.5.2 An example of another set of wavelets

Haar wavelets are simple, but there are many other sets of wavelets with various properties that may be more flexible in fitting both global features and local features in a function. These other wavelets do not have recognizable functional forms, but digitalized forms can be found. In general, a family of wavelets starts with two functions, a *father* wavelet and a *mother* wavelet. The father and mother wavelets are orthogonal and have squared integral of 1. The wavelets are then replicated by shifting or rescaling. For example, suppose ψ is the mother wavelet. It is shifted by integers, that is, for integer k,

$$\psi_{0,k}(x) = \psi(x-k). \tag{3.86}$$

These are the **level 0** wavelets. The wavelet can be scaled by expanding or contracting the x-axis by a power of 2:

$$\psi_{j,0}(x) = 2^{j/2}\psi(2^j x). \tag{3.87}$$

Further shifting by integers k yields the **level** j wavelets,

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k). \tag{3.88}$$

Note that k and j can be any integers, not just nonnegative ones. The amazing property of the entire set of wavelets, the father plus all levels of the mother, is that they are mutually orthogonal. (They also have squared integral of 1, which is not so amazing.) The Haar father and mother are

$$\phi_{Father}(x) = \begin{cases} 1 & \text{if } 0 < x < 1 \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad \psi_{Mother}(x) \begin{cases} 1 & \text{if } 0 < x < 1/2 \\ -1 & \text{if } 1/2 \le x < 1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.89}$$

Figures 3.27 and 3.28 show some mother wavelets of levels 0, 1 and 2.

Another set of wavelets is the "Daubechies orthonormal compactly supported wavelet N=2, extremal phase family (DaubEx 2)." We choose this because it is the default family

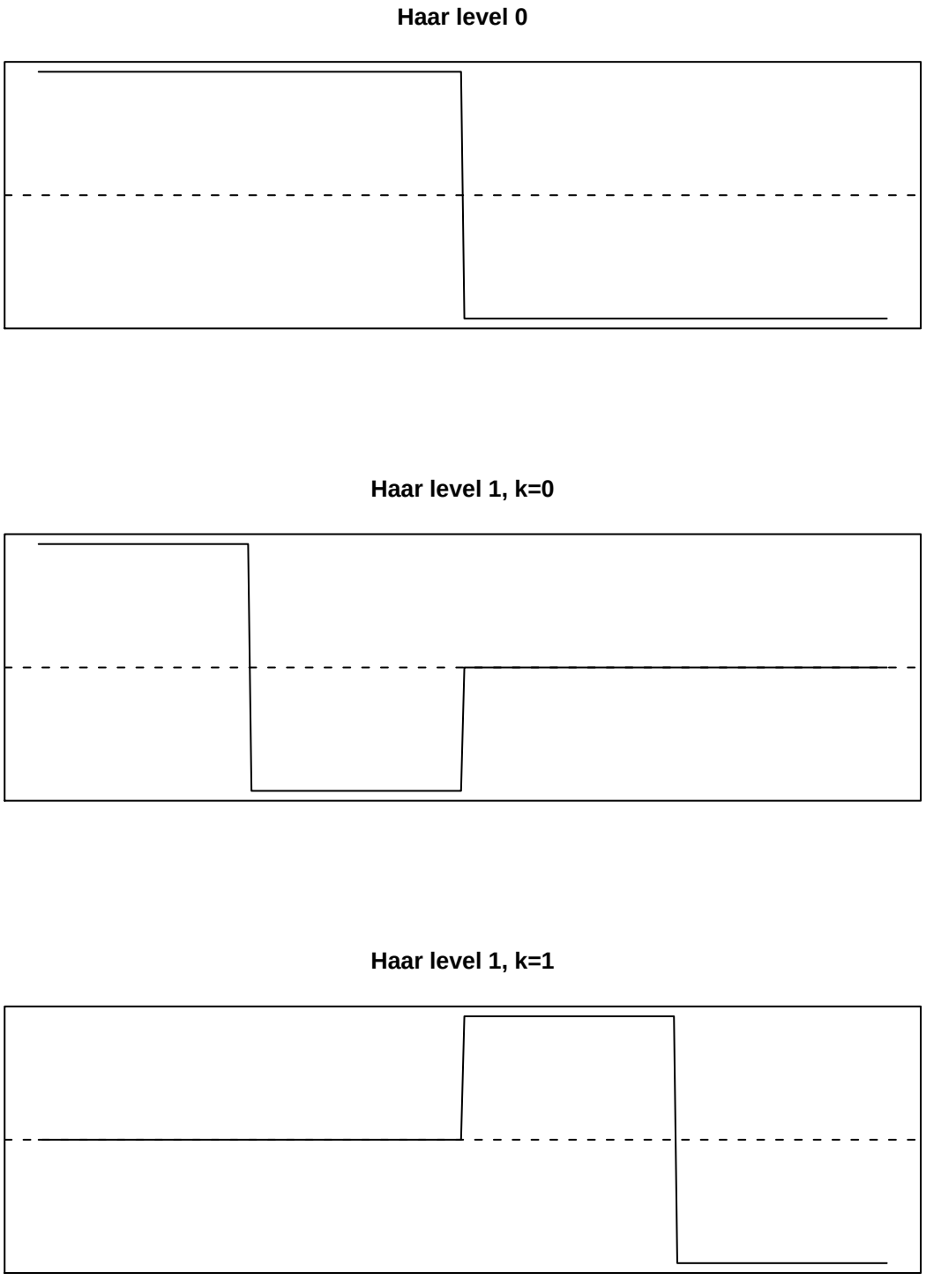


Figure 3.27: Some Haar wavelets

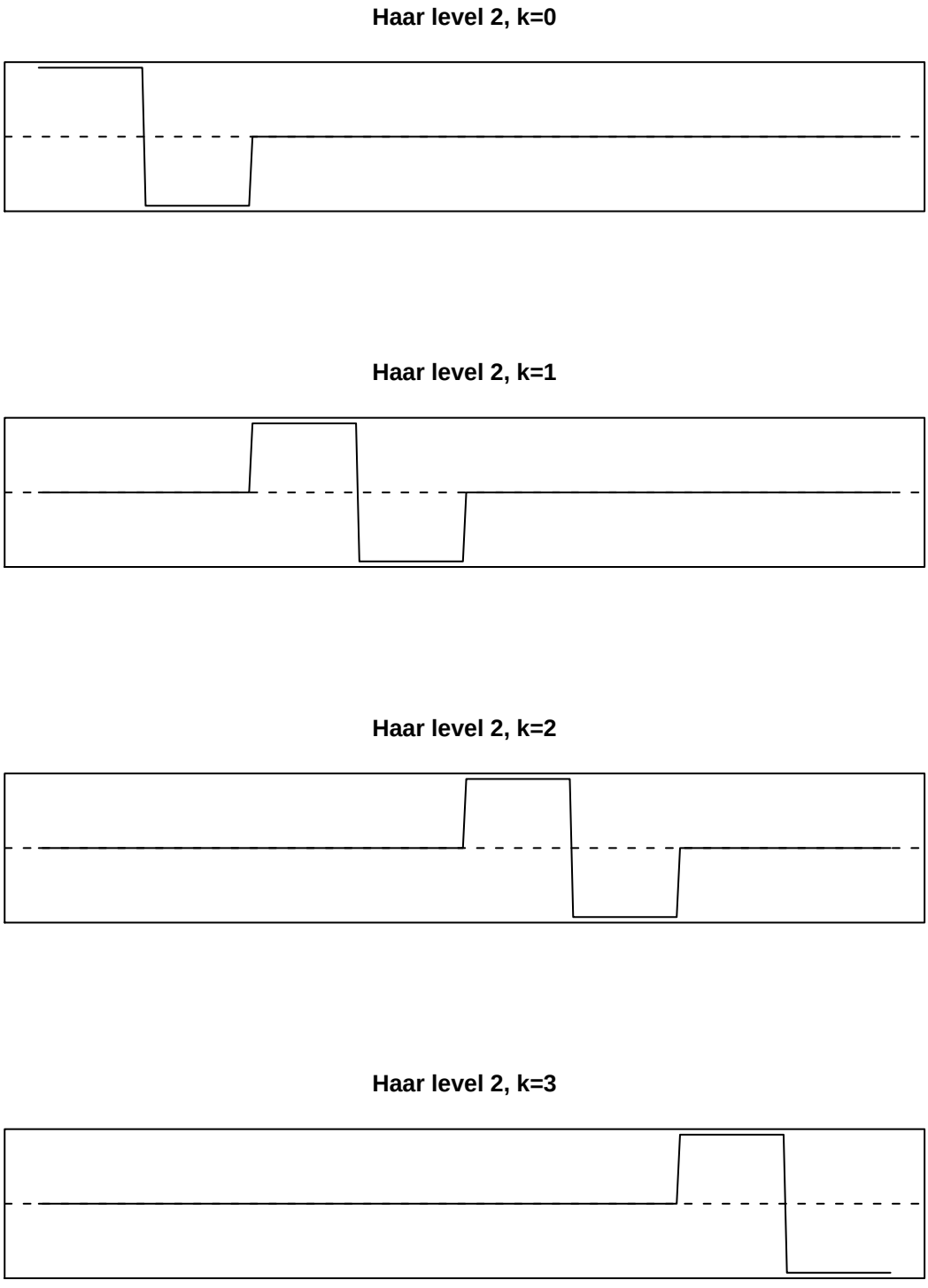


Figure 3.28: More Haar wavelets

used in the R package `wavethresh`[8]. The domains of the DaubEx 2 wavelets of a given level overlap, unlike the Haar wavelets. Thus the wavelets are not always contained within the range of x's for a given data set. One fix is to recycle the part of the wavelet that occurs after the last of the observed x's at the beginning of the x's. This is the approach in Figures 3.29 and 3.30.

3.5.3 Example Using R

We will fit DaubEx 2 wavelets to the phoneme data using `wavethresh`. The y-values are in the vector `aa`. There are three steps:

1. **Decomposition**. Find the least squares estimates of the coefficients:

   ```
   w <- wd(aa)
   ```

   This `w` contains the estimates in `w[[2]]`, but in a non-obvious order. To see the coefficients for the level d wavelets, use `accessD(w,level=d)`. To see the plot of the coefficients as in Figure 3.31, use `plot(w)`.

2. **Thresholding**. Take the least squares estimates, and threshold them. Use hard or soft thresholding depending on whether you wish to keep the ones remaining at their least squares values, or decrease them by the threshold value. The default is hard thresholding. If you wish to input your threshold, `lambda/2`, use the following:

   ```
   whard <- threshold(w,policy="manual",value=lambda/2)  # Hard
   wsoft <- threshold(w,policy="manual",value=lambda/2,type="soft")  # Soft
   ```

   The default, `wdef <- threshold(w)`, uses the "universal" value of Donoho and Johnstone, which takes

   $$\lambda = 2s\sqrt{2\log(N)}, \tag{3.90}$$

   where s is the sample standard deviation of the least squares coefficients. The idea is that if the β_j's are all zero, then their least squares estimates are iid $N(0, \sigma_e^2)$, hence

   $$E[\max|\widehat{\beta}_j|] \approx \sigma_e\sqrt{2\log(N)}. \tag{3.91}$$

   Coefficients with absolute value below that number are considered to have $\beta_j = 0$, and those above to have $\beta_j \neq 0$, thus it makes sense to threshold at an estimate of that number.

3. **Reconstruction**. Once you have the thresholded estimates, you can find the fits $\underline{\widehat{y}}$ using

[8] Author: Guy Nason of R-port: Arne Kovac (1997) and Martin Maechler (1999).

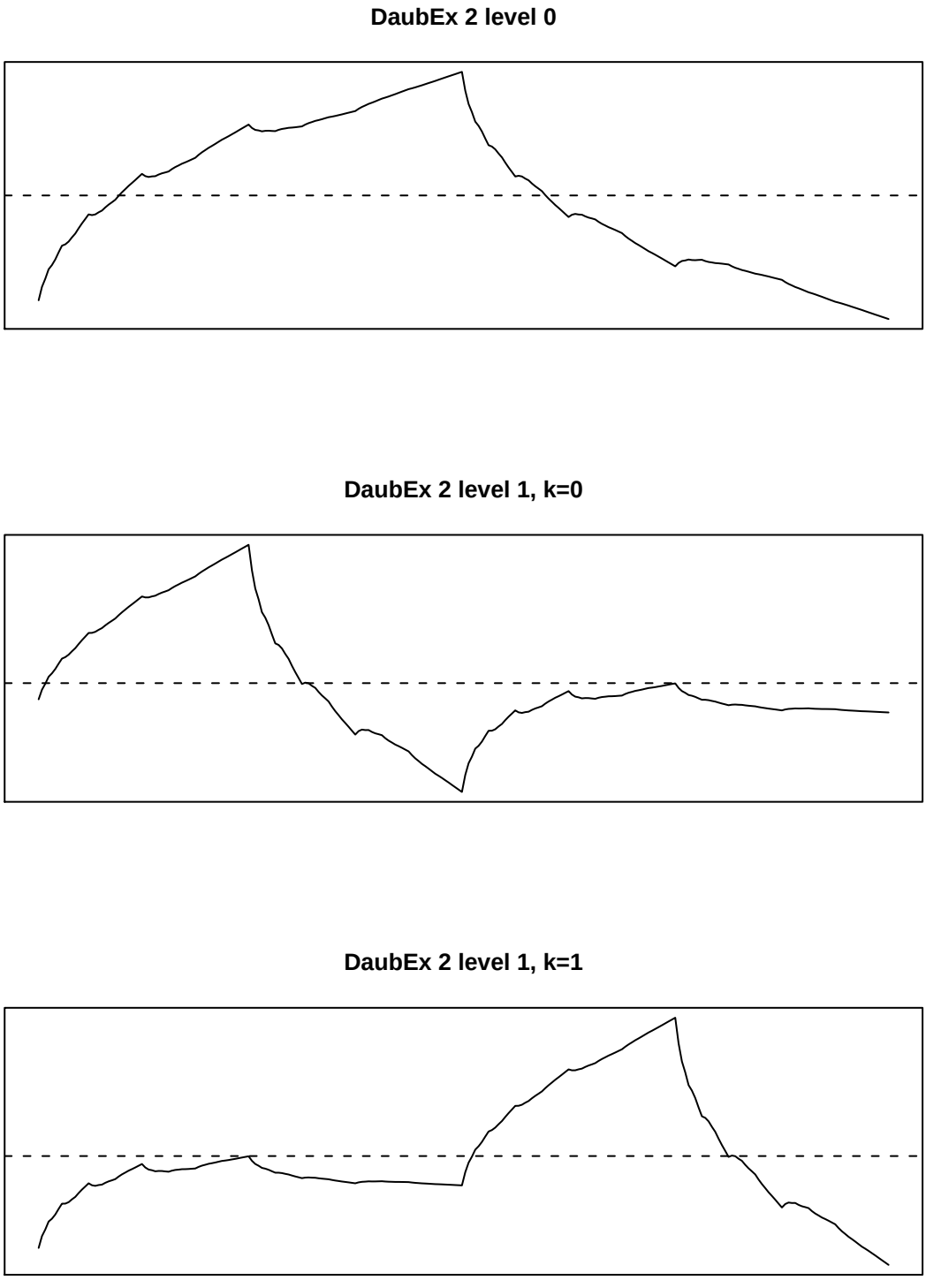


Figure 3.29: Some DaubEx 2 wavelets

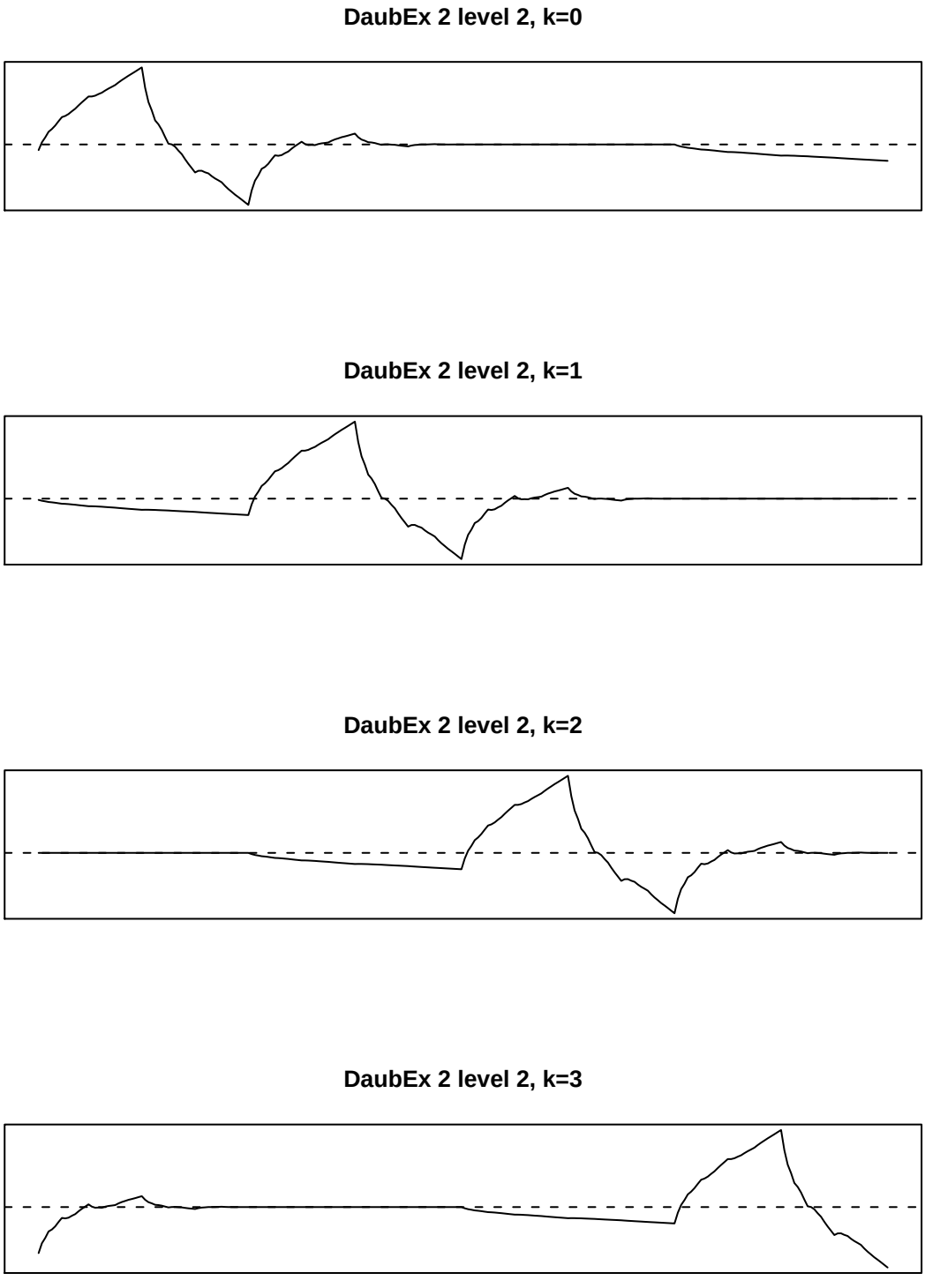


Figure 3.30: More DaubEx 2 wavelets

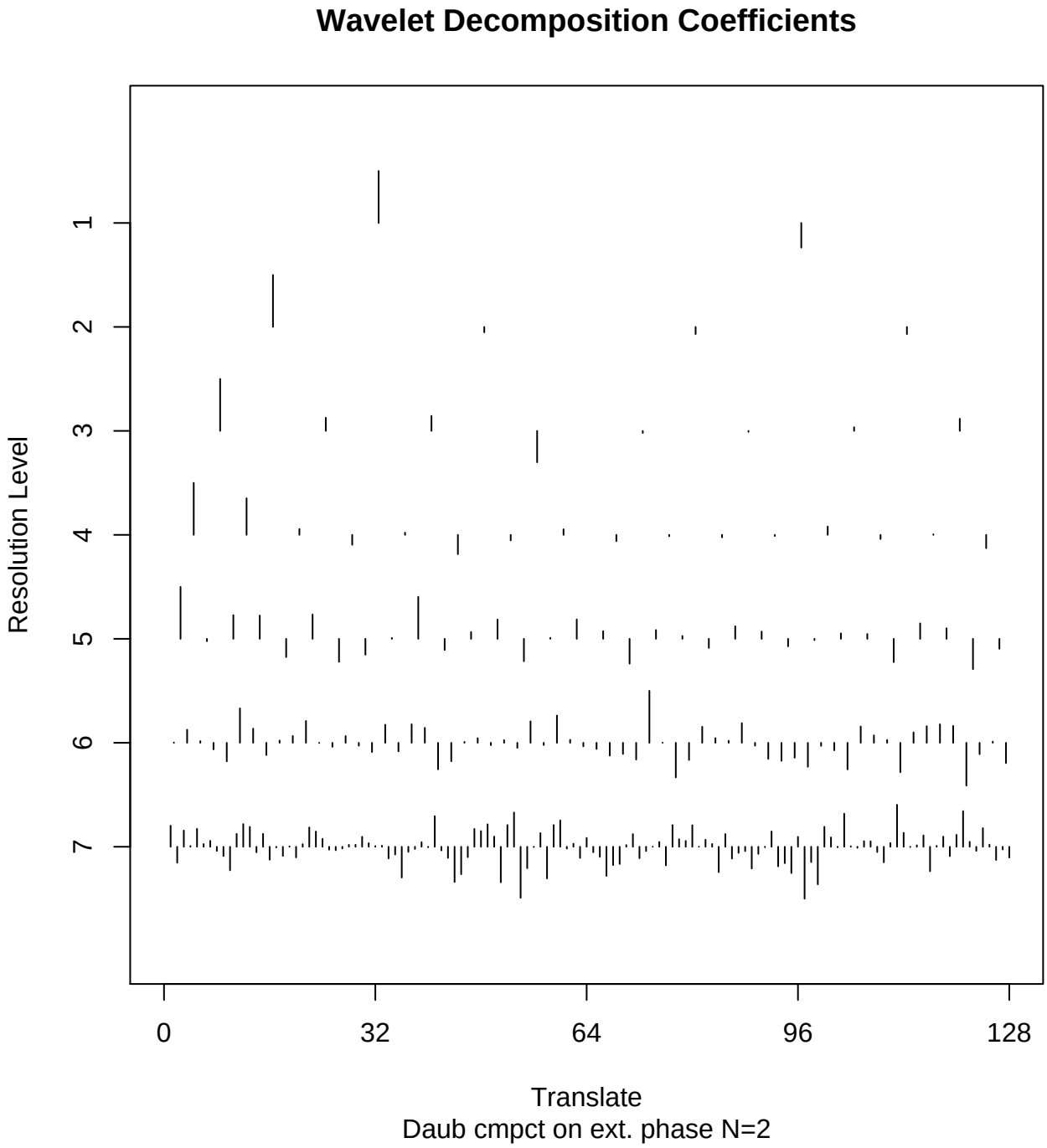


Figure 3.31: Least squares estimates of the coefficients.

```
yhat <- wr(whard) # or wr(wsoft) or wr(wdef)
```

To choose λ, you could just use the universal value in (3.90). The C_p approach is the same as before, e.g., for sines and cosines or polynomials. For each value of λ you wish to try, find the fits, then from that find the residual sums of squares. Then as usual

$$\widehat{ERR}_{in,\lambda} = \overline{err}_{in,\lambda} + 2\widehat{\sigma}_e^2 \, \frac{edf}{N}. \tag{3.92}$$

We need some estimate of σ_e^2. Using the QQ-plot on the $\widehat{\beta}_j^2$'s (which should be compared to χ_1^2's this time), we estimate the top seven coefficients can be removed, and obtain $\widehat{\sigma}_e^2 = 2.60$. The effective degrees of freedom is the number of coefficients not set to zero, plus the grand mean, which can be obtained using `dof(whard)`. Figure 3.32 shows the results. Both have a minimum at $edf = 23$, where the threshold value was 3.12. The universal threshold is 15.46, which yielded $edf = 11$. Figure 3.33 contains the fits of the best subset, best lasso, and universal fits. The soft threshold fit (lasso) is less spiky than the hard threshold fit (subset), and fairly similar to the fit using the universal threshold. All three fits successfully find the spike at the very left-hand side, and are overall preferable to the Haar fits in 3.26.

3.5.4 Remarks

Remark 1. Wavelets themselves are continuous function, while applying them to data requires digitalizing them, i.e., use them on just a finite set of points. (The same is true of polynomials and sines and cosines, of course.) Because of the dyadic way in which wavelets are defined, one needs to digitize on $N = 2^d$ equally-spaced values in order to have the orthogonality transfer. In addition, wavelet decomposition and reconstruction uses a clever pyramid scheme that is infinitely more efficient than using the usual $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\underline{y}$ approach. This approach needs the $N = 2^d$ as well. So the question arises of what to do when $N \neq 2^d$. Some *ad hoc* are to remove a few observations from the ends, or tack on a few fake observations to the ends, if that will bring the number of points to a power of two. For example, in the birthrate data, $N = 133$, so removing five points leaves 2^7. Another possibility is to apply the wavelets to the first 2^d points, and again to the last 2^d points, then combine the two fits on the overlap.

For example, the Bush approval numbers in 3.25 has $N = 122$. Figure 3.34 shows the fits to the first 64 and last 64 points, where there is some discrepancy in the overlapping portion. In practice, one would average the fits in the overlap, or use the first fit up until point 61, and the second fit from point 62 to the end. In any case, not how well these wavelets pick up the jumps in approval at the crucial times.

Remark 2. The problems in the above remark spill over to using cross-validation with wavelets. Leaving one observation out, or randomly leaving out several, will ruin the efficient calculation properties. Nason [1996] has some interesting ways around the problem. When

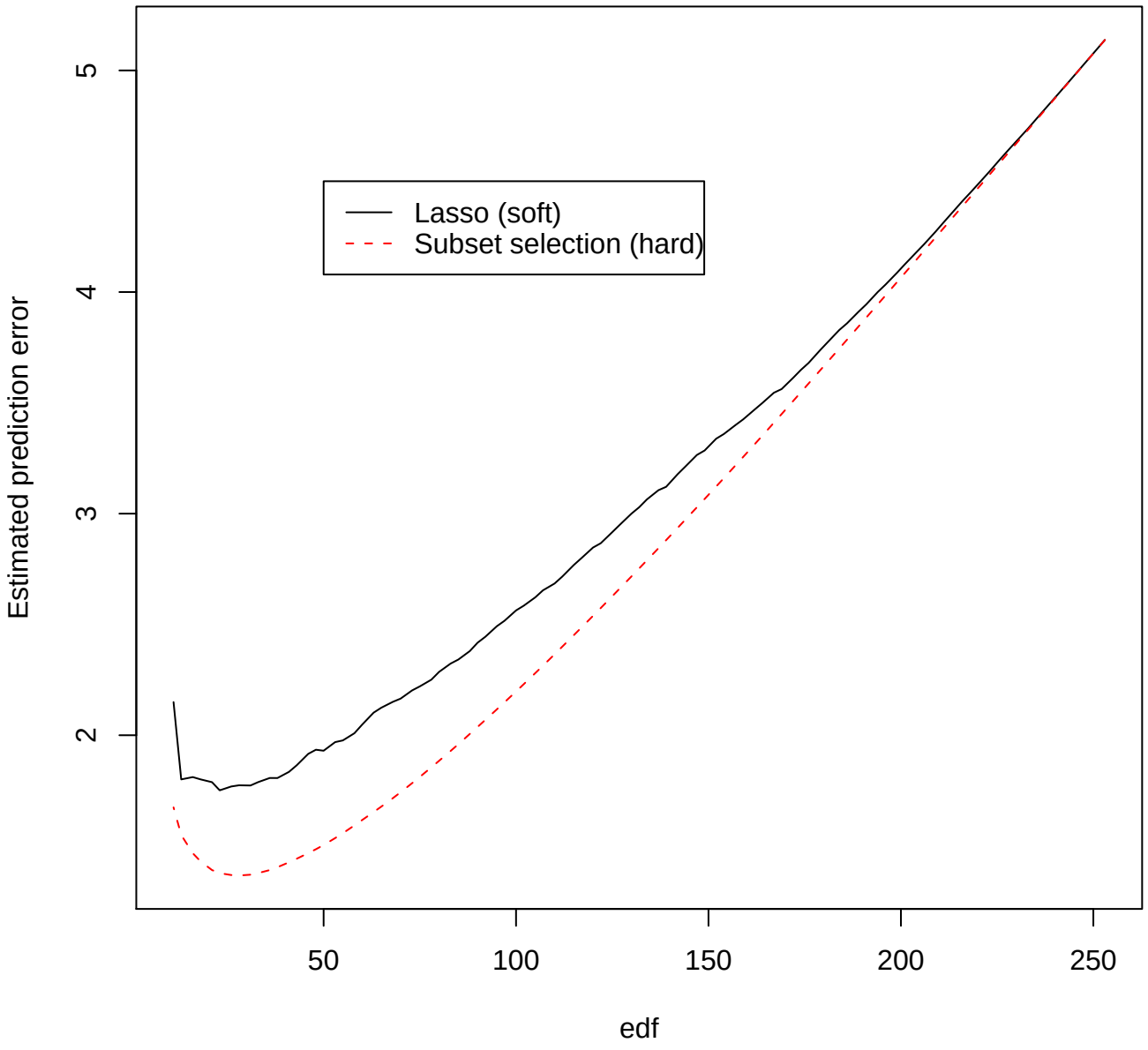


Figure 3.32: Prediction errors for the subset and lasso estimates.

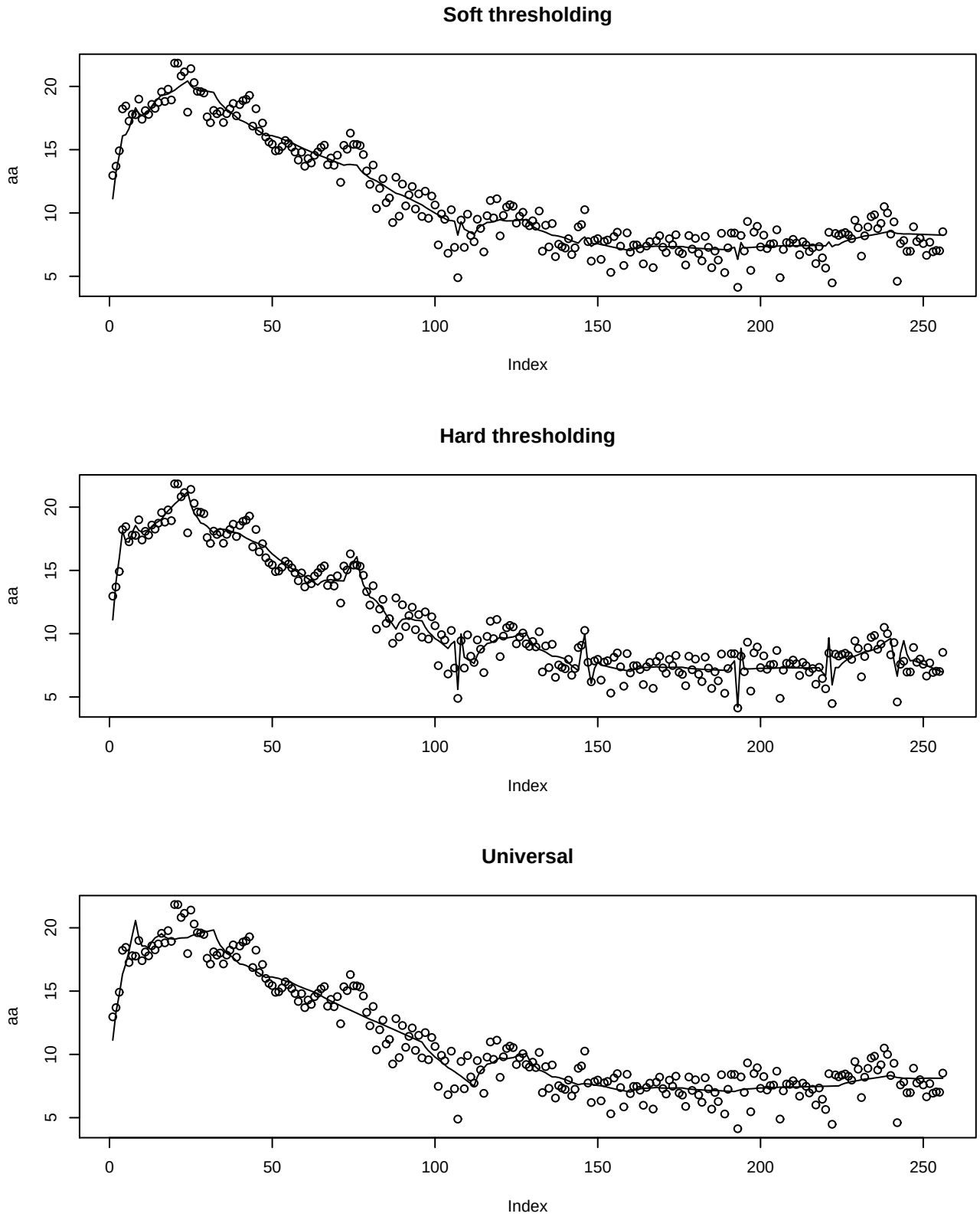


Figure 3.33: The DaubEx 2 fits to the phoneme data.

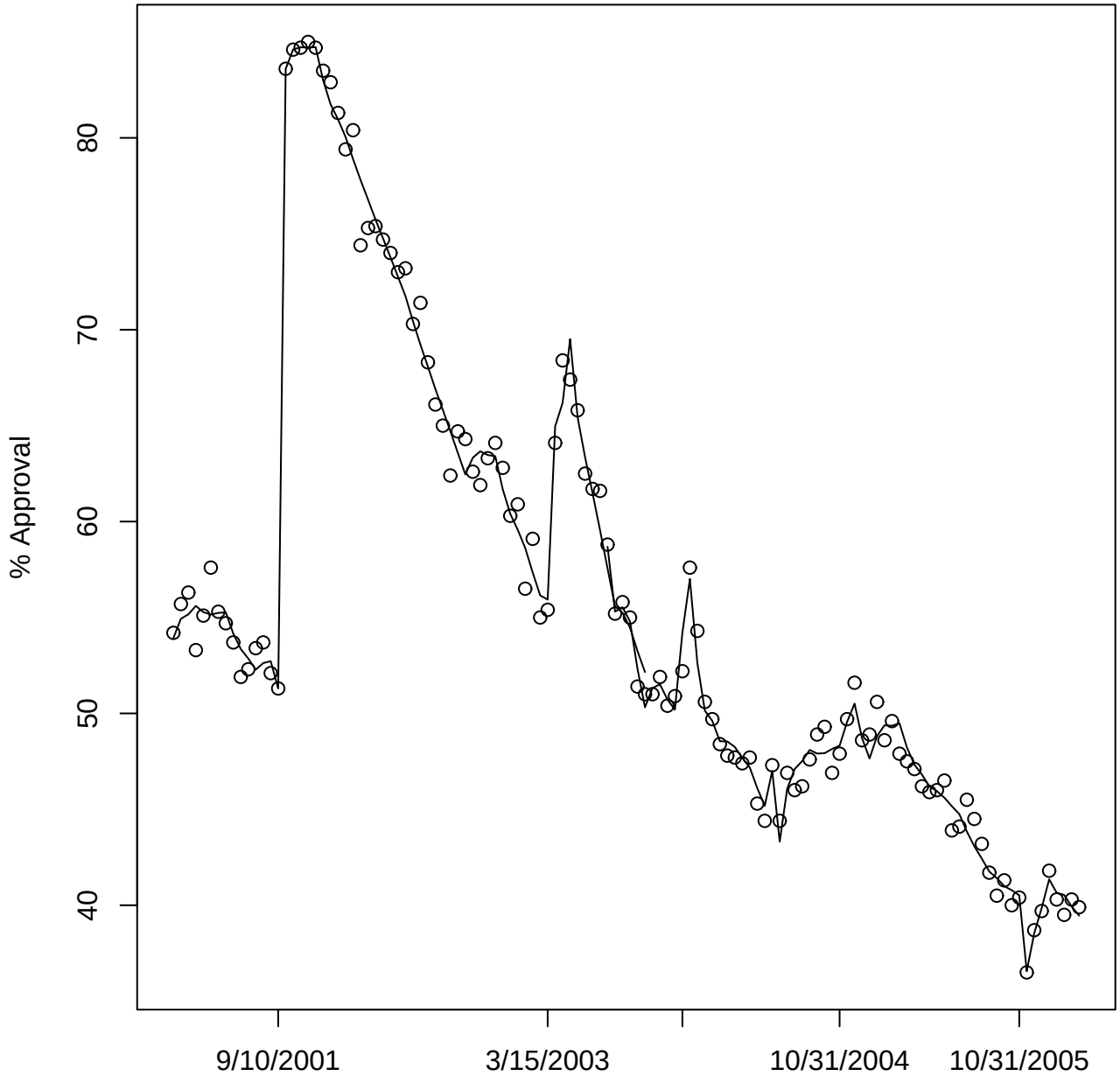


Figure 3.34: The fits to the Bush approval data

N is not too large, we can use the regular least squares formula (3.15) for leave-one-out cross-validation (or any cv) as long as we can find the $\mathbf{X}$ matrix. Then because $\mathbf{X}^{*'}\mathbf{X}^* = \mathbf{I}_{p^*+1}$,

$$h_{ii} = \|\underline{x}_i^*\|^2, \tag{3.93}$$

where $\underline{x}_i^*$ is the i^{th} row of the $\mathbf{X}^*$ matrix used in the fit. Because $\underline{\widehat{\beta}}_{LS} = \mathbf{X}'\underline{y}$, one way to find the $\mathbf{X}$ matrix for the wavelets is to find the coefficients when $\underline{y}$ has components with 1 in the i^{th} place and the rest 0. Then the least squares coefficients form the i^{th} row of the $\mathbf{X}$. In R, for $N = 256$,

```
z <- diag(256) # I_256
xd2 <- NULL
for(i in 1:256) xd2 <- rbind(xd2,wd(z[,i])[[2]])
```

Unfortunately, as we saw in the sines and cosines example in Section 3.2.2, leave-one-out cross-validation for the phoneme data leads to the full model.

Chapter 4

Model-based Classification

Classification is prediction for which the y_i's take values in a finite set. Two famous examples are

- **Fisher/Anderson Iris Data**. These data have 50 observations on each of three iris species (setosa, versicolor, and virginica). ($N = 150$.) There are four variables: sepal length and width, and petal length and width. Figure 4.1 exhibits the data in a scatter plot matrix.

- **Hewlett-Packard Spam Data**[1]. A set of $N = 4601$ emails were classified into either spam or not spam. Variables included various word and symbol frequencies, such as frequency of the word "credit" or "George" or "hp." The emails were sent to George Forman (not the boxer) at Hewlett-Packard labs, hence emails with the words "George" or "hp" would likely indicate non-spam, while "credit" or "!" would suggest spam.

We assume the training sample consists of $(y_1, \underline{x}_1), \ldots, (y_N, \underline{x}_N)$ iid pairs, where the $\underline{x}_i$'s are $p \times 1$ vectors of predictors, and the y_i's indicate which group the observation is from:

$$y_i \in \{1, 2, \ldots, K\}. \tag{4.1}$$

In the iris data, $N = 150$, $p = 4$ variable, and $K = 3$ groups (1 = setosa, 2 = versicolor, 3 = virginica). In the spam data, there are $K = 2$ groups, spam or not spam, $N = 4601$, and $p = 57$ explanatory variables.

A predictor is a function $G(\underline{x})$ that takes values in $\{1, \ldots, K\}$, so that for a new observation $(y^{new}, \underline{x}^{new})$,

$$\widehat{y}^{new} = G(\underline{x}^{New}). \tag{4.2}$$

Usually, this G will also depend on some unknown parameters, in which case the predictions will be based on an estimated function, $\widehat{G}$. (To compare, in linear prediction from the previous two chapters, $G(\underline{x})$ would correspond to $\underline{\beta}'\underline{x}$ and $\widehat{G}(\underline{x})$ would correpsond to $\underline{\widehat{\beta}}'\underline{x}$.)

[1] http://www.ics.uci.edu/∼mlearn/databases/spambase/spambase.DOCUMENTATION

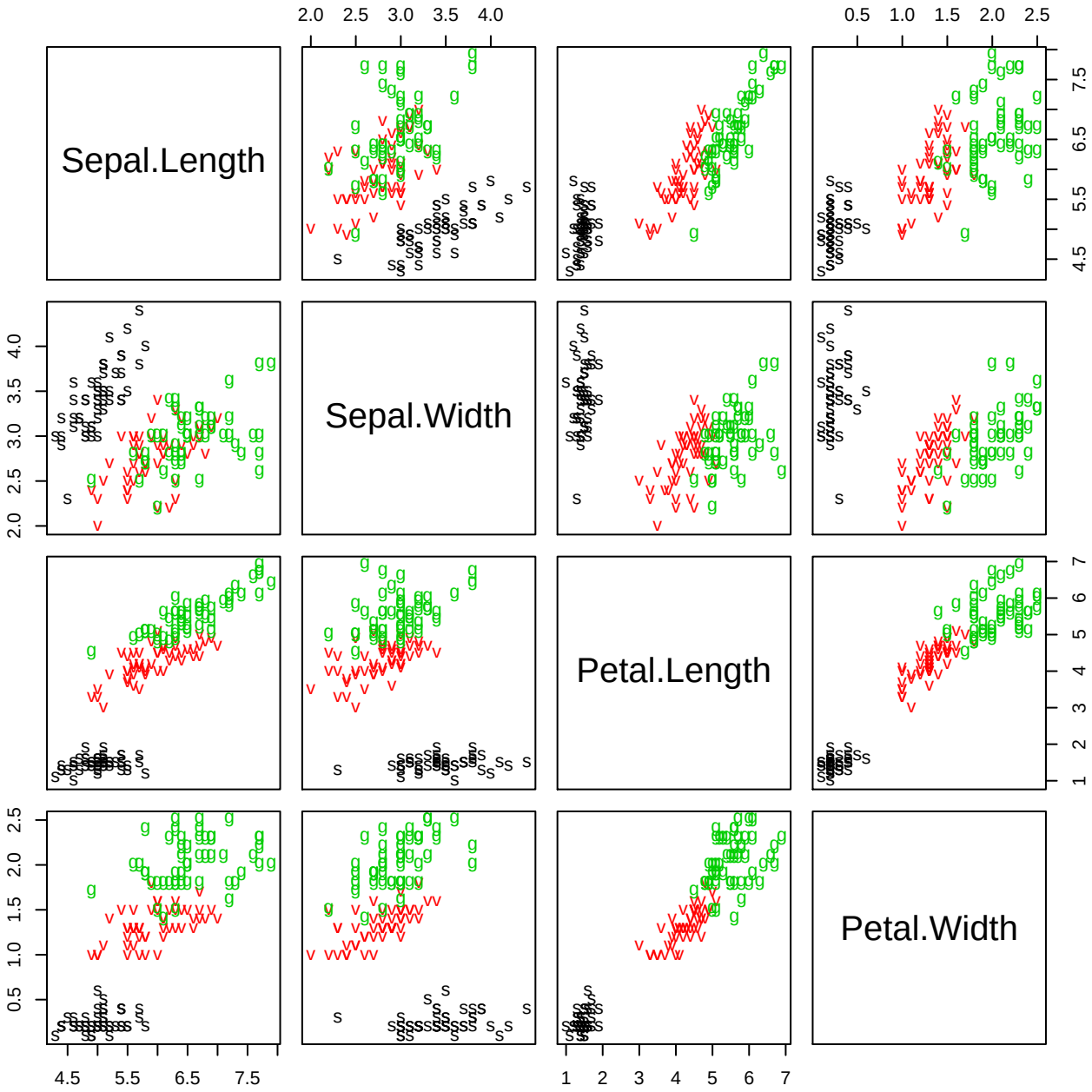


Figure 4.1: Answerson/Fisher Iris Data: s = Setosa, v = Versicolor, g = Virginica

So far, we have considered squared error, $E[\|\underline{Y}^{New} - \underline{\widehat{Y}}^{New}\|^2]$, as the criterion to be minimized, which is not necessarily reasonable for classification, especially when $K > 2$ and the order of the categories is not meaningful, such as for the iris species. There are many possibilities for loss functions, but the most direct is the chance of misclassification. Parallelling the development in Section 2.1, we imagine a set of new observations with the same $\underline{x}_i$'s as the old but new y_i's: $(y_1^{New}, \underline{x}_1), \ldots, (y_N^{New}, \underline{x}_N)$. Then an error occurs if $\widehat{y}_i^{New}$, which equals $\widehat{G}(\underline{x}_i)$, does not equal the true but unobserved y_i^{New}, that is

$$\text{Error}_i = I[Y_i^{New} \neq \widehat{G}(\underline{x}_i)] = \begin{cases} 1 & \text{if} \quad Y_i^{New} \neq \widehat{G}(\underline{x}_i) \\ 0 & \text{if} \quad Y_i^{New} = \widehat{G}(\underline{x}_i) \end{cases} . \tag{4.3}$$

The **in-sample** error is then the average conditional expectations of those errors,

$$\begin{aligned} ERR_{in} &= \frac{1}{N} \sum_{i=1}^{N} E[I[Y_i^{New} \neq \widehat{G}(\underline{x}_i) \mid \underline{X}_i = \underline{x}_i] \\ &= \frac{1}{N} \sum_{i=1}^{N} P[Y_i^{New} \neq \widehat{G}(\underline{x}_i) \mid \underline{X}_i = \underline{x}_i], \end{aligned} \tag{4.4}$$

which is the conditional (on $\underline{x}_i$) probability of misclassification using $\widehat{G}$. (The Y_i^{New}'s are random, as is $\widehat{G}$. There is some ambiguity about whether the randomness in $\widehat{G}$ is due to the uncodnitional distribution of the training set (the $(Y_i, \underline{X}_i)$'s), or the conditional distributions $Y_i|\underline{X}_i = \underline{x}_i$. Either way is reasonable.

A model for the data should specifiy the joint distribution of $(Y, \underline{X})$, up to unknown parameters, One way to specify the model is to condition on y, which amounts to specifying the distribution of $\underline{X}$ within each group k. That is,

$$\begin{aligned} \underline{X} \mid Y = k &\sim f_k(\underline{x}) = f(\underline{x} \mid \underline{\theta}_k) \\ P[Y = k] &= \pi_k. \end{aligned} \tag{4.5}$$

Here, π_k is the population proportion of those in group k, and $f_k(\underline{x})$ is the density of the $\underline{X}$ for group k. The second expression for f_k indicates that the distribution for $\underline{X}$ for each group is from the same family, but each group would have possibly different parameter values, e.g., different means.

If the parameters are known, then it is not difficult to find the best classifier G. Consider

$$P[Y_i^{New} \neq G(\underline{x}_i) \mid \underline{X}_i = \underline{x}_i] = 1 - P[Y_i^{New} = G(\underline{x}_i) \mid \underline{X}_i = \underline{x}_i]. \tag{4.6}$$

The G has values in $\{i, \ldots, K\}$, so to minimize (4.6), we maximize the final probablity over $G(\underline{x}_i)$, which means find the k to maximize

$$P[Y = k \mid \underline{X} = \underline{x}_i]. \tag{4.7}$$

Using Bayes Theorem, we have

$$\begin{aligned} P[Y = k \mid \underline{X} = \underline{x}_i] &= \frac{P[\underline{X} = \underline{x}_i \mid Y = k] \; P[Y = k]}{P[\underline{X} = \underline{x}_i \mid Y = 1] \; P[Y = 1] + \cdots + P[\underline{X} = \underline{x}_i \mid Y = K] \; P[Y = K]} \\ &= \frac{f_k(\underline{x}_i)\pi_k}{f_1(\underline{x}_i)\pi_1 + \cdots + f_K(\underline{x}_i)\pi_K}. \end{aligned} \tag{4.8}$$

Thus for any $\underline{x}$,

$$G(\underline{x}) = k \text{ that maximizes } \frac{f_k(\underline{x})\pi_k}{f_1(\underline{x})\pi_1 + \cdots + f_K(\underline{x})\pi_K}. \tag{4.9}$$

The rest of this chapter considers assumptions on the f_k's and estimation of the parameters.

4.1 The multivariate normal distribution and linear discrimination

Looking at the iris data in Figure 4.1, it is not much of a stretch to see the data within each species being multivariate normal. In this section we take

$$\underline{X} \mid Y = k \sim N_p(\underline{\mu}_k, \mathbf{\Sigma}), \tag{4.10}$$

that is, $\underline{X}$ for group k is p-variate multivariate normal with mean μ_k and covariance matrix $\mathbf{\Sigma}$. (See Section 2.3.) Note that we are assuming different means but the same covariance matrix for the different groups. We wish to find the G in (4.9). The multivariate normal density is

$$f(\underline{x} \mid \underline{\mu}, \mathbf{\Sigma}) = \frac{1}{(\sqrt{2\pi})^p} \; \frac{1}{|\mathbf{\Sigma}|^{1/2}} \; e^{-\frac{1}{2} \; (\underline{x}-\underline{\mu})' \mathbf{\Sigma}^{-1} (\underline{x}-\underline{\mu})}. \tag{4.11}$$

The $|\mathbf{\Sigma}|$ indicates determinant of $\mathbf{\Sigma}$. We are assuming that $\mathbf{\Sigma}$ is invertible.

Look at G in (4.9) with $f_k(\underline{x}) = f(\underline{x} \mid \underline{\mu}_k, \mathbf{\Sigma})$. We want to find a simler expression for defining G. Divide the numerator and denominator by the term for the last group, $f_k(\underline{x})\pi_K$. Then we have that

$$G(\underline{x}) = k \text{ that maximizes } \frac{e^{d_k(\underline{x})}}{e^{d_1(\underline{x})} + \cdots + e^{d_{K-1}(\underline{x})} + 1}, \tag{4.12}$$

where

$$\begin{aligned}
d_k(\underline{x}) &= -\frac{1}{2} \; (\underline{x} - \underline{\mu}_k)' \mathbf{\Sigma}^{-1} (\underline{x} - \underline{\mu}_k) + \log(\pi_k) + \frac{1}{2} \; (\underline{x} - \underline{\mu}_K)' \mathbf{\Sigma}^{-1} (\underline{x} - \underline{\mu}_K) - log(\pi_K) \\
&= -\frac{1}{2} \; (\underline{x}' \mathbf{\Sigma}^{-1} \underline{x} - 2\underline{\mu}_k' \mathbf{\Sigma}^{-1} \underline{x} + \underline{\mu}_k' \mathbf{\Sigma}^{-1} \underline{\mu}_k - \underline{x}' \mathbf{\Sigma}^{-1} \underline{x} + 2\underline{\mu}_K' \mathbf{\Sigma}^{-1} \underline{x} - \underline{\mu}_K' \mathbf{\Sigma}^{-1} \underline{\mu}_K) \\
&\qquad + \log(\pi_k/\pi_K) \\
&= (\underline{\mu}_k - \underline{\mu}_K)' \mathbf{\Sigma}^{-1} \underline{x} - \frac{1}{2} \; (\underline{\mu}_k' \mathbf{\Sigma}^{-1} \underline{\mu}_k - \underline{\mu}_K' \mathbf{\Sigma}^{-1} \underline{\mu}_K) + \log(\pi_k/\pi_K) \\
&= \alpha_k + \underline{\beta}_k' \underline{x},
\end{aligned} \tag{4.13}$$

the new parameters being

$$\alpha_k = -\frac{1}{2} \; (\underline{\mu}_k' \mathbf{\Sigma}^{-1} \underline{\mu}_k - \underline{\mu}_K' \mathbf{\Sigma}^{-1} \underline{\mu}_K) + \log(\pi_k/\pi_K) \;\; \text{and} \;\; \underline{\beta}_k' = (\underline{\mu}_k - \underline{\mu}_K)' \mathbf{\Sigma}^{-1}. \tag{4.14}$$

Since the denominator in (4.12) does not depend on k, we can maximize the ratio by maximizing d_k, that is,

$$G(\underline{x}) = k \text{ that maximizes } d_k(\underline{x}) = \alpha_k + \underline{\beta}_k' \underline{x}. \tag{4.15}$$

(Note that $d_K(\underline{x}) = 0$.) Thus the classifier is based on linear functions of the $\underline{x}$.

This G is fine if the parameters are known, but typically they must be estimated. There are a number of approaches, model-based and otherwise:

1. **Maximum Likelihood Estimate (MLE)**, using the **joint** likelihood of the $(y_i, \underline{x}_i)$'s;
2. **MLE**, using the **conditional** likelihood of the $y_i \mid \underline{x}_i$'s;
3. Minimizing an objective function not tied to the model.

This section takes the first tack. The next section looks at the second. The third approach encompasses a number of methods, to be found in future chapters. As an example, one could find the (possibly non-unique) $(\alpha_k, \underline{\beta}_k)$'s to minimize the number of observed misclassifications.

4.1.1 Finding the joint MLE

So let us look at the joint likelihood. We know for each i, the density is $f_k(\underline{x}_i)\pi_k$ if $y_i = k$. Thus the likelihood for the complete training sample is

$$\begin{aligned} L(\underline{\mu}_1, \ldots, \underline{\mu}_K, \mathbf{\Sigma}, \pi_1, \ldots, \pi_K \quad ; \quad (y_1, \underline{x}_1), \ldots, (y_N, \underline{x}_N)) &= \prod_{i=1}^{N} [f_{y_i}(\underline{x}_i)\pi_{y_i}] \\ &= [\prod_{k=1}^{K} \prod_{y_i=k} \pi_k] \times [\prod_{k=1}^{K} \prod_{y_i=k} f(\underline{x}_i; \underline{\mu}_k, \mathbf{\Sigma})] \\ &= [\prod_{k=1}^{K} \pi_k^{N_k}] \times [\prod_{k=1}^{K} \prod_{y_i=k} f(\underline{x}_i; \underline{\mu}_k, \mathbf{\Sigma})], \end{aligned} \tag{4.16}$$

where $N_k = \#\{y_i = k\}$. Maximizing over the π_k's is straightforward, keeping in mind that they sum to 1, yielding the MLE's

$$\widehat{\pi}_k = \frac{N_k}{N}, \tag{4.17}$$

as for the multinomial. Maximizing the likelihood over the $\underline{\mu}_k$'s (for fixed $\mathbf{\Sigma}$) is equivalent to minimizing

$$-\frac{1}{2} \sum_{y_i=k} (\underline{x}_i - \underline{\mu}_k)' \mathbf{\Sigma}^{-1} (\underline{x}_i - \underline{\mu}_k) \tag{4.18}$$

for each k. Recalling $trace(\mathbf{AB}) = trace(\mathbf{BA})$,

$$\begin{aligned} \sum_{y_i=k} (\underline{x}_i - \underline{\mu}_k)' \mathbf{\Sigma}^{-1} (\underline{x}_i - \underline{\mu}_k) &= \sum_{y_i=k} trace\left(\mathbf{\Sigma}^{-1} (\underline{x}_i - \underline{\mu}_k)(\underline{x}_i - \underline{\mu}_k)' \right) \\ &= trace\left(\mathbf{\Sigma}^{-1} \sum_{y_i=k} (\underline{x}_i - \underline{\mu}_k)(\underline{x}_i - \underline{\mu}_k)' \right). \end{aligned} \tag{4.19}$$

Let $\underline{\overline{x}}_k$ be the sample mean of the $\underline{x}_i$'s in the k^{th} group:

$$\underline{\overline{x}}_k = \frac{1}{N_k} \sum_{y_i=k} \underline{x}_i. \tag{4.20}$$

Then

$$\begin{aligned}
\sum_{y_i=k} (\underline{x}_i - \underline{\mu}_k)(\underline{x}_i - \underline{\mu}_k)' &= \sum_{y_i=k} (\underline{x}_i - \underline{\overline{x}}_k + \underline{\overline{x}}_k - \underline{\mu}_k)(\underline{x}_i - \underline{\overline{x}}_k + \underline{\overline{x}}_k - \underline{\mu}_k)' \\
&= \sum_{y_i=k} (\underline{x}_i - \underline{\overline{x}}_k)(\underline{x}_i - \underline{\overline{x}}_k)' + 2 \sum_{y_i=k} (\underline{x}_i - \underline{\overline{x}}_k)(\underline{\overline{x}}_k - \underline{\mu}_k)' \\
&\qquad + \sum_{y_i=k} (\underline{\overline{x}}_k - \underline{\mu}_k)(\underline{\overline{x}}_k - \underline{\mu}_k)' \\
&= \sum_{y_i=k} (\underline{x}_i - \underline{\overline{x}}_k)(\underline{x}_i - \underline{\overline{x}}_k)' + \sum_{y_i=k} (\underline{\overline{x}}_k - \underline{\mu}_k)(\underline{\overline{x}}_k - \underline{\mu}_k)' \qquad (4.21)
\end{aligned}$$

Putting this equation into (4.19),

$$\begin{aligned}
trace\left(\mathbf{\Sigma}^{-1} \sum_{y_i=k} (\underline{x}_i - \underline{\mu}_k)(\underline{x}_i - \underline{\mu}_k)'\right) &= trace\left(\mathbf{\Sigma}^{-1} \sum_{y_i=k} (\underline{x}_i - \underline{\overline{x}}_k)(\underline{x}_i - \underline{\overline{x}}_k)'\right) \\
&\qquad + \sum_{y_i=k} (\underline{\overline{x}}_k - \underline{\mu}_k)' \mathbf{\Sigma}^{-1} (\underline{\overline{x}}_k - \underline{\mu}_k). \qquad (4.22)
\end{aligned}$$

The $\underline{\mu}_k$ appears on the right-hand side only in the second term. It is easy to see that that term is minimized uniquely by 0, where the minimizer is $\underline{\overline{x}}_k$. We then have that the MLE is

$$\underline{\widehat{\mu}}_k = \underline{\overline{x}}_k. \tag{4.23}$$

Next, use (4.11), (4.22) and (4.23) to show that

$$\begin{aligned}
\log([\prod_{k=1}^{K} \prod_{y_i=k} f(\underline{x}_i; \underline{\widehat{\mu}}_k, \mathbf{\Sigma})]) &= (constant) - \frac{N}{2} \log(|\mathbf{\Sigma}|) \\
&\qquad - \frac{1}{2} \sum_{k=1}^{K} trace\left(\mathbf{\Sigma}^{-1} \sum_{y_i=k} (\underline{x}_i - \underline{\overline{x}}_k)(\underline{x}_i - \underline{\overline{x}}_k)'\right) \\
&= (constant) - \frac{N}{2} \log(|\mathbf{\Sigma}|) - \frac{1}{2} \sum_{k=1}^{K} trace\left(\mathbf{\Sigma}^{-1} \mathbf{S}\right), \qquad (4.24)
\end{aligned}$$

where

$$\mathbf{S} = \sum_{k=1}^{K} \sum_{y_i=k} (\underline{x}_i - \underline{\overline{x}}_k)(\underline{x}_i - \underline{\overline{x}}_k)'. \tag{4.25}$$

It is not obvious, but (4.25) is maximized over $\mathbf{\Sigma}$ by taking $\mathbf{S}/N$, that is, the MLE is

$$\widehat{\mathbf{\Sigma}} = \frac{1}{N} \mathbf{S}. \tag{4.26}$$

See Section 4.1.3. This estimate is the pooled sample covariance matrix. (Although in order for it to be unbiased, one needs to divide by $N - K$ instead of N.)

Because

$$trace(\widehat{\mathbf{\Sigma}}^{-1}S) = trace((S/N)^{-1}S) = N\ trace(\mathbf{I}_p) = Np, \tag{4.27}$$

$$\log([\prod_{k=1}^{K}\prod_{y_i=k} f(\underline{x}_i;\underline{\widehat{\mu}}_k,\widehat{\mathbf{\Sigma}})]) = (constant) - \frac{N}{2}\ \log(|\mathbf{S}/N|) - \frac{1}{2}\ Np. \tag{4.28}$$

Finally, the estimates of the coefficients in (4.14) are found by plugging in the MLE's of the parameters,

$$\widehat{\alpha}_k = -\frac{1}{2}\ (\underline{\widehat{\mu}}_k'\widehat{\mathbf{\Sigma}}^{-1}\underline{\widehat{\mu}}_k - \underline{\widehat{\mu}}_K'\widehat{\mathbf{\Sigma}}^{-1}\underline{\widehat{\mu}}_K) + \log(\widehat{\pi}_k/\widehat{\pi}_K) \ \text{ and } \ \underline{\widehat{\beta}}_k' = (\underline{\widehat{\mu}}_k - \underline{\widehat{\mu}}_K)'\widehat{\mathbf{\Sigma}}^{-1}. \tag{4.29}$$

4.1.2 Using R

The iris data is in the data frame `iris`. You may have to load the `datatsets` package. The first four columns is the $N \times p$ matrix of $\underline{x}_i$'s, $N = 150$, $p = 4$. The fifth column has the species, 50 each of setosa, versicolor, and virginica. The basic variables are then

```
x <- as.matrix(iris[,1:4])
y <- rep(1:3,c(50,50,50))  # gets vector (1,...,1,2,...,2,3,...,3)
K <- 3
N <- 150
p <- 4
```

The mean vectors and pooled covariance matrix are found using

```
m <- NULL
v <- matrix(0,ncol=p,nrow=p)
for(k in 1:K) {
xk <- x[y==k,]
        m <- cbind(m,apply(xk,2,mean))
        v <- v + var(xk)*(nrow(xk)-1) # gets numerator of sample covariance
}
v <- v/N
p <- table(y)/N # This finds the pi-hats.
```

Then `m` is $p \times K$, column k containing $\overline{\underline{x}}_k$.

```
round(m,2)
             [,1] [,2] [,3]
Sepal.Length 5.01 5.94 6.59
Sepal.Width  3.43 2.77 2.97
Petal.Length 1.46 4.26 5.55
Petal.Width  0.25 1.33 2.03
```

```
round(v,3)
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length        0.260       0.091        0.164       0.038
Sepal.Width         0.091       0.113        0.054       0.032
Petal.Length        0.164       0.054        0.181       0.042
Petal.Width         0.038       0.032        0.042       0.041
```

Next, plus these in (4.29).

```
alpha <- NULL
beta <- NULL
vi <- solve(v)  #v inverse
for(k in 1:K) {
      a <- -(1/2)*(m[,k]%*%vi%*%m[,k]-m[,K]%*%vi%*%m[,K])+p[k]-p[K]
      alpha <- c(alpha,a)
      b <- vi%*%(mu[,k]-mu[,K])
      beta <- cbind(beta,b)
}

round(alpha,4)
[1] 18.4284 32.1589  0.0000

round(beta,4)
                  [,1]     [,2] [,3]
Sepal.Length  11.3248   3.3187    0
Sepal.Width   20.3088   3.4564    0
Petal.Length -29.7930  -7.7093    0
Petal.Width  -39.2628 -14.9438    0
```

One actually needs to find only the first $K-1$ coefficients, because the K^{th}'s are 0.

To see how well this classification scheme works on the training set, we first find the $d_k(\underline{x}_i)$'s for each i, then classify each observation according to thier lowest d_k.

```
dd <- NULL
for(k in 1:K) {
      dk <- alpha[k]+x%*%beta[,k]
      dd <- cbind(dd,dk)
}

dd[c(1,51,101),]
          [,1]      [,2] [,3]
1     97.70283  47.39995    0
51   -32.30503   9.29550    0
101 -120.12154 -19.14218    0
```

The last command prints out one observation from each species. We can see that these three are classified correctly, having $k = 1$, 2, and 3, respectively. To find the $\widehat{y}_i$'s for all the observations, use

```
yhat <- apply(dd,1,imax)
```

where `imax` is a little function I wrote to give the index of the largest value in a vector:

```
imax <- function(z) (1:length(z))[z==max(z)]
```

To see how well close the predictions are to the true, use the `table` command:

```
table(yhat,y)
```

```
    y
yhat  1  2  3
   1 50  0  0
   2  0 48  1
   3  0  2 49
```

Thus there were 3 observations misclassified, two versicolors were classified as virginica, and one virginica was classified as versicolor. Not too bad. The observed misclassification rate is

$$\overline{err} = \frac{\#\{\widehat{y}_i \neq y_i\}}{N} = \frac{3}{150} = 0.02. \tag{4.30}$$

Note that this is estimate is likely to be an optimistic (underestimate) of ERR_{in} in (4.4), because it uses the same data to find the classifier and to test it out. There are a number of ways to obtain a better estimate. Here we use cross-validation. The following function calculates the error for leaving observations out. The argument `leftout` is a vector with the indices of the observations you want left out. `varin` is a vector of the indices of the variables you want to use. It outputs the number of errors made in predicting the left out observations. Note this function is for the iris data, with `x` as the $\mathbf{X}$ and `y` and the $\underline{y}$.

```
cviris <- function(leftout,varin) {
    pstar <- length(varin)
    xr <- as.matrix(x[-leftout,varin])
    spr <- sp[-leftout]
    m <- NULL
    v <- matrix(0,pstar,pstar)
    for(i in 1:3) {
       xri <- as.matrix(xr[spr==i,])
       m <- cbind(m,apply(xri,2,mean))
       v <- v+var(xri)*(nrow(xri)-1)
    }
```

```
    vi <- solve(v/nrow(xr))
    dd <- NULL
    for(i in leftout) {
       xn <- x[i,varin]
       d0 <- NULL
       for(j in 1:3) {
          d0 <- c(d0,(xn-m[,j])%*%vi%*%(xn-m[,j]))
       }
       dd<-c(dd,imin(d0))
    }
    sum(dd!=y[leftout])
}
```

The leave-one-out cross-validation, using all four variables, can be found using a loop:

```
err <- NULL
for(i in 1:150) err <- c(err,cviris(i,1:4))
sum(err)/N
[1] 0.02
```

Interestingly, the cv estimate is the same as the observed error, in (4.30). Also, the same observations were misclassified.

4.1.3 Maximizing over $\boldsymbol{\Sigma}$

Lemma 1 *Suppose $a > 0$ and $\mathbf{S} \in \mathcal{S}_q \equiv$ the set of $q \times q$ symmetric positive definite matrices. Then*

$$g(\boldsymbol{\Sigma}) = \frac{1}{|\boldsymbol{\Sigma}|^{a/2}} \, e^{-\frac{1}{2} trace \boldsymbol{\Sigma}^{-1}\mathbf{S}} \tag{4.31}$$

is uniquely maximized over $\boldsymbol{\Sigma} \in \mathcal{S}_q$ by

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{a} \, \mathbf{S}, \tag{4.32}$$

and the maximum is

$$g(\widehat{\boldsymbol{\Sigma}}) = \frac{1}{|\widehat{\boldsymbol{\Sigma}}|^{a/2}} \, e^{-\frac{aq}{2}}. \tag{4.33}$$

Proof. Because $\mathbf{S}$ is positive definite and symmetric, it has an invertible symmetric square root, $\mathbf{S}^{1/2}$. Let $\lambda = \mathbf{S}^{-1/2}\boldsymbol{\Sigma}\mathbf{S}^{-1/2}$, and from (4.31) write

$$g(\boldsymbol{\Sigma}) = h(\mathbf{S}^{-1/2}\boldsymbol{\Sigma}\mathbf{S}^{-1/2}), \quad \text{where} \quad h(\lambda) \equiv \frac{1}{|\mathbf{S}|^{a/2}} \, \frac{1}{|\lambda|^{a/2}} \, e^{-\frac{1}{2} \, trace \lambda^{-1}} \tag{4.34}$$

is a function of $\lambda \in \mathcal{S}_q$. To find the λ that maximizes h, we need only consider the factor without the $\mathbf{S}$, which can be written

$$\frac{1}{|\lambda|^{a/2}}\ e^{-\frac{1}{2}\ trace\lambda^{-1}} = [\prod_{i=1}^{q}\omega_i]^{a/2}\ e^{-\frac{1}{2}\ \sum_{i=1}^{q}\omega_i} = \prod_{i=1}^{q}[\omega_i^{a/2}\ e^{-\frac{1}{2}\ \omega_i}], \tag{4.35}$$

where $\omega_1 \geq \omega_2 \geq \cdots \geq \omega_q > 0$ are the eigenvalues of λ^{-1}. The i^{th} term in the product is easily seen to be maximized over $\omega_i > 0$ by $\widehat{\omega}_i = a$. Because those $\widehat{\omega}_i$'s satisfy the necessary inequalities, the maximizer of (4.35) over λ is $\widehat{\lambda} = (1/a)\mathbf{I}_q$, and

$$h(\widehat{\lambda}) = \frac{a^{a/2}}{|\mathbf{S}|^{a/2}}\ e^{-\frac{1}{2}\ a\cdot trace(\mathbf{I}_q)}, \tag{4.36}$$

from which follows (4.33). Also,

$$\widehat{\lambda} = \mathbf{S}^{-1/2}\widehat{\mathbf{\Sigma}}\mathbf{S}^{-1/2} \Rightarrow \widehat{\mathbf{\Sigma}} = \mathbf{S}^{1/2}\frac{1}{a}\mathbf{I}_q\mathbf{S}^{1/2}, \tag{4.37}$$

which proves (4.32). □

4.2 Quadratic discrimination

The linear discrimination in Section 4.1 developed by assuming the distributions within each group had the same covariance matrix, $\mathbf{\Sigma}$. Here we relax that assumption, but stick with the multivariate normal, so that for group k,

$$\underline{X} \mid Y = k \sim N_p(\underline{\mu}_k, \mathbf{\Sigma}_k), \tag{4.38}$$

The best classifier is similar to the one in (4.15):

$$G(\underline{x}) = k \text{ that maximizes } d_k(\underline{x}), \tag{4.39}$$

where this time

$$d_k(\underline{x}) = -\frac{1}{2}\ (\underline{x} - \underline{\mu}_k)'\mathbf{\Sigma}^{-1}(\underline{x} - \underline{\mu}_k) + \log(\pi_k). \tag{4.40}$$

We could work it so that $d_K(\underline{x}) = 0$, but it does not simplify things much. The procedure is very sensible. It calculates the distance the $\underline{x}$ is from each of the K means, then classifies the y in the closest group, modified a bit by the prior's π_k's. Notice that these disriminant functions are now quadratic rather than linear, hence Fisher termed this approach *quadratic discrimination*.

To implement this classifier, we again need to estimate the parameters. Taking the MLE works as in Section 4.1, but we do not have a pooled covariance estimate. That is, for each k, as in (4.23),

$$\widehat{\underline{\mu}}_k = \overline{\underline{x}}_k, \tag{4.41}$$

and

$$\widehat{\mathbf{\Sigma}}_k = \frac{1}{N_k} \sum_{y_i=k} (\underline{x}_i - \overline{\underline{x}}_k)(\underline{x}_i - \overline{\underline{x}}_k)'. \tag{4.42}$$

The MLE of π_k is again N_k/N, as in (4.17). Note that the we can write the pooled estimate in (4.26) as

$$\widehat{\mathbf{\Sigma}} = \widehat{\pi}_1 \widehat{\mathbf{\Sigma}}_1 + \cdots + \widehat{\pi}_N \widehat{\mathbf{\Sigma}}_N. \tag{4.43}$$

4.2.1 Using R

We again use the iris data. The only difference here from Section 4.1.2 is that we estimate three separate covariance matrices. Using the same setup as in that section, we first estimate the parameters:

```
m <- NULL
v <- vector("list",K) # To hold the covariance matrices
for(k in 1:K) {
xk <- x[y==k,]
        m <- cbind(m,apply(xk,2,mean))
        v[[k]] <- var(xk)*(nrow(xk)-1)/nrow(xk)
}
p <- table(y)/N
```

Next, we find the estimated d_k's from (4.40):

```
dd <- NULL
for(k in 1:K) {
      dk <- apply(x,1,function(xi) -(1/2)*
             (xi-m[,k])%*%solve(v[[k]],xi-m[,k])+log(p[k]))
      dd <- cbind(dd,dk)
}
yhat <- apply(dd,1,imax)
table(yhat,y)
```

```
    y
yhat  1  2  3
   1 50  0  0
   2  0 47  0
   3  0  3 50
```

Three observations were misclassified again, $\overline{err} = 3/150 = 0.02$. Leave-one-out cross-validation came up with an estimate of $4/150 = 0.0267$, which is slightly worse than that for linear discrimination. It does not appear that the extra complication of having three covariance matrices improves the classification rate, but see Section 4.3.2.

4.3 The Akaike Information Criterion (AIC)

In almost all areas of statistical inference, one is confronted with choosing between a variety of models. E.g., in subset regression, we are choosing between the 2^p possible linear models based on the subsets of the x-variables. Similar considerations arise in factor analysis, time series, loglinear models, clustering, etc. In most cases, the task is to balance fit and complexity, that is, find a model that fits well but also has a small number of parameters. The C_p criterion in (2.59) and (2.60) is one possibility, at least for linear models with the least squares loss function. This criterion is based on the squared error loss in predicting a new observation, $(\widehat{y}^{New} - y^{New})^2$.

Consider a general model, one not necessarily having to do with linear models or classification. Let the data be $\underline{W}_1, \ldots, \underline{W}_N$ iid with density $f(\underline{w} \mid \underline{\theta})$. In our models, the $\underline{w}$ would be the pair $(\underline{y}, \underline{x})$. The goal is to predict a new observation $\underline{W}^{New}$ that is independent of the other $\underline{W}_i$'s, but has the same distribution. (We are not given and part of this new observation, unlike the predictions we have been consdiering so far in which $\underline{x}^{New}$ is given.) The prediction is the likeliest value of $\underline{W}^{New}$, which is the value that maximizes the estimated density:

$$\underline{\widehat{w}}^{New} = \underline{w} \text{ that maximizes } f(\underline{w} \mid \underline{\widehat{\theta}}), \tag{4.44}$$

where $\underline{\widehat{\theta}}$ is the MLE of $\underline{\theta}$ based upon the data. The actual value $\underline{w}^{New}$ will be different that the predictor, but we hope it is "close," where by close we mean "has a high likelihood," rather than meaning close in a Euclidean distance sense. Thus our utility is a function of $f(\mathbf{w}^{New} \mid \underline{\widehat{\theta}})$. Turning this utility into a loss, the loss function is decreasing in the likelihood. Specifically, we take

$$Loss(\underline{w}^{New}) = -2 \log \left(f(\underline{w}^{New} \mid \underline{\widehat{\theta}}) \right). \tag{4.45}$$

The expected value of the loss is what we wish tominimize, so it takes the role of ERR:

$$ERR = -2 \; E \left[\log \left(f(\underline{W}^{New} \mid \underline{\widehat{\theta}}) \right) \right]. \tag{4.46}$$

The expected value is over $\underline{W}^{New}$ and the data $\underline{W}_1, \ldots, \underline{W}_N$, the latter through the randomness of $\underline{\widehat{\theta}}$, assumiong that the true model is $f(\cdot \mid \underline{\theta})$.

The observed analog of ERR plugs in the $\underline{w}_i$'s for $\underline{w}^{New}$, then averages:

$$\overline{err} = -\frac{2}{N} \sum_{i=1}^{N} \log \left(f(\underline{w}_i \mid \underline{\widehat{\theta}}) \right). \tag{4.47}$$

As in (2.54) for subset selection in linear models, we expect $\overline{err}$ to be an underestimate of ERR since the former assesses the prediction of the same observations used to estimate $\underline{\theta}$. Thus we would like to adjust $\overline{err}$ up a little. We try to find, at least approximately,

$$\Delta = ERR - E[\overline{err}], \tag{4.48}$$

so that $\overline{err} + \Delta$ would be a better estimate of ERR.

We will sketch the calculations for a regular q-dimensional exponential family. That is, we assume that f has density

$$f(\underline{w} \mid \underline{\theta}) = a(\underline{w}) e^{\underline{\theta}' \underline{T}(\underline{w}) - \psi(\underline{\theta})}, \tag{4.49}$$

where $\underline{\theta}$ is the $q \times 1$ *natural parameter*, $\underline{T}$ is the $q \times 1$ *natural sufficient statistic*, and $\psi(\underline{\theta})$ is the normalizing constant.

The mean vector and covariance matrix of the $\underline{T}$ can be found by taking derivatives of the ψ:

$$\underline{\mu}(\underline{\theta}) = E_{\underline{\theta}}[\underline{T}(\underline{W})] = \begin{pmatrix} \frac{\partial}{\partial \theta_1} \psi(\underline{\theta}) \\ \frac{\partial}{\partial \theta_2} \psi(\underline{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_q} \psi(\underline{\theta}) \end{pmatrix} \tag{4.50}$$

and

$$\begin{aligned} \mathbf{\Sigma}(\underline{\theta}) = Cov_{\underline{\theta}}[\underline{T}(\underline{W})] &= \begin{pmatrix} \frac{\partial^2}{\partial \theta_1^2} \psi(\underline{\theta}) & \frac{\partial^2}{\partial \theta_1 \partial \theta_2} \psi(\underline{\theta}) & \cdots & \frac{\partial^2}{\partial \theta_1 \partial \theta_q} \psi(\underline{\theta}) \\ \frac{\partial^2}{\partial \theta_2 \partial \theta_1} \psi(\underline{\theta}) & \frac{\partial^2}{\partial \theta_2^2} \psi(\underline{\theta}) & \cdots & \frac{\partial^2}{\partial \theta_2 \partial \theta_q} \psi(\underline{\theta}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial \theta_q \partial \theta_1} \psi(\underline{\theta}) & \frac{\partial^2}{\partial \theta_q \partial \theta_2} \psi(\underline{\theta}) & \cdots & \frac{\partial^2}{\partial \theta_q^2} \psi(\underline{\theta}) \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial}{\partial \theta_1} \mu_1(\underline{\theta}) & \frac{\partial}{\partial \theta_1} \mu_2(\underline{\theta}) & \cdots & \frac{\partial}{\partial \theta_1} \mu_q(\underline{\theta}) \\ \frac{\partial}{\partial \theta_2} \mu_1(\underline{\theta}) & \frac{\partial}{\partial \theta_2} \mu_2(\underline{\theta}) & \cdots & \frac{\partial}{\partial \theta_2} \mu_q(\underline{\theta}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial \theta_q} \mu_1(\underline{\theta}) & \frac{\partial}{\partial \theta_q} \mu_2(\underline{\theta}) & \cdots & \frac{\partial}{\partial \theta_q} \mu_q(\underline{\theta}) \end{pmatrix}. \end{aligned} \tag{4.51}$$

Turning to the likelihoods,

$$-2 \log\left(f(\underline{w} \mid \underline{\theta})\right) = -2\underline{\theta}' \underline{T}(\underline{w}) + 2\psi(\underline{\theta}) - 2\log(a(\underline{w})), \tag{4.52}$$

hence

$$\begin{aligned} -2 \log\left(f(\underline{w}^{New} \mid \widehat{\underline{\theta}})\right) &= -2\widehat{\underline{\theta}}' \underline{T}(\underline{w}^{New}) + 2\psi(\widehat{\underline{\theta}}) - 2\log(a(\underline{w}^{New})); \\ -\frac{2}{N} \sum_{i=1}^{N} \log\left(f(\underline{w}_i \mid \widehat{\underline{\theta}})\right) &= -2\widehat{\underline{\theta}}' \overline{\underline{t}} + 2\psi(\widehat{\underline{\theta}}) - \frac{2}{N} \sum_{i=1}^{N} \log(a(\underline{w}_i)), \end{aligned} \tag{4.53}$$

where $\overline{\underline{t}}$ is the sample mean of the $\underline{T}(\underline{w}_i)$'s,

$$\overline{\underline{t}} = \frac{1}{N} \sum_{i=1}^{N} T(\underline{w}_i). \tag{4.54}$$

From (4.46),

$$\begin{aligned} ERR &= -2\ E\left[\log\left(f(\underline{w}^{New} \mid \widehat{\underline{\theta}})\right)\right] \\ &= -2E[\widehat{\underline{\theta}}]' E[\underline{T}(\underline{W}^{New})] + 2E[\psi(\widehat{\underline{\theta}})] - 2E[\log(a(\underline{W}^{New}))] \\ &= -2E[\widehat{\underline{\theta}}]' \underline{\mu}(\underline{\theta}) + 2E[\psi(\widehat{\underline{\theta}})] - 2E[\log(a(\underline{W}))], \end{aligned} \tag{4.55}$$

because $\underline{W}^{New}$ and $\widehat{\underline{\theta}}$ are independent, and $\underline{\mu}$ is the mean of $\underline{T}$. From (4.47)

$$\begin{aligned} E[\overline{err}] &= -\frac{2}{N}\sum_{i=1}^{N} E[\log\left(f(\underline{w}_i \mid \widehat{\underline{\theta}})\right)] \\ &= -2E[\widehat{\underline{\theta}}'\overline{\underline{T}}] + 2E[\psi(\widehat{\underline{\theta}})] - 2\ E[\log(a(\underline{W}))], \end{aligned} \tag{4.56}$$

because all the $a(\underline{W}_i)$'s have the same expected value. Thus from (4.48),

$$\Delta = -2\left(E[\widehat{\underline{\theta}}]'\underline{\mu}(\underline{\theta}) - E[\widehat{\underline{\theta}}'\overline{\underline{T}}]\right) = 2\left(E\left[\widehat{\underline{\theta}}'(\overline{\underline{T}} - \underline{\mu}(\underline{\theta}))\right]\right). \tag{4.57}$$

Because $E[\overline{\underline{T}}] = \underline{\mu}(\underline{\theta})$, $E[\underline{\theta}'(\overline{\underline{T}} - \underline{\mu}(\underline{\theta})) = 0$, so we can add that in, to obtain

$$\Delta = 2\left(E\left[(\widehat{\underline{\theta}} - \underline{\theta})'(\overline{\underline{T}} - \underline{\mu}(\underline{\theta}))\right]\right). \tag{4.58}$$

Now the MLE of $\widehat{\underline{\theta}}$ has the property that the theoretical mean of $\underline{T}$ at the MLE equals the sample mean:

$$\underline{\mu}(\widehat{\underline{\theta}}) = \overline{\underline{t}}. \tag{4.59}$$

Expanding the $\underline{\mu}$ in a Taylor Series about $\widehat{\underline{\theta}} = \underline{\theta}$, since the $\mathbf{\Sigma}$ cotains the derivatives of $\underline{\mu}$ as in (4.51),

$$\underline{\mu}(\widehat{\underline{\theta}}) = \underline{\mu}(\underline{\theta}) + \mathbf{\Sigma}(\underline{\theta}^*)(\widehat{\underline{\theta}} - \underline{\theta}), \tag{4.60}$$

where $\underline{\theta}^*$ is between $\underline{\theta}$ and $\widehat{\underline{\theta}}$. By (4.59), (4.60) can be manipulated to

$$\widehat{\underline{\theta}} - \underline{\theta} = \mathbf{\Sigma}^{-1}(\underline{\theta}^*)(\overline{\underline{t}} - \underline{\mu}(\underline{\theta})) \approx \mathbf{\Sigma}^{-1}(\underline{\theta})(\overline{\underline{t}} - \underline{\mu}(\underline{\theta})). \tag{4.61}$$

The last step sees us approximating $\underline{\theta}^*$ by $\underline{\theta}$, which can be justified for large N. Inserting (4.61) into (4.57), we have

$$\begin{aligned} \Delta &\approx 2\left(E\left[(\overline{\underline{T}} - \underline{\mu}(\underline{\theta}))'\mathbf{\Sigma}^{-1}(\underline{\theta})(\overline{\underline{T}} - \underline{\mu}(\underline{\theta}))\right]\right) \\ &= 2\left(E\left[trace(\mathbf{\Sigma}^{-1}(\underline{\theta})(\overline{\underline{T}} - \underline{\mu}(\underline{\theta}))(\overline{\underline{T}} - \underline{\mu}(\underline{\theta}))')\right]\right) \\ &= 2\ trace\left(\mathbf{\Sigma}^{-1}(\underline{\theta})\left(E\left[(\overline{\underline{T}} - \underline{\mu}(\underline{\theta}))(\overline{\underline{T}} - \underline{\mu}(\underline{\theta}))'\right]\right)\right) \\ &= 2\ trace\left(\mathbf{\Sigma}^{-1}(\underline{\theta})\ Cov[\overline{\underline{T}}]\right) \\ &= 2\ trace\left(\mathbf{\Sigma}^{-1}(\underline{\theta})\ (\mathbf{\Sigma}(\underline{\theta})/N)\right) \\ &= 2\ trace(\mathbf{I}_q)/N \\ &= 2\ \frac{q}{N}. \end{aligned} \tag{4.62}$$

Finally, return to (4.48). With $\Delta \approx 2q/N$, we have that

$$AIC \equiv \widehat{ERR} = \overline{err} + 2\ \frac{q}{N} \tag{4.63}$$

is an approximately unbiased estimate of ERR. It is very close to the C_p statistic in (2.59), but does not have the $\widehat{\sigma}_e^2$. As in (2.60), the AIC balances observed error and number of parameters. This criterion is very useful in any situation where a number of models are being considered. One calculates the AIC's for the models, then chooses the one with lowest AIC, or at least not much largerthan the lowest.

4.3.1 Bayes Information Criterion (BIC)

An alternative criterion arises when using a Bayesian formulation in which each model consists of the density of $\underline{W} \mid \underline{\theta}$, which is $f(\underline{w} \mid \underline{\theta})$, plus a prior density $\rho(\underline{\theta})$ for the parameter. Instead of trying to estimate the likelihood, we try to estimate the marginal likelihood of the training sample:

$$f(\underline{w}_1, \ldots, \underline{w}_N) = \int f(\underline{w}_1, \ldots, \underline{w}_N \mid \underline{\theta}) \rho(\underline{\theta}) d\underline{\theta}. \tag{4.64}$$

I won't go into the derivation, but the estimate of $-2\log(f)$ is

$$BIC = -2 \ \log(\widehat{f}(\underline{w}_1, \ldots, \underline{w}_N)) = N \ \overline{err} + \log(N) \ q, \tag{4.65}$$

where $\overline{err}$ is the same as before, in (4.47). The prior ρ does not show up in the end. It is assumed N is large enough that the prior is relatively uniformative.

Note that the BIC/N is the same as the AIC, but with $\log(N)$ instead of the "2." Thus for large N, the BIC tends to choose simpler models than the AIC. An advantage of the BIC is that we can use it to estimate the posterior probability of the models. That is, suppose we have M models, each with its own density, set of parameters, and prior, $(f_m(\underline{w}_1 \ldots, \underline{w}_N \mid \underline{\theta}_m), \rho_m(\underline{\theta}_m))$. There is also Π_m, the prior probability that model m is the true one. (So that ρ_m is the conditional density on $\underline{\theta}_m$ given that model m is the true one.) Then the distribution of the data given model m is

$$f(Data \mid Model = m) = f_m(\underline{w}_1 \ldots, \underline{w}_N) = \int f_m(\underline{w}_1 \ldots, \underline{w}_N \mid \underline{\theta}_m) \rho_m(\underline{\theta}_m) d\underline{\theta}_m, \tag{4.66}$$

and the probability that model m is the true one given the data is

$$P[Model = m \mid Data] = \frac{f_m(\underline{w}_1 \ldots, \underline{w}_N)\Pi_m}{f_1(\underline{w}_1 \ldots, \underline{w}_N)\Pi_1 + \cdots + f_M(\underline{w}_1 \ldots, \underline{w}_N)\Pi_M}. \tag{4.67}$$

Using the estimate in (4.65), where BIC_m is that for model m, we can estimate the posterior probabilities,

$$\widehat{P}[Model = m \mid Data] = \frac{e^{-\frac{1}{2} \ BIC_m}\Pi_m}{e^{-\frac{1}{2} \ BIC_1}\Pi_1 + \cdots + e^{-\frac{1}{2} \ BIC_M}\Pi_M}. \tag{4.68}$$

If, as often is done, one assumes that the prior probablities for the models are the same $1/M$, then (4.68) simplifies even more by dropping the Π_m's.

4.3.2 Example: Iris data

In Sections 4.1 and 4.2 we considered two models for the iris data based on whether the covariances are equal. Here we use AIC and BIC to compare the models. For each model,

we need to find the likelihood, and figure out q, the number of free parameters. When using AIC or BIC, we need the joint likelihood of the $\underline{w}_i = (y_i, \underline{x}_i)$'s:

$$f(y_i, \underline{x}_i \mid \underline{\theta}) = f(\underline{x}_i \mid y_i, \underline{\theta})\pi_{y_i}, \tag{4.69}$$

where the $\underline{\theta}$ contains the $\underline{\mu}_k$'s, $\mathbf{\Sigma}_k$'s (or $\mathbf{\Sigma}$), and π_k's. The conditional distribution of the $\underline{X}_i$'s given the Y_i's are multivariate normal as in (4.38) (see (4.11)), hence

$$\begin{aligned} -2\log(f(y_i, \underline{x}_i \mid \underline{\theta})) &= -2\log(f(\underline{x} \mid y_i, \underline{\theta})) - 2\log(\pi_{y_i}) \\ &= -2\log(f(\underline{x} \mid \mu_k, \mathbf{\Sigma}_k)) - 2\log(\pi_k) \quad \text{if } \; y_i = k \\ &= \log(|\mathbf{\Sigma}_k|) + (\underline{x}_i - \underline{\mu}_k)'\mathbf{\Sigma}_k^{-1}(\underline{x}_i - \underline{\mu}_k) - 2\log(\pi_k) \quad \text{if } \; y_i = k \end{aligned} \tag{4.70}$$

Averaging (4.70) over the observations, and inserting the MLE's, yields the $\overline{err}$ for the model with different covariances:

$$\begin{aligned} \overline{err}_{Diff} &= \frac{21}{N}\sum_{k=1}^{K}\sum_{y_i=k}\Big(\log(|\widehat{\mathbf{\Sigma}}_k|) + (\underline{x}_i - \overline{\underline{x}}_k)'\widehat{\mathbf{\Sigma}}_k^{-1}(\underline{x}_i - \overline{\underline{x}}_k) - 2\log(\widehat{\pi}_k)\Big) \\ &= \frac{1}{N}\sum_{k=1}^{3} N_k \log(|\widehat{\mathbf{\Sigma}}_k|) + \frac{1}{N}\sum_{k=1}^{K}\sum_{y_i=k}\Big((\underline{x}_i - \overline{\underline{x}}_k)'\widehat{\mathbf{\Sigma}}_k^{-1}(\underline{x}_i - \overline{\underline{x}}_k)\Big) - \frac{2}{N}\sum_{k=1}^{K} N_k \log(\widehat{\pi}_k) \\ &= \sum_{k=1}^{K}\widehat{\pi}_k \log(|\widehat{\mathbf{\Sigma}}_k|) + \frac{1}{N}\sum_{k=1}^{K}\sum_{y_i=k} trace\left(\widehat{\mathbf{\Sigma}}_k^{-1}(\underline{x}_i - \overline{\underline{x}}_k)(\underline{x}_i - \overline{\underline{x}}_k)'\right) - 2\sum_{k=1}^{K}\widehat{\pi}_k\log(\widehat{\pi}_k) \\ &= \sum_{k=1}^{K}\widehat{\pi}_k \log(|\widehat{\mathbf{\Sigma}}_k|) + \frac{1}{N}\sum_{k=1}^{K} trace\left(\widehat{\mathbf{\Sigma}}_k^{-1}\sum_{y_i=k}(\underline{x}_i - \overline{\underline{x}}_k)(\underline{x}_i - \overline{\underline{x}}_k)'\right) - 2\sum_{k=1}^{K}\widehat{\pi}_k\log(\widehat{\pi}_k) \\ &= \sum_{k=1}^{K}\widehat{\pi}_k \log(|\widehat{\mathbf{\Sigma}}_k|) + \frac{1}{N}\sum_{k=1}^{K} trace\left(\widehat{\mathbf{\Sigma}}_k^{-1}N_k\widehat{\mathbf{\Sigma}}_k\right) - 2\sum_{k=1}^{K}\widehat{\pi}_k\log(\widehat{\pi}_k) \quad \text{(Eqn. 4.15)} \\ &= \sum_{k=1}^{K}\widehat{\pi}_k \log(|\widehat{\mathbf{\Sigma}}_k|) + \frac{1}{N}\sum_{k=1}^{K} N_k p - 2\sum_{k=1}^{K}\widehat{\pi}_k\log(\widehat{\pi}_k) \quad (\widehat{\mathbf{\Sigma}}_k \text{ is } p\times p) \\ &= \sum_{k=1}^{K}\widehat{\pi}_k \log(|\widehat{\mathbf{\Sigma}}_k|) + p - 2\sum_{k=1}^{K}\widehat{\pi}_k\log(\widehat{\pi}_k) \end{aligned} \tag{4.71}$$

Under the model (4.10) that the covariance matrices are equal, the calculations are very similar but with $\widehat{\mathbf{\Sigma}}$ in place of the three $\widehat{\mathbf{\Sigma}}_k$'s. The result is

$$\overline{err}_{Same} = \log(|\widehat{\mathbf{\Sigma}}|) + p - 2\sum_{k=1}^{K}\widehat{\pi}_k\log(\widehat{\pi}_k) \tag{4.72}$$

The numbers of free parameters q for the two models are counted next (recall $K = 3$ and

$p = 4$):

Model	Covariance parameters	Mean parameters	$\pi_k's$	Total
Same covariance	$p(p+1)/2 = 10$	$K \cdot p = 12$	$K - 1 = 2$	$q = 24$
Different covariances	$K \cdot p(p+1)/2 = 30$	$K \cdot p = 12$	$K - 1 = 2$	$q = 44$

(4.73)

There are only $K - 1$ free π_k's because they sum to 1.

To find the AIC and BIC, we first obtain the log determinants:

$$\begin{aligned} \log(|\widehat{\mathbf{\Sigma}}_1|) &= -13.14817 \\ \log(|\widehat{\mathbf{\Sigma}}_2|) &= -10.95514 \\ \log(|\widehat{\mathbf{\Sigma}}_3|) &= -9.00787 \\ \log(|\widehat{\mathbf{\Sigma}}|) &= -10.03935. \end{aligned} \tag{4.74}$$

For both cases, $p = 4$ and $-2\ \sum_{k=1}^{K} \widehat{\pi}_k \log(\widehat{\pi}_k) = 2\log(3) = 2.197225$. Then

$$\begin{aligned} \overline{err}_{Diff} &= \frac{-13.14817 - 10.95514 - 9.00787}{3} + 4 + 2.197225 = -4.839835, \\ \overline{err}_{Same} &= -10.03935 + 4 + 2.197225 = -3.842125. \end{aligned} \tag{4.75}$$

Thus the model with different covariance matrices has an average observed error about 1 better than the model with the same covariance. Next we add in the penalties for the AIC (4.63) and BIC (4.65), where $N = 150$:

	AIC	BIC/N
Different covariances	$-4.839835 + 2\ \frac{44}{150} = -4.25$	$-4.839835 + \log(150)\ \frac{44}{150} = -3.37$
Same covariance	$-3.842125 + 2\ \frac{24}{150} = -3.52$	$-3.842125 + \log(150)\ \frac{24}{150} = -3.04$

(4.76)

Even with the penalty added, the model with different covariances is chosen over the one with the same covariance by both AIC and BIC, although for BIC there is not as large of a difference. Using (4.68), we can estimate the posterior odds for the two models:

$$\frac{P[Diff \mid Data]}{P[Same \mid Data]} = \frac{e^{-\frac{1}{2}\ BIC_{Diff}}}{e^{-\frac{1}{2}\ BIC_{Same}}} = \frac{e^{-\frac{1}{2}\ 150\times(-3.37)}}{e^{-\frac{1}{2}\ 150\times(-3.04)}} \approx e^{25}. \tag{4.77}$$

Thus the model with different covariances is close to infinitely more probable.

In the *R* Sections 4.1.2 and 4.2.1, cross-validation barely chose the simpler model over that with three covariances, 0.02 versus 0.0267. Thus there seems to be a conflict between AIC/BIC and cross-validation. The conflict can be explained by noting that AIC/BIC are trying to model the $\underline{x}_i$'s and y_i's jointly, while cross-validation tries to model the conditional distribution of the y_i's given the $\underline{x}_i$'s. The latter does not really care about the distribution of the $\underline{x}_i$'s, except to the extent it helps in predicting the y_i's.

Looking at (4.74), it appears that the difference in the models is mainly due to the first covariance being high. The first group is the setosa species, which is easy to distinguish from the other two species. Consider the two models without the setosas. Then we have

	AIC	BIC/N
Different covariances	-4.00	-3.21
Same covariance	-3.82	-3.30

(4.78)

Here, AIC chooses the different covariances, and BIC chooses the same. The posterior odds here are

$$\frac{P[Diff \mid Data]}{P[Same \mid Data]} = \frac{e^{-\frac{1}{2}\ BIC_{Diff}}}{e^{-\frac{1}{2}\ BIC_{Same}}} = \frac{e^{-\frac{1}{2}\ 100\times(-3.21)}}{e^{-\frac{1}{2}\ 100\times(-3.30)}} \approx 0.013. \tag{4.79}$$

Thus the simpler model has estimated posterior probability of about 98.7%, suggesting strongly that whereas the covariance for the setosas is different than that for the other two, versicolor and virginica's covariance matrices can be taken to be equal.

4.3.3 Hypothesis testing

Although not directly aimed at classification, the likelihood ratio test for testing the hypothesis that the covariances are equal is based on the statistic

$$2\log(\Lambda) \equiv N(\overline{err}_{Same} - \overline{err}_{Diff}). \tag{4.80}$$

Under the null hypothesis, as $N \to \infty$,

$$2\log(\Lambda) \longrightarrow \chi^2_{q_{Diff}-q_{Same}}. \tag{4.81}$$

When testing all three species, from (4.75),

$$2\log(\Lambda) = 150 \times (-3.842125 + 4.839835) = 149.66. \tag{4.82}$$

the degrees of freedom for the chi-square are $44 - 24 = 20$, so we stribgly reject the null hypothesis that the three covariance matrices are equal.

If we look at just virginica and versicolor, we find

$$2\log(\Lambda) = 100 \times (-4.221285 + 4.595208) = 37.39 \tag{4.83}$$

on 10 degrees of freedom, which is also significant.

4.4 Other exponential familes

The multivariate normal is an exponential family as in (4.49). From (4.11), the natural sufficient consists of all the x_j's and all the products $x_j x_k$. E.g., for $p = 3$,

$$T(\underline{x}) = (x_1, x_2, x_3, x_1^2, x_1x_2, x_1x_3, x_2^2, x_2x_3, x_3^2)'. \tag{4.84}$$

The natural parameter $\underline{\theta}$ is a rather unnatural function of the $\underline{\mu}$ and $\mathbf{\Sigma}$. Other exponential families have other statistics and parameters. More generally, suppose that the conditional distribution of $\underline{X}_i$ given Y_i is an exponential family distirbution:

$$\underline{X}_i \mid Y_i = k \sim f(\underline{x}_i \mid \underline{\theta}_k), \tag{4.85}$$

where

$$f(\underline{x} \mid \underline{\theta}) = a(\underline{x})e^{\underline{\theta}'\underline{T}(\underline{x})-\psi(\underline{\theta})}. \tag{4.86}$$

The best classifier is again (4.9)

$$\begin{aligned} G(\underline{x}) &= k \text{ that maximizes } \frac{f_k(\underline{x})\pi_k}{f_1(\underline{x})\pi_1 + \cdots + f_K(\underline{x})\pi_K} \\ &= k \text{ that maximizes } \frac{e^{d_k(\underline{x})}}{e^{d_1(\underline{x})} + \cdots + e^{d_{K-1}(\underline{x})} + 1}, \end{aligned} \tag{4.87}$$

where now

$$d_k(\underline{x}) = (\underline{\theta}_k'\underline{T}(\underline{x}) - \psi(\underline{\theta}_k) + \log(\pi_k)) - (\underline{\theta}_K'\underline{T}(\underline{x}) - \psi(\underline{\theta}_K) + \log(\pi_K)). \tag{4.88}$$

(The $a(\underline{x})$ cancels.) These d_k's are like (4.15), linear functions in the $\underline{T}$. The classifier can therefore be written

$$G(\underline{x}) = k \text{ that maximizes } d_k(\underline{x}) = \alpha_k + \underline{\beta}_k'\underline{T}(\underline{x}), \tag{4.89}$$

where

$$\alpha_k = -\psi(\underline{\theta}_k) + \log(\pi_k) - \psi(\underline{\theta}_K) + \log(\pi_K) \text{ and } \underline{\beta}_k = \underline{\theta}_k - \underline{\theta}_K. \tag{4.90}$$

To implement the procedure, we have to estimate the α_k's and $\underline{\beta}_k$'s, which is not difficult once we have the estimates of the $\underline{\theta}_k$'s and π_k's. These parameters can be estimated using maximum likelihood, where as before, $\widehat{\pi}_k = N_k/N$. This approach depends on knowing the f in (4.86). The next section we show how to finesse this estimation.

4.5 Conditioning on X: Logistic regression

Suppose the exponential family assumption (4.85) holds for $\underline{X}$ given $Y = k$. Then, as from (4.8) and (4.9), the best classifier chooses the k to maximize

$$P[Y = k \mid \underline{X} = \underline{x}] = p(k \mid \underline{x}) = \frac{e^{d_k(\underline{x})}}{e^{d_1(\underline{x})} + \cdots + e^{d_{K-1}(\underline{x})} + 1} \tag{4.91}$$

for the d_k's in (4.89). In this section, instead of estimating the parameters using maximum likelihood of the $(y_i, \underline{x}_i)$'s, we will estimate the $(\alpha_k, \underline{\beta}_k)$'s using *conditional* maximum likelihood, conditioning on the $\underline{x}_i$'s. The conditional likelihood for y_i is $P[Y_i = y_i \mid \underline{X}_i = \underline{x}_i]$ from (4.91), so that for the entire training sample is

$$L((\alpha_1, \underline{\beta}_1), \ldots, (\alpha_{K-1}, \underline{\beta}_{K-1}) \mid Data] = \prod_{i=1}^{N} p(y_i \mid \underline{x}_i). \tag{4.92}$$

Bibliography

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

George M. Furnival and Jr Wilson, Robert W. Regression by leaps and bounds. *Technometrics*, 16:499–511, 1974.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

Guy P. Nason. Wavelet shrinkage by cross-validation. *Journal of the Royal Statistical Society B*, 58:463–479, 1996. URL `citeseer.ist.psu.edu/nason96wavelet.html`.