Exploiting the Linear Model Framework

DAAG Chapter 7
Section 7.1: Factors

- Regression with "nominal" predictors:

```r
> sugar
  weight trt
 1  82.0 Control
 2  97.8 Control
 3  69.9 Control
 4  58.3    A
 5  67.9    A
 6  59.3    A
 7  68.1    B
 8  70.8    B
 9  63.6    B
10 50.7    C
11 47.1    C
12 48.9    C

> sugar2
   y control A B C
 1  82.0     1 0 0 0
 2  97.8     1 0 0 0
 3  69.9     1 0 0 0
 4  58.3     0 1 0 0
 5  67.9     0 1 0 0
 6  59.3     0 1 0 0
 7  68.1     0 0 1 0
 8  70.8     0 0 1 0
 9  63.6     0 0 1 0
10 50.7     0 0 0 1
11 47.1     0 0 0 1
12 48.9     0 0 0 1
```

___ represent as???
> summary(lm(y~control+A+B+C,data=sugar2))

Call:
lm(formula = y ~ control + A + B + C, data = sugar2)

Residuals:
      Min       1Q   Median       3Q      Max
-13.3333 -2.7833  -0.6167  2.1750 14.5667

Coefficients: (1 not defined because of singularities)
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)          48.900    4.473 10.9333   4.35e-06 ***
control              34.333    6.325  5.4280   0.000625 ***
A                     12.933    6.325  2.0450   0.075129 .
B                     18.600    6.325  2.9410   0.018695 *
C
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.747 on 8 degrees of freedom
Multiple R-Squared: 0.7915, Adjusted R-squared: 0.7133
F-statistic: 10.12 on 3 and 8 DF,  p-value: 0.004248
```r
> summary(lm(weight~trt, data=sugar))

Call:
  lm(formula = weight ~ trt, data = sugar)

Residuals:
    Min     1Q Median     3Q    Max
  -13.333 -2.7833 -0.6167  2.1750 14.5667

Coefficients:  
                  Estimate Std. Error  t value Pr(>|t|)
(Intercept)  83.2333    4.4733  18.6090 7.17e-08 ***
trtA         -21.4000    6.3250  -3.3833 0.009597 **
trtB         -15.7333    6.3250  -2.4870 0.037680 *
trtC         -34.3333    6.3250  -5.4280 0.000625 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 7.747 on 8 degrees of freedom
Multiple R-Squared: 0.7915, Adjusted R-squared: 0.7133
F-statistic: 10.12 on 3 and 8 DF,  p-value: 0.004248
```

can change to, e.g., difference from overall mean using options(contrasts = <your choice here>)
Section 7.2: Block designs

• We will discuss factorial designs and designed experiments next semester
Section 7.3: Nominal and Continuous Predictors

```r
> leaftemp
   CO2level  vapPress tempDiff
1  high     2.56     1.50
2  low      1.88     1.36
3  medium   2.38     1.94
4  high     2.55     0.85
5  low      2.20     0.60
6  medium   2.72     0.83
7  high     2.17    -0.04
8  medium   2.21    -0.11
9  high     1.64     1.25
10 low      1.75     0.23
11 medium   1.67     0.85
```
> leaf.lm1 <- lm(tempDiff ~ 1, data=leaftemp)
> leaf.lm2 <- lm(tempDiff ~ vapPress, data=leaftemp)
> leaf.lm3 <- lm(tempDiff ~ CO2level + vapPress, data=leaftemp)
> leaf.lm4 <- lm(tempDiff ~ CO2level + vapPress + vapPress:CO2level, data=leaftemp)
A: Single line
\[ \text{tempDiff} = 3.1 + 0.96 \times \text{vapPress} \]

B: Parallel lines
Intercepts are: 2.99, 3.349
Slopes are: 0.84

C: Separate lines
Intercepts are: 1.296, 4.89
Slopes are: -0.02, -0.76, -1.40
> anova(leaf.lm1, leaf.lm2, leaf.lm3, leaf.lm4)

Analysis of Variance Table

Model 1: tempDiff ~ 1
Model 2: tempDiff ~ vapPress
Model 3: tempDiff ~ CO2level + vapPress
Model 4: tempDiff ~ CO2level + vapPress + vapPress:CO2level

<table>
<thead>
<tr>
<th>Res.Df</th>
<th>RSS</th>
<th>Df Sum of Sq</th>
<th>F</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>61</td>
<td>39.999</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>60</td>
<td>34.727</td>
<td>1</td>
<td>5.272</td>
</tr>
<tr>
<td>3</td>
<td>58</td>
<td>28.183</td>
<td>2</td>
<td>6.544</td>
</tr>
<tr>
<td>4</td>
<td>56</td>
<td>26.056</td>
<td>2</td>
<td>2.126</td>
</tr>
</tbody>
</table>
---
plot(leaf.lm3)
Section 7.4: Polynomial Regression

- Reading
Section 7.5: Curve Fitting
Basis Expansions for Linear Models

\[ f(X) = \sum_{m=1}^{M} \beta_m h_m(X) \]

Here the \( h_m \)'s might be:

- \( h_m(X) = X_m \), \( m = 1, \ldots, p \) recovers the original model
- \( h_m(X) = X_j^2 \) or \( h_m(X) = X_j X_k \)
- \( h_m(X) = I(L_m \leq X_k \leq U_m) \),
Regression Splines

Bottom left panel uses:

\[ h_1(X) = 1 \]
\[ h_2(X) = X \]
\[ h_3(X) = (X - \xi_1)_+ \]
\[ h_4(X) = (X - \xi_2)_+ \]

Number of parameters = (3 regions) \( \times \) (2 params per region) 
- (2 knots \( \times \) 1 constraint per knot) 
\[ = 4 \]
cubic spline

Discontinuous

Continuous

Continuous First Derivative

Continuous Second Derivative

\( \xi_1 \quad \xi_2 \)
Cubic Spline
continuous first and second derivatives

\[ h_1(X) = 1 \]
\[ h_2(X) = X \]
\[ h_3(X) = X^2 \]
\[ h_4(X) = X^3 \]
\[ h_5(X) = (X - \xi_1)_+^3 \]
\[ h_6(X) = (X - \xi_2)_+^3 \]

Number of parameters = (3 regions) \times (4 params per region) - (2 knots \times 3 constraints per knot) = 6

Knot discontinuity essentially invisible to the human eye
Natural Cubic Spline

Adds a further constraint that the fitted function is linear beyond the boundary knots

A natural cubic spline model with $K$ knots is represented by $K$ basis functions:

$$H_1(X) = 1$$

$$H_2(X) = X$$

$$H_{k+2}(X) = d_k(X) - d_{K-1}(X), \quad \text{where}$$

$$d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k}$$

Each of these basis functions has zero $2^{\text{nd}}$ and $3^{\text{rd}}$ derivative outside the boundary knots
Natural Cubic Spline Models

Can use these ideas in, for example, regression models.

For example, if you use 4 knots and hence 4 basis functions per predictor variable, then simply fit logistic regression model with four times the number of predictor variables…
Smoothing Splines

Consider this problem: among all functions $f(x)$ with two continuous derivatives, find the one that minimizes the penalized residual sum of squares:

$$RSS(f, \lambda) = \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt$$

$\lambda=0$ : $f$ can be any function that interpolates the data
$\lambda=\infty$ : least squares line
Smoothing Splines

Theorem: The unique minimizer of this penalized RSS is a natural cubic spline with knots at the unique values of $x_i, i=1,\ldots,N$

Seems like there will be $N$ features and presumably overfitting of the data. But,… the smoothing term shrinks the model towards the linear fit

$$f(x) = \sum_{i=1}^{N} H_j(x)\theta_j$$

$$RSS(\theta, \lambda) = (y - H\theta)^T (y - H\theta) + \lambda \theta^T \Omega_H \theta$$

where

$$\{H\}_{ij} = H_j(x_i) \quad \text{and} \quad \{\Omega_H\}_{jk} = \int H_j''(t)H_k''(t)dt$$

$$\hat{\theta} = (H^T H + \lambda \Omega_H)^{-1} H^T y = S_\lambda y$$

This is a generalized ridge regression

Can show that $$S_\lambda = (I + \lambda K)^{-1}$$ where $K$ does not depend on $\lambda$
Figure 5.6: The response is the relative change in bone mineral density measured at the spine in adolescents, as a function of age. A separate smoothing spline was fit to the males and females, with $\lambda \approx 0.00022$. This choice corresponds to about 12 degrees of freedom.
Nonparametric Logistic Regression

Consider logistic regression with a single $x$:

$$\text{logit}(Y) := \log \frac{\Pr(Y = 1 \mid X = x)}{\Pr(Y = 0 \mid X = x)} = f(x)$$

and a penalized log-likelihood criterion:

$$l(f, \lambda) = \sum_{i=1}^{N} \left\{ y_i \log p(x_i) + (1 - y_i) \log(1 - p(x_i)) \right\} - \frac{1}{2} \lambda \int \left\{ f''(t) \right\} dt$$

$$= \sum_{i=1}^{N} \left\{ y_i f(x_i) + \log(1 + e^{f(x_i)}) \right\} - \frac{1}{2} \lambda \int \left\{ f''(t) \right\} dt$$

Again can show that the optimal $f$ is a natural spline with knots at the datapoint.

Can use Newton-Raphson to do the fitting.
Thin-Plate Splines

The discussion up to this point has been one-dimensional. The higher-dimensional analogue of smoothing splines are “thin-plate splines.” In 2-D, instead of minimizing:

$$RSS(f, \lambda) = \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda f''(t) dt$$

minimize:

$$RSS(f, \lambda) = \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda J(f)$$

where

$$J(f) = \iint \left( \frac{\partial^2 f(x)}{\partial x_1^2} \right)^2 + \left( \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} \right)^2 + \left( \frac{\partial^2 f(x)}{\partial x_2^2} \right)^2 dx_1 dx_2$$
Thin-Plate Splines

The solution has the form:

\[
f(x) = \beta_0 + \beta^T x + \sum_{j=1}^{N} \alpha_j h_j(x) \quad \text{where}
\]

\[
h_j(x) = \eta(\|x - x_j\|) \quad \text{and} \quad \eta(z) = z^2 \log z^2
\]

a type of “radial basis function”
Figure 5.12: A thin-plate spline fit to the heart disease data, displayed as a contour plot. The response is systolic blood pressure, modeled as a function of age and obesity. The data points are indicated, as well as the lattice of points used as knots. Care should be taken to use knots from the lattice inside the convex hull of the data (red), and ignore those outside (green).
ns \{splines\}

Generate a Basis Matrix for Natural Cubic Splines

Description

Generate the B-spline basis matrix for a natural cubic spline.

Usage

```r
ns(x, df = NULL, knots = NULL, intercept = FALSE,
   Boundary.knots = range(x))
```

Arguments

- `x`  
  the predictor variable. Missing values are allowed.

- `df`  
  degrees of freedom. One can supply `df` rather than knots; `ns()` then chooses `df - 1` intercept knots at suitably chosen quantiles of `x` (which will ignore missing values).

- `knots`  
  breakpoints that define the spline. The default is no knots; together with the natural boundary conditions this results in a basis for linear regression on `x`. Typical values are the mean or median for one knot, quantiles for more knots. See also `Boundary.knots`.

- `intercept`  
  if `TRUE`, an intercept is included in the basis; default is `FALSE`.

- `Boundary.knots`  
  boundary points at which to impose the natural boundary conditions and anchor the B-spline basis (default the range of the data). If both `knots` and `Boundary.knots` are supplied, the basis parameters do not depend on `x`. Data can extend beyond `Boundary.knots`.
library(splines)
attach(fruitohms)

plot(ohms, juice)
abline(lm(juice~ohms))
par(mfrow=c(2,2))
plot(lm(juice~ohms))

fruit.lmb4 <- lm(ohms ~ ns(juice,4))
summary(lmb4)
par(mfrow=c(2,2))
plot(fruit.lmb4)
plot(ohms~juice, cex=0.8, xlab="Apparent juice content (%)", ylab="Resistance (ohms)",main="natural spline, 2 knots")
CIcurves(form=ohms ~ ns(juice,3), data=fruitohms, newdata=data.frame(juice=pretty(fruitohms$juice,20)))

plot(ohms~juice, cex=0.8, xlab="Apparent juice content (%)", ylab="Resistance (ohms)",main="natural spline, 3 knots")
CIcurves(form=ohms ~ ns(juice,4), data=fruitohms, newdata=data.frame(juice=pretty(fruitohms$juice,20)))

plot(ohms~juice, cex=0.8, xlab="Apparent juice content (%)", ylab="Resistance (ohms)",main="polynomial, degree 3")
CIcurves(form=ohms ~ poly(juice,3), data=fruitohms, newdata=data.frame(juice=pretty(fruitohms$juice,20)))

plot(ohms~juice, cex=0.8, xlab="Apparent juice content (%)", ylab="Resistance (ohms)",main="polynomial, degree 4")
CIcurves(form=ohms ~ poly(juice,4), data=fruitohms, newdata=data.frame(juice=pretty(fruitohms$juice,20)))
## Code to fit the line, the curve, determine the pointwise coverage
## bounds and create the plots

CIcurves <-
  function(form=grain~rate, data=seedrates, lty=1, col=3, 
            newdata=data.frame(rate=seq(from=50, to=175, by=25))){
    seedrates.lm <- lm(form, data=data)
    x <- newdata[, all.vars(form)[2]]
    hat <- predict(seedrates.lm, newdata=newdata, interval="confidence")
    lines(spline(x, hat[, "fit"]))
    lines(spline(x, hat[, "lwr"], lty=lty, col=col)
    lines(spline(x, hat[, "upr"], lty=lty, col=col)
  }