Companion to Linear Models with R (Faraway 2014)

Adam Wang

November 20, 2020

Contents

Preface

This is a book companion to Linear Models with R, by Julian Faraway (2014). This book, created with R package bookdown, was made while working through the text and contains my solutions to exercises and other supplemental material.

Errors not in errata.

p93 and p192, Figures 6.12 and 12.7. Residuals vs Leverage plots should show Cook's distance contours of 0.5 and 1, not 0 and 1. The right margins are cut off.

p103. The simex() measurement.error argument should be the standard deviation of measurement errors, not the variance. (This results in a SIMEX estimate of 3.99 instead of 3.96.)

p136, last paragraph. "The Box–Cox method is not the only way of transforming the predictors". Replace "predictors" with "response".

p150, first line. "In some situations, the model space is structued. . . ". Replace "structued" with "structured".

p154, last line. ". . . it finds the variables that produce the minumum RSS." Replace "minumum" with "minimum".

p160, Exercise 5. ". . . stack.loss as the predictor. . . ". Replace "predictor" with "response".

p176, Figure 11.10. The y-axis label cuts off the "hat" in $\hat{\beta}$.

p252, displayed equation. The blocking effect is β_j in the equation, but the text that directly follows states ρ_j is the blocking effect.

Chapter 1

Introduction

```
library(faraway)
```

1.1 Before You Start

1.2 Initial Data Analysis

1.3 When to Use Linear Modeling

1.4 History

Manilius data. The `aggregate()` code is similar in function and speed to the following.

```
df <- manilius
moon3 <- data.frame(matrix(ncol=4, nrow=0))
colnames(moon3) <- names(df)
for (group in 1:3){
  group_rows <- (df$group == group)
  group_df <- df[group_rows, ]
  group_sum <- colSums(group_df)
  moon3[group, ] <- c(group_sum)
}
moon3 <- moon3[, 1:3]
moon3
```

```
##      arc  sinang  cosang
## 1 118.13  8.4987 -0.7932
## 2 140.28 -6.1404  1.7443
## 3 127.53  2.9777  7.9649
```

In Python,

```
import pandas as pd

df = r.manilius
moon3 = pd.DataFrame(index=range(1, 4), columns=df.columns)
```

```
for group in range(1, 4):
    group_rows = (df['group'] == group)
    group_df = df[group_rows]
    group_sum = group_df.sum()
    moon3.loc[group] = group_sum
moon3 = moon3.iloc[:, 0:3]
moon3
```

```
##        arc  sinang  cosang
## 1  118.133  8.4987 -0.7932
## 2  140.283 -6.1404  1.7443
## 3  127.533  2.9777  7.9649
```

A more concise solution, similar to the book, uses the `groupby()` method.

```
df.groupby(['group']).sum()
```

```
##               arc  sinang  cosang
## group
## 1      118.133333  8.4987 -0.7932
## 2      140.283333 -6.1404  1.7443
## 3      127.533333  2.9777  7.9649
```

Alternatively, we could use the means

```
moon3 = df.groupby(['group']).mean()
```

since we are interested in solving the averaged group equations

$$\overline{\texttt{arc}}_i = \beta + \alpha\,\overline{\texttt{sinang}}_i + \gamma\,\overline{\texttt{cosang}}_i, \qquad i = 1, 2, 3.$$

In matrix form,

$$\begin{pmatrix} \overline{\texttt{arc}}_1 \\ \overline{\texttt{arc}}_2 \\ \overline{\texttt{arc}}_3 \end{pmatrix} = \begin{pmatrix} 1 & \overline{\texttt{sinang}}_1 & \overline{\texttt{cosang}}_1 \\ 1 & \overline{\texttt{sinang}}_2 & \overline{\texttt{cosang}}_2 \\ 1 & \overline{\texttt{sinang}}_3 & \overline{\texttt{cosang}}_3 \end{pmatrix} \begin{pmatrix} \beta \\ \alpha \\ \gamma \end{pmatrix},$$

or more concisely $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}$. Solving numerically,

```
import numpy as np

A = moon3[['sinang', 'cosang']]
A.insert(loc=0, column='intercept', value=1)
b = moon3['arc']
np.linalg.solve(A, b)
```

```
## array([14.54458591, -1.48982207,  0.13412639])
```

Exercises

I use `ggplot2`, `dplyr`, and the following function to plot univariate summaries as a visual alternative to `summary()`.

```
library(ggplot2)
library(dplyr)
```

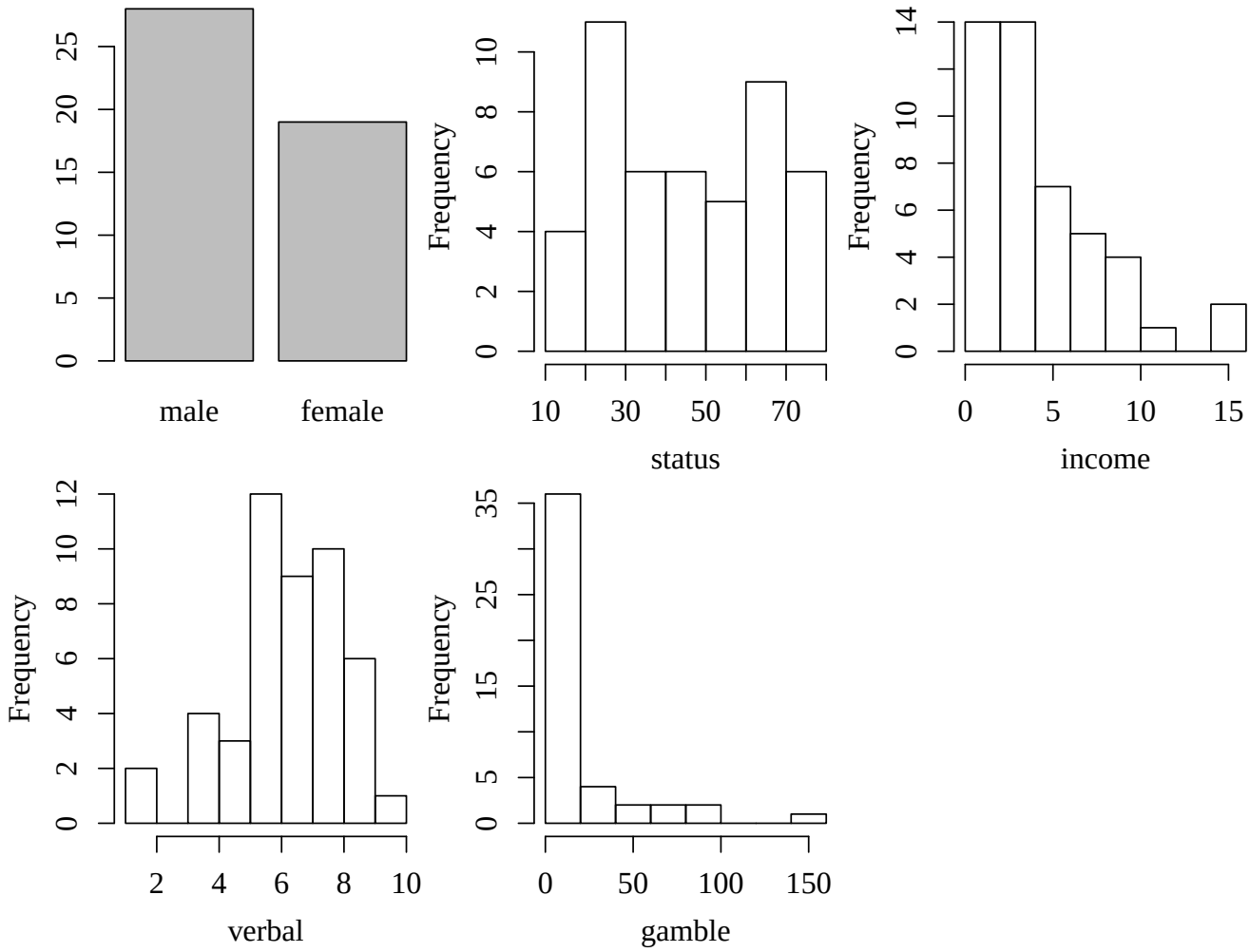
```
graphical_summary <- function(df) {
  # Plot bar charts of category counts for categorical variables
  # and histograms for continuous variables.
  for (var in names(df)) {
    x <- df[[var]]
    if (is.factor(x)) {
      plot(x)
    } else {
      hist(x, xlab = var, main = '')
    }
  }
}
```

I also use the following function to count the number rows in a data frame with at least one missing value.

```
count_na_rows <- function(df) {
  sum(apply(is.na(df), 1, sum) > 0)
}
```

Exercise 1: Overview of teenage gambling data. Univariate summaries are plotted below, after converting `sex` to a categorical variable.

```
df <- teengamb
df$sex <- factor(df$sex, levels = c(0, 1), labels = c('male', 'female'))
graphical_summary(df)
```



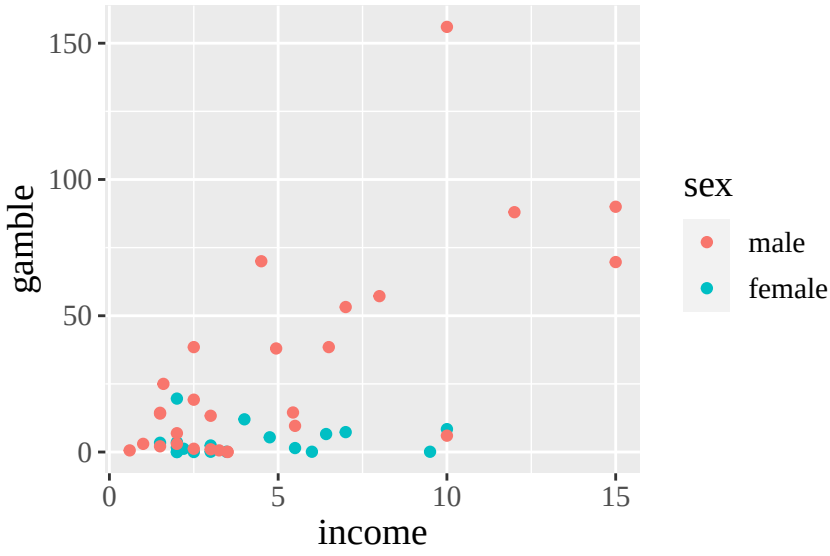
There are 0 missing values. Inspecting the correlation matrix among continuous variables,

```
cor(df[, 2:5], method='spearman')
```

```
##          status   income   verbal   gamble
## status  1.00000 -0.28037  0.53250  0.16451
## income -0.28037  1.00000 -0.17193  0.36612
## verbal  0.53250 -0.17193  1.00000 -0.14953
## gamble  0.16451  0.36612 -0.14953  1.00000
```

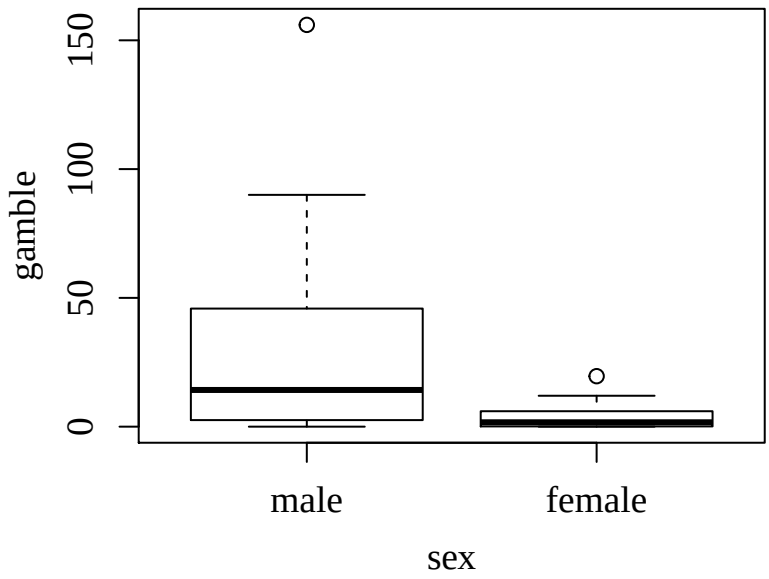
we observe `gamble` is most strongly correlated with `income`. The correlation is particularly strong among males: $\rho = 0.57666$. A scatterplot grouped by `sex` visualizes this and also reveals males tend to gamble more across all incomes.

```
ggplot(df, aes(x = income, y = gamble, color = sex)) +
  geom_point()
```



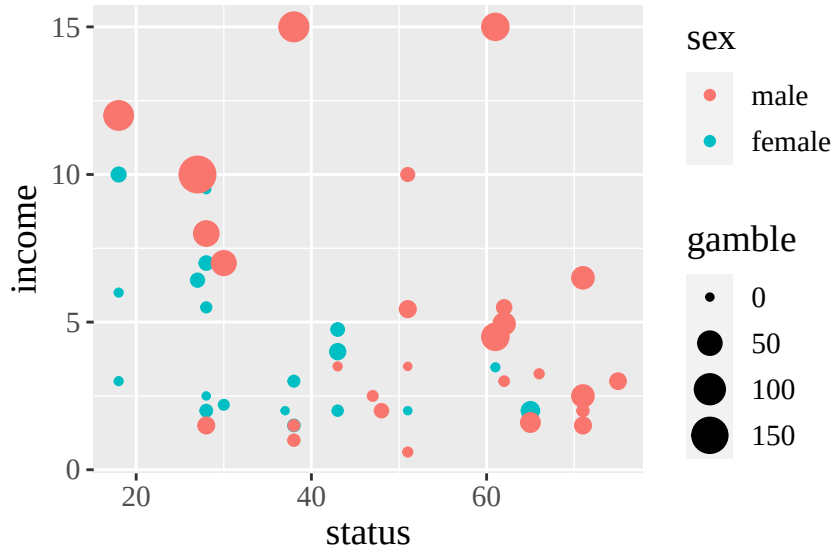
A boxplot summarizes the sex difference.

```
plot(gamble ~ sex, df)
```



Surprisingly, `status` and `income` are negatively correlated for both sexes.

```
ggplot(df, aes(x = status, y = income, size = gamble, color = sex)) +
  geom_point()
```



For females, this correlation is particularly strong: $\rho = -0.53767$. For males, there is almost no correlation after removing the three largest values of `gamble` $(156, 90, 88)$: $\rho = -0.02116$.

Exercise 2: Overview of US wages data. I convert categorical variables to factors and combine all binary location data to a single categorical factor `loc` with 4 levels.

```
df <- uswages
df$race <- factor(df$race, levels = c(0, 1), labels = c('white', 'black'))
df$smsa <- factor(df$smsa, levels = c(0, 1), labels = c('not SMSA', 'SMSA'))
df$pt <- factor(df$pt, levels = c(0, 1), labels = c('no pt', 'pt'))
loc_inds <- 6:9
df$loc <- factor(
  max.col(df[, loc_inds]),
  levels = c(1, 2, 3, 4),
  labels = c('ne', 'mw', 'so', 'we')
)
```

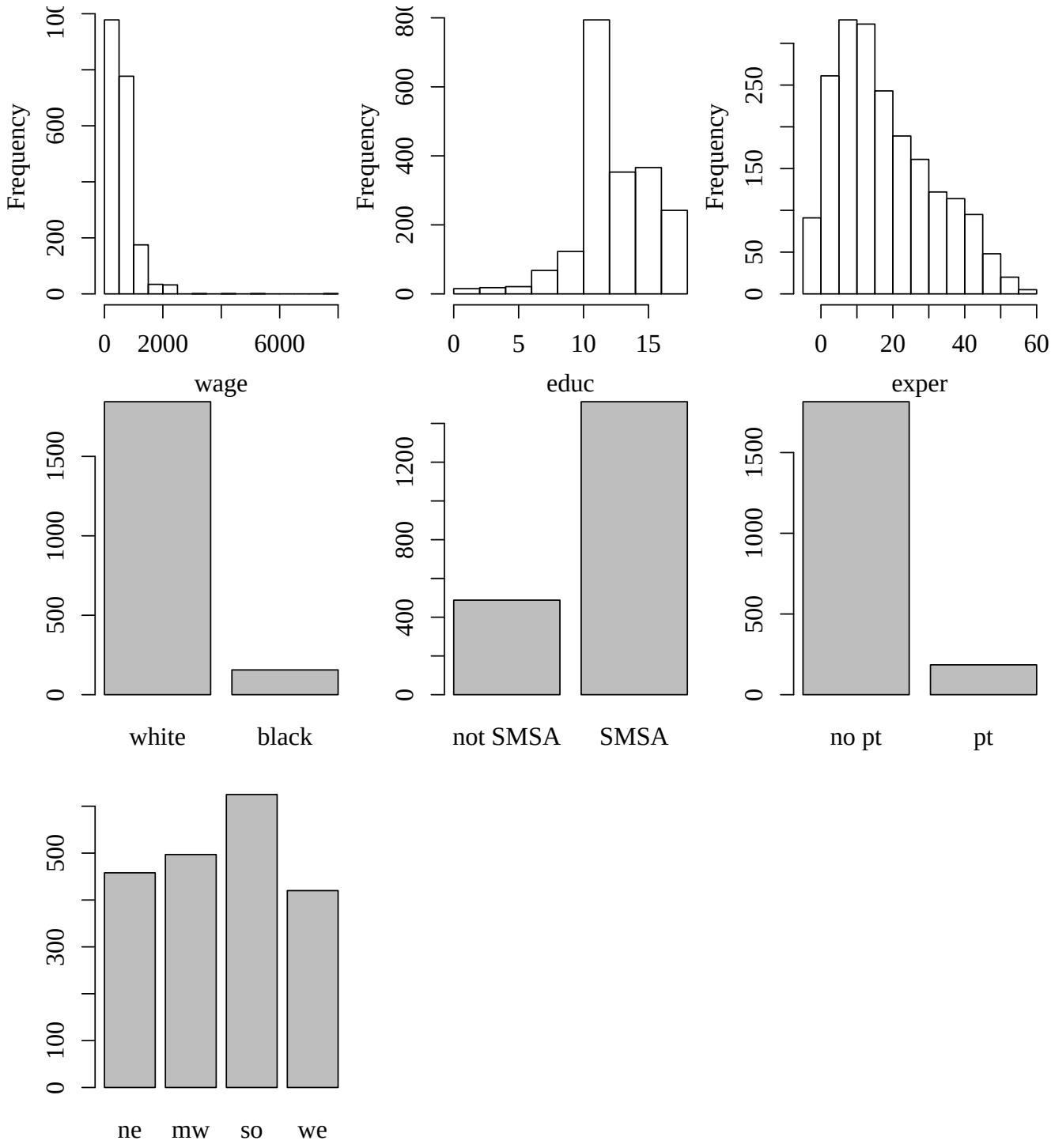
There are 0 missing values. A quick logic check shows everyone lives in one unique location.

```
sum((df$ne + df$mw + df$we + df$so) == 1) / nrow(df)
```

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

Univariate summaries are plotted below.

```
graphical_summary(df[, -loc_inds])
```



All continuous variables possess skewed distributions. Most sampled workers are white, live in Standard Metropolitan Statistical Area (SMSA), and work full time. The geographic distribution is almost uniform, with a small southern majority.

Inspecting the correlation matrix among continuous variables,

```
cont_inds <- 1:3
cor(df[, cont_inds], method='spearman')
```

```
##           wage      educ     exper
## wage  1.00000  0.30309  0.30597
```

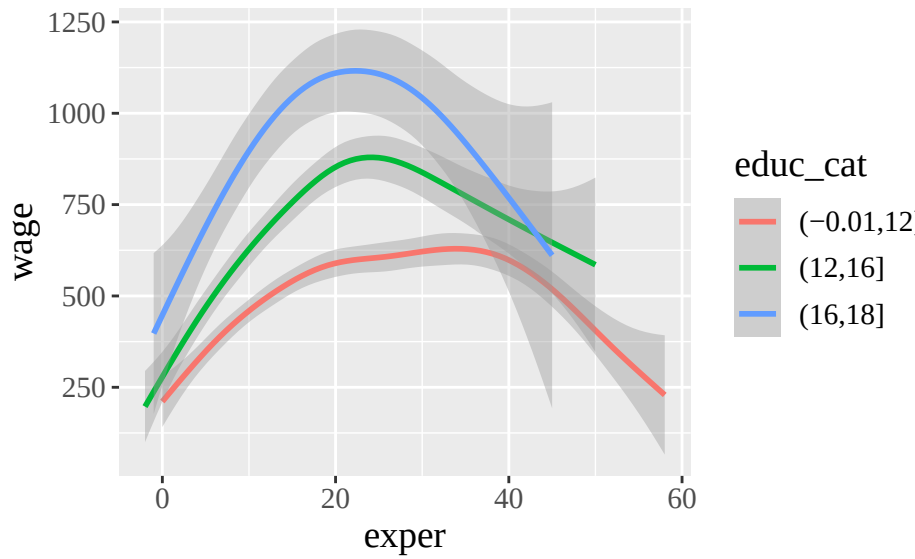
```
## educ   0.30309  1.00000 -0.23385
## exper 0.30597 -0.23385  1.00000
```

we observe wage has a positive correlation with years of education and experience. The negative `educ`/`exper` correlation is a result of a tradeoff between time spent obtaining education against working.

Numerical summaries are not sufficient to capture the underlying relationship. A smoothed trend line of wage against experience, grouped by education, demonstrates this. (The two largest wages are removed.)

```
df$educ_cat <- cut(df$educ, breaks = c(-.01, 12, 16, 18))
df %>% filter(wage < 5000) %>%
  ggplot(aes(x = exper, y = wage, color = educ_cat)) +
  geom_smooth()
```

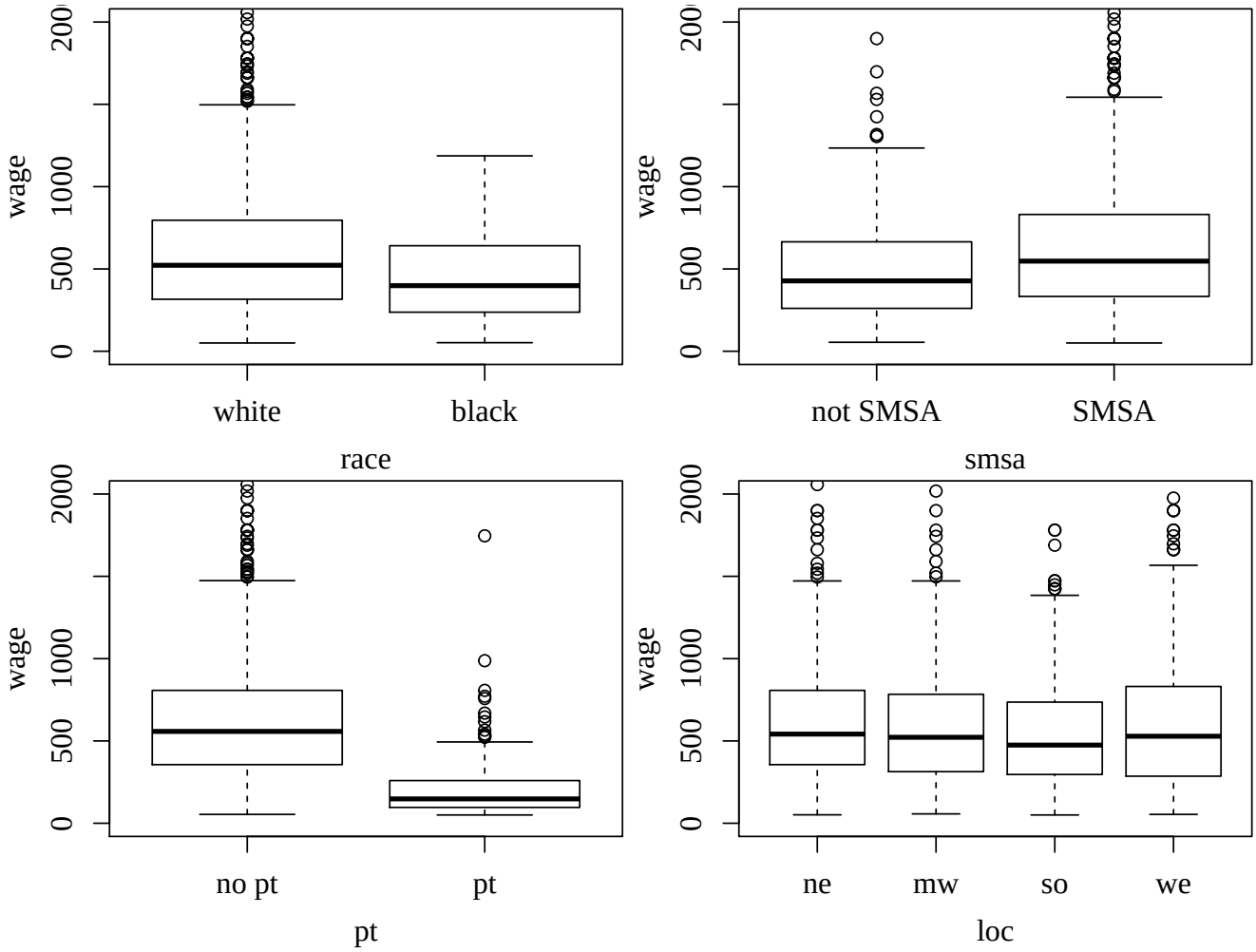
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Wage tends to be higher for the more educated groups. For all groups, wage increases with experience initially, but then decreases. The decrease might be explained by experienced workers being older, and older workers being less desirable to employers.

Categorical associations with wage can be visualized with boxplots.

```
plot(wage ~ race, df, ylim = c(0, 2000))
plot(wage ~ smsa, df, ylim = c(0, 2000))
plot(wage ~ pt, df, ylim = c(0, 2000))
plot(wage ~ loc, df, ylim = c(0, 2000))
```



Wages tend to be slightly higher for `white` and those who live in SMSAs, and substantially higher for those who work full time. Wages are similar across regions.

Chapter 2

Estimation

2.1 Linear Model

2.2 Matrix Representation

2.3 Estimating β

Geometric approach. Consider an extreme example where $n = p = 2$. Because the data $\boldsymbol{y}$ have the same dimension as the model with parameters $\boldsymbol{\beta}$, $\boldsymbol{\beta}$ can be determined exactly. Mathematically, we are solving two equations with two unknowns. Geometrically, the data and parameters exist in the same plane, hence the parameters can be chosen to map directly onto the data.

Introducing a third data point so $n = 3$ and $p = 2$, we can no longer determine $\boldsymbol{\beta}$ exactly. Mathematically, there are more equations than unknowns. Geometrically, the data live in 3-dimensions while the parameters can only perfectly map a point in a 2-dimensional plane; this scenario is represented by Figure 2.1. At best, we can only approximate $\boldsymbol{y} \approx \boldsymbol{X}\boldsymbol{\beta}$, which we do by incorporating an error term $\boldsymbol{\varepsilon}$. Least squares estimation seeks to minimize the sum of squares $\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}$. This is done geometrically by projecting $\boldsymbol{y}$ onto 2-dimensional $\widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}$. Projecting minimizes the length of the estimated error vector $\widehat{\boldsymbol{\varepsilon}}$ by ensuring it is orthogonal to $\widehat{\boldsymbol{y}}$.

For general n and p, the intuition behind the geometric approach is the same, although we cannot visualize it. The data $\boldsymbol{y}$ live in an n-dimensional space and we seek to explain it in a p-dimensional subspace with parameters $\boldsymbol{\beta}$. We acknowledge there are $(n - p)$ dimensions we cannot explain, namely errors $\boldsymbol{\varepsilon}$. We minimize the unexplainable errors, or equivalently maximize the amount the model can explain, by projecting $\boldsymbol{y}$ onto $\widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}$.

2.4 Least Squares Estimation

Geometric approach. By construction, the residual vector $\widehat{\boldsymbol{\varepsilon}}$ is orthogonal to the predicted values $\widehat{\boldsymbol{y}}$. More generally, $\widehat{\boldsymbol{\varepsilon}}$ is orthogonal to any vector in the p-dimensional space spanned by the columns of $\boldsymbol{X}$ (e.g., see Figure 2.1). It follows that the dot product between $\widehat{\boldsymbol{\varepsilon}}$ and any column of $\boldsymbol{X}$ is zero. In matrix notation,

$$\boldsymbol{X}^T\widehat{\boldsymbol{\varepsilon}} = \boldsymbol{0},$$

where $\boldsymbol{0}$ is a $p \times 1$ vector of zeros. Substituting $\widehat{\boldsymbol{\varepsilon}} = \boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}$,

$$\boldsymbol{0} = \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}) = \boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{X}\widehat{\boldsymbol{\beta}},$$

which implies the *normal equations*

$$\boldsymbol{X}^T\boldsymbol{X}\widehat{\boldsymbol{\beta}} = \boldsymbol{X}^T\boldsymbol{y}.$$

Comment. The normal equations cannot be simplified to $\boldsymbol{X}\widehat{\boldsymbol{\beta}} = \boldsymbol{y}$ because $\boldsymbol{X}$ is not in general square, hence its inverse does not (in general) exist. The matrix product $\boldsymbol{X}^T\boldsymbol{X}$ is square, so least squares solutions for $\widehat{\boldsymbol{\beta}}$ do exist.

Projection operator. The *hat matrix* $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$ is common in linear algebra where it is known as a *projection operator*, e.g. see Wikipedia. It projects any vector onto the space spanned by the columns of $\boldsymbol{X}$, which can be viewed as a (not necessarily orthonormal) basis. Indeed it is simple to verify $\boldsymbol{H}^2 = \boldsymbol{H}$. The predicted values $\widehat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}$ indeed are projections of $\boldsymbol{y}$ onto the space spanned by the columns of $\boldsymbol{X}$.

Residual maker matrix. The hat matrix projects $\boldsymbol{y}$ onto $\widehat{\boldsymbol{y}}$, i.e. $\boldsymbol{H}\boldsymbol{y} = \widehat{\boldsymbol{y}}$. The remaining components are the residuals such that $\boldsymbol{y} = \widehat{\boldsymbol{y}} + \widehat{\boldsymbol{\varepsilon}}$. We can define a related operator $\boldsymbol{M} = (\boldsymbol{I} - \boldsymbol{H})$ that extracts the residuals, i.e. $\boldsymbol{M}\boldsymbol{y} = \widehat{\boldsymbol{\varepsilon}}$, hence its name *residual maker matrix*. The residual maker matrix satisfies $\boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{M}$, used in the RSS formula. To prove this, note $\boldsymbol{I}^T = \boldsymbol{I}$ and for matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, $(\boldsymbol{A} + \boldsymbol{B})^T = \boldsymbol{A}^T + \boldsymbol{B}^T$. Expanding yields

$$\boldsymbol{M}^T\boldsymbol{M} = (\boldsymbol{I} - \boldsymbol{H})^T(\boldsymbol{I} - \boldsymbol{H}) = \boldsymbol{I} - \boldsymbol{H} - \boldsymbol{H}^T + \boldsymbol{H}^T\boldsymbol{H}. \tag{2.1}$$

Using the definition of the hat matrix and some matrix properties,

$$\begin{aligned}
\boldsymbol{H}^T &= \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T && \text{(definition)} \\
&= \boldsymbol{X}[(\boldsymbol{X}^T\boldsymbol{X})^{-1}]^T\boldsymbol{X}^T && [(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T\boldsymbol{A}^T] \\
&= \boldsymbol{X}[(\boldsymbol{X}^T\boldsymbol{X})^T]^{-1}\boldsymbol{X}^T && [(\boldsymbol{A}^{-1})^T = (\boldsymbol{A}^T)^{-1}] \\
&= \boldsymbol{X}[\boldsymbol{X}^T\boldsymbol{X}]^{-1}\boldsymbol{X}^T && [(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T\boldsymbol{A}^T] \\
&= \boldsymbol{H}.
\end{aligned}$$

Furthermore from $\boldsymbol{H}^2 = \boldsymbol{H}$, the last two terms of Equation (2.1) cancel, leaving $\boldsymbol{M}^T\boldsymbol{M} = \boldsymbol{M}$.

Estimator of RSS. Proving $\mathbb{E}(\widehat{\boldsymbol{\varepsilon}}^T\widehat{\boldsymbol{\varepsilon}}) = \sigma^2(n-p)$ requires a lengthy amount of matrix algebra. We can make sense of the formula by considering limiting cases. Suppose the model contains only an intercept $\boldsymbol{\beta}_0$ so $p = 1$. $\boldsymbol{\beta}_0$ is just a length n vector of sample means $\overline{y}$, so the residual sum of squares simplifies to $\widehat{\boldsymbol{\varepsilon}}^T\widehat{\boldsymbol{\varepsilon}} = \sum_{i=1}^n (y_i - \overline{y})^2$. This is proportional to the sample variance of $\boldsymbol{y}$, $S^2 = \sum_{i=1}^n (y_i - \overline{y})^2/(n-1)$, which is an unbiased estimator of σ^2. It follows that $\mathbb{E}(\widehat{\boldsymbol{\varepsilon}}^T\widehat{\boldsymbol{\varepsilon}}) = \sigma^2(n-1)$. On the other extreme, if our model contains $p = n$ parameters, we would obtain a 'perfect' fit (e.g., fitting a line to two data points), and obtain $\mathbb{E}(\widehat{\boldsymbol{\varepsilon}}^T\widehat{\boldsymbol{\varepsilon}}) = 0$. In between, the RSS decreases monotonically as a function of p. The reduction in degrees of freedom as more parameters are estimated must be accounted for to construct an unbiased estimator of σ^2.

2.5 Examples of Calculating $\widehat{\beta}$

2.6 Example

2.7 QR Decomposition

2.8 Gauss–Markov Theorem

Estimable functions. The quantity $\psi = \boldsymbol{c}^T\boldsymbol{\beta}$ is useful to estimate because it is a prediction on a future observation with predictors $\boldsymbol{x} = \boldsymbol{c}$. ψ is not estimable when $\boldsymbol{X}$ has linearly dependent

predictors so that $\boldsymbol{X}$ is not full rank. To see this, observe

$$\mathbb{E}\,\boldsymbol{a}^T\boldsymbol{y} = \boldsymbol{a}^T\boldsymbol{X}\boldsymbol{\beta}.$$

If $\boldsymbol{X}$ is not full rank, $\boldsymbol{a}^T\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{c}^T\boldsymbol{\beta}$ does not hold for all $\boldsymbol{\beta}$. For instance, if

$$\boldsymbol{X} = \begin{pmatrix} 1 & 2 \\ 0 & 0 \end{pmatrix} \qquad \Rightarrow \qquad \boldsymbol{a}^T\boldsymbol{X}\boldsymbol{\beta} = a_1\beta_1,$$

then $\psi = c_1\beta_1 + c_2\beta_2$ cannot be estimable when $\beta_1 = 0$ and c_2 and β_2 are nonzero.

2.9 Goodness of Fit

2.10 Identifiability

2.11 Orthogonality

Exercises

```
library(faraway)
```

Residual properties.

(1) $\sum_{i=1}^n \widehat{\varepsilon}_i = 0$ when an intercept is included in the model, which implies the mean of the residuals is zero.
Proof. Rearrange the normal equations in the book to

$$\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}) = \boldsymbol{X}^T\widehat{\boldsymbol{\varepsilon}} = \boldsymbol{0}.$$

If an intercept is included, $\boldsymbol{X}^T$ contains a row of ones which imposes constraint

$$\boldsymbol{1}^T\widehat{\boldsymbol{\varepsilon}} = \sum_{i=1}^n \widehat{\varepsilon}_i = 0. \qquad \blacksquare$$

(2) $\operatorname{cov}(\widehat{\varepsilon}, x_j) = 0$ for any predictor x_j, $j = 1, 2, \ldots, p$. ($\widehat{\varepsilon}$ and x_j are random variables with n sample values in vectors $\widehat{\boldsymbol{\varepsilon}}$ and $\boldsymbol{x}_j$.)
Proof. Since the mean of the residuals is zero, from the definition of covariance,

$$\operatorname{cov}(\widehat{\varepsilon}, x_j) = \mathbb{E}(\widehat{\varepsilon}x_j) - \underbrace{\mathbb{E}\,\widehat{\varepsilon}}_{=0}\mathbb{E}\,x_j = \mathbb{E}(\widehat{\varepsilon}x_j) = \frac{1}{n}\boldsymbol{x}_j^T\widehat{\boldsymbol{\varepsilon}}.$$

Analogous to the proof in (1), this is equal to zero from the normal equations for all j. $\blacksquare$

(3) $\operatorname{cov}(\widehat{\varepsilon}, \widehat{y}) = 0$.
Proof. $\widehat{y}$ is a linear combination of the x_j (plus a constant). It follows that $\operatorname{cov}(\widehat{\varepsilon}, \widehat{y})$ is a linear combination of covariances of the form $\beta_j \operatorname{cov}(\widehat{\varepsilon}, x_j)$ (plus $\operatorname{cov}(\widehat{\varepsilon}, \beta_0)$), each of which is equal to zero from (2). $\blacksquare$

(4) $\operatorname{cov}(\widehat{\varepsilon}, y) \neq 0$ in general.
Proof. $\operatorname{cov}(\widehat{\varepsilon}, y) = \operatorname{cov}(\widehat{\varepsilon}, \widehat{y} + \widehat{\varepsilon}) = \operatorname{cov}(\widehat{\varepsilon}, \widehat{y}) + \operatorname{cov}(\widehat{\varepsilon}, \widehat{\varepsilon}) = \operatorname{var}(\widehat{\varepsilon})$ from (3). $\blacksquare$

Comment. Properties (2) and (3) only measure linear relationships. Residuals can still depend on x_j and $\widehat{y}$ while maintaining zero covariance, e.g. quadratically, indicating model misspecification. Residuals are rarely plotted against observed values y due to property (4), making detection of model misspecification more difficult.

Exercise 1: Regression of teenage gambling data.

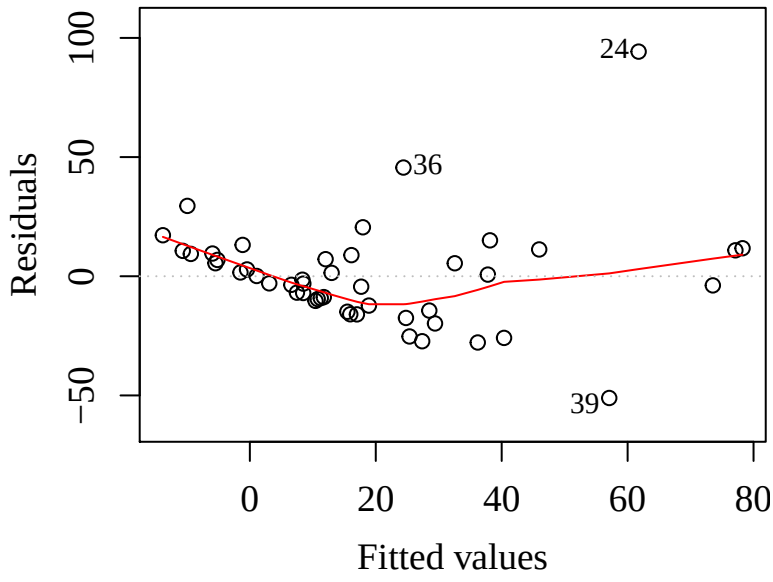
```
df <- teengamb
df$sex <- factor(df$sex, levels = c(0, 1), labels = c('male', 'female'))
lmod <- lm(gamble ~ sex + status + income + verbal, data = df)
lmodsum <- summary(lmod)
lmodsum
```

```
##
## Call:
## lm(formula = gamble ~ sex + status + income + verbal, data = df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -51.08 -11.32  -1.45   9.45  94.25
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  22.5557    17.1968    1.31     0.20
## sexfemale   -22.1183     8.2111   -2.69     0.01 *
## status        0.0522     0.2811    0.19     0.85
## income        4.9620     1.0254    4.84  1.8e-05 ***
## verbal       -2.9595     2.1722   -1.36     0.18
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22.7 on 42 degrees of freedom
## Multiple R-squared:  0.527,Adjusted R-squared:  0.482
## F-statistic: 11.7 on 4 and 42 DF,  p-value: 1.81e-06
```

(a) $R^2 \approx 0.527$ proportion of variation in the response is explained by the predictors.

(b) R plots this automatically.

```
plot(lmod, which = 1)
```



Alternatively, we can obtain the value and case number as follows.

```
c(max(lmod$residuals), which(lmod$residuals == max(lmod$residuals)))
```

```
##            24
## 94.252 24.000
```

(c–e) As expected from residual property (1), the mean of the residuals is zero.

```
c(mean(lmod$residuals), median(lmod$residuals))
```

```
## [1] -3.0653e-17 -1.4514e+00
```

As expected from residual properties (2) and (3), the correlation between residuals and fitted values and predictors is zero.

```
cor(lmod$residuals, lmod$fitted.values)
```

```
## [1] -1.0707e-16
```

```
cor(lmod$residuals, df$income)
```

```
## [1] -7.2424e-17
```

(The correlation with observed values is nonzero, as expected from residual property (4). It is equal to 0.68795.)

(f) The difference in predicted response for a female compared to a male, all other variables held constant, is given by the estimated `sex` beta coefficient. To see why, write the fitted value as

$$\widehat{y} = \widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \cdots + \widehat{\beta}_{\texttt{sex}} \texttt{sex} + \cdots + \beta_p x_p.$$

Substituting values of sex, 0 for male and 1 for female, and taking the difference:

$$\widehat{y}_{\text{female}} - \widehat{y}_{\text{male}} = \widehat{\beta}_{\text{sex}}.$$

Our model predicts, all other variables constant, females gamble $|\widehat{\beta}_{\text{sex}}| \approx 22.1$ pounds per year less than males.

Exercise 2: Regression of US wages data. Untransformed model:

```
df <- uswages
summary(lm(wage ~ educ + exper, data = df))
```

```
##
## Call:
## lm(formula = wage ~ educ + exper, data = df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -1018   -238    -51    150   7229
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -242.799     50.682   -4.79  1.8e-06 ***
## educ          51.175      3.342   15.31  < 2e-16 ***
## exper          9.775      0.751   13.02  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 428 on 1997 degrees of freedom
## Multiple R-squared:  0.135,Adjusted R-squared:  0.134
## F-statistic:  156 on 2 and 1997 DF,  p-value: <2e-16
```

The education coefficient represents the predicted change in weekly wages for a one year increase in education, all other variables held constant. The proof is similar to Exercise 1(f).

Log-transformed model:

```
summary(lm(log(wage) ~ educ + exper, data = df))
```

```
##
## Call:
## lm(formula = log(wage) ~ educ + exper, data = df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.753 -0.350  0.107  0.438  3.570
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.65032    0.07835    59.4   <2e-16 ***
## educ         0.09051    0.00517    17.5   <2e-16 ***
## exper        0.01808    0.00116    15.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.661 on 1997 degrees of freedom
## Multiple R-squared:  0.175,Adjusted R-squared:  0.174
## F-statistic:  212 on 2 and 1997 DF,  p-value: <2e-16
```

Now the education coefficient represents the predicted change in the log of weekly wages for a one year increase in education, all other variables held constant. The untransformed model has

a more natural interpretation while the log-transformed model may better meet the structural model assumptions. In particular, the log-transformed wage distribution better resembles a normal distribution.

Exercise 3: Computing least squares regression coefficients. Using `lm()` for a cubic polynomial:

```
set.seed(1)
x <- 1:20
y <- x + rnorm(20)
form <- y ~ poly(x, degree = 3, raw = TRUE)
unname(lm(form)$coefficients)
```

```
## [1] -1.0251886  1.5393918 -0.0611986  0.0019656
```

A direct computation agrees:

```
X <- model.matrix(form)
unname(t(solve(t(X) %*% X) %*% t(X) %*% y))
```

```
##          [,1]   [,2]      [,3]      [,4]
## [1,] -1.0252 1.5394 -0.061199 0.0019656
```

but fails for degree 7+.

Exercise 4: Regression of prostate cancer data. First, convert categorical variables to factors and initialize metric storage arrays.

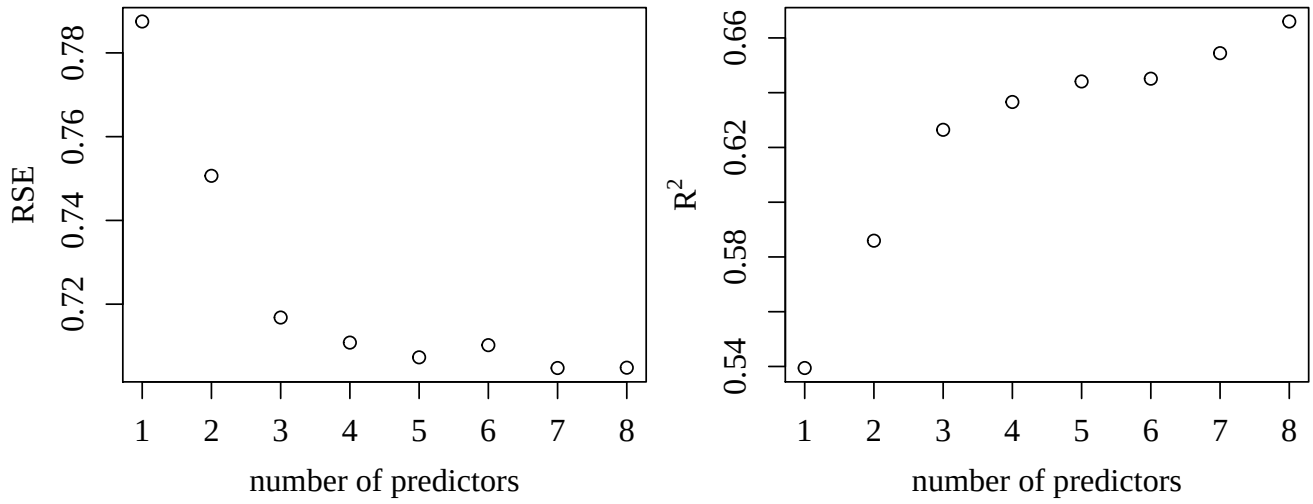
```
df <- prostate
df$svi <- factor(df$svi)
df$gleason <- factor(df$gleason)
predictors <- c(
  'lcavol', 'lweight', 'svi', 'lbph', 'age', 'lcp', 'pgg45', 'gleason'
)
p <- length(predictors)
rse <- vector(mode = 'numeric', length = p)
R2 <- vector(mode = 'numeric', length = p)
```

Next, systematically add variables, fit models, and store metrics.

```
form_str <- 'lpsa ~'
for (i in 1:p) {
  var <- predictors[i]
  if (i == 1) {
    form_str <- paste(form_str, var)
  } else {
    form_str <- paste(form_str, '+', var)
  }
  lmod <- lm(as.formula(form_str), df)
  lmodsum <- summary(lmod)
  rse[i] <- lmodsum$sigma
  R2[i] <- lmodsum$r.squared
}
```

Finally, plot the results.

```
plot(rse, xlab = 'number of predictors', ylab = 'RSE')
plot(R2, xlab = 'number of predictors', ylab = expr(R^2))
```



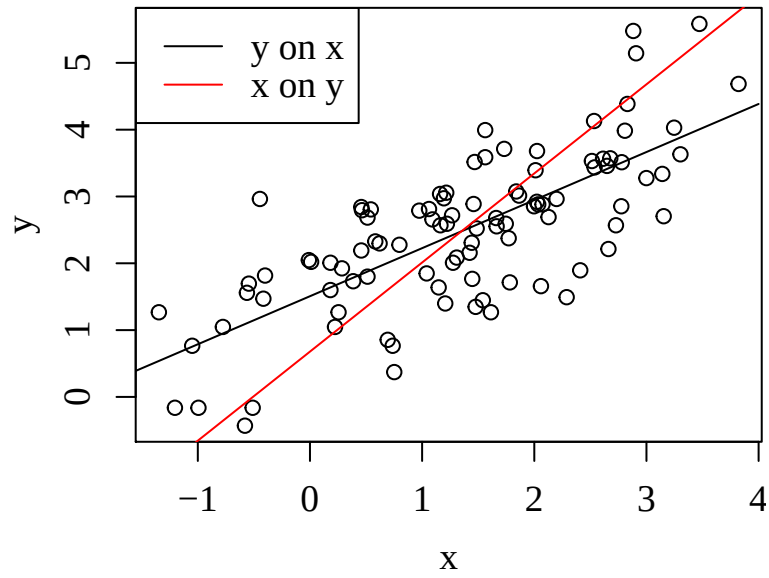
The RSE $\hat{\sigma}$ tends to decrease as more predictors are added but with diminishing returns. It may also increase slightly as the degrees of freedom (d.f.) decrease. The R^2 index monotonically increases, but also with diminishing returns. The adjusted R^2, accounting for d.f., would exhibit a small dip after the 6th predictor seen in the RSE plot.

Exercise 5: Swapping the response and predictor variable in simple linear regression. Denote `lpsa` as the y variable and `lcavol` as the x variable and fit two models.

```
df <- prostate
lmod_yx <- lm(lpsa ~ lcavol, df)
lmod_xy <- lm(lcavol ~ lpsa, df)
```

Create predictions on new data for both models and plot.

```
x_new <- data.frame(lcavol = seq(-2, 4, .1))
y_new <- data.frame(lpsa = seq(-1, 6, .1))
y_pred <- predict(lmod_yx, x_new)
x_pred <- predict(lmod_xy, y_new)

plot(lpsa ~ lcavol, df, xlab = 'x', ylab = 'y')
lines(x_new$lcavol, y_pred, col = 1)
lines(x_pred, y_new$lpsa, col = 2)
legend('topleft', legend = c('y on x', 'x on y'), col = c(1, 2), lty=1)
```



The two lines intersect at the mean of `lcavol` and `lpsa`, which is the point $(x, y) = (1.35001, 2.47839)$, because in simple linear regression, the fit must pass through the point $(\overline{x}, \overline{y})$.

Comment. The two fit lines are not the same in general because the errors they minimize are different. Regressing y on x minimizes the (sum of square) errors in the y dimension, while regressing x and y minimizes the errors in the x dimension.

Chapter 3

Inference

```
library(faraway)
```

Distribution of $\widehat{\beta}$ assuming $\varepsilon \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. Since $\boldsymbol{A} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$ is a matrix of constants,

$$\mathbb{E}(\boldsymbol{A}\boldsymbol{y}) = \boldsymbol{A}\,\mathbb{E}\,\boldsymbol{y} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{I}\boldsymbol{\beta} = \boldsymbol{\beta}$$

and

$$\operatorname{var}(\boldsymbol{A}\boldsymbol{y}) = \boldsymbol{A}(\operatorname{var}\boldsymbol{y})\boldsymbol{A}^T = \boldsymbol{A}(\sigma^2\boldsymbol{I})\boldsymbol{A}^T = \sigma^2\boldsymbol{A}\boldsymbol{A}^T.$$

Using the fact that the transpose of the inverse is the inverse of the transpose and $(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T\boldsymbol{A}^T$,

$$\boldsymbol{A}^T = \boldsymbol{X}[(\boldsymbol{X}^T\boldsymbol{X})^T]^{-1} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} \qquad \Rightarrow \qquad \boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{I}(\boldsymbol{X}^T\boldsymbol{X})^{-1} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}.$$

Finally since $\widehat{\boldsymbol{\beta}}$ is a linear combination of normal r.v. (from the normal errors assumption), it follows $\widehat{\boldsymbol{\beta}}$ is normal with mean and variance given above.

3.1 Hypothesis Tests to Compare Models

3.2 Testing Examples

Testing a subspace: equality of coefficients. Consider the book's example of testing $H_0 : \beta_1 = \beta_2 \equiv \gamma$ with full model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \varepsilon$$

and reduced model

$$y = \beta_0 + \gamma x_1 + \gamma x_2 + \beta_3 x_3 + \cdots + \varepsilon.$$

Since the relevant terms of the reduced model factor into $\gamma(x_1 + x_2)$, the reduced model in R is specified as in the book: `I(x1 + x2)`. This approach generalizes to testing equality of k coefficients $H_0 : \beta_1 = \beta_2 = \cdots = \beta_k$.

Testing a subspace: simultaneously testing multiple offsets. Consider an extension of the book where we want to test $H_0 : \beta_1 = 0.5 \cap \beta_2 = -0.5$. This is done in the nested model formalism in R as follows.

```
lmod <- lm(Species ~ Area + Elevation + Nearest + Scruz + Adjacent, gala)
lmods <- lm(
  Species ~ (
    Area + offset(0.5*Elevation) + Nearest + offset(-.5*Scruz) + Adjacent
  ),
  gala,
)

anova(lmods, lmod)
```

```
## Analysis of Variance Table
##
## Model 1: Species ~ (Area + offset(0.5 * Elevation) + Nearest + offset(-0.5 *
##     Scruz) + Adjacent)
## Model 2: Species ~ Area + Elevation + Nearest + Scruz + Adjacent
##   Res.Df    RSS Df Sum of Sq    F Pr(>F)
## 1     26 140194
## 2     24  89231  2     50963 6.85 0.0044 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The small p-value leads to rejection of H_0. We would expect a large p-value when H_0 tests whether the offsets are close to the estimates obtained in the full model; indeed this is the case.

```
lmods <- lm(
  Species ~ (
    Area + offset(0.32*Elevation) + Nearest + offset(-.24*Scruz) + Adjacent
  ),
  gala
)

anova(lmods, lmod)
```

```
## Analysis of Variance Table
##
## Model 1: Species ~ (Area + offset(0.32 * Elevation) + Nearest + offset(-0.24 *
##     Scruz) + Adjacent)
## Model 2: Species ~ Area + Elevation + Nearest + Scruz + Adjacent
##   Res.Df   RSS Df Sum of Sq  F Pr(>F)
## 1     26 89232
## 2     24 89231  2     0.378  0      1
```

3.3 Permutation Tests

A permutation test computes the distribution of a test statistic T by computing T over all possible permutations of the observed data. It is an exact test when the observations are exchangeable under H_0. In a regression setting, the test for overall association with $H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0$ uses the F-statistic. Assuming H_0 to be true, there is no relation between the predictors and response. In other words, the values of $\boldsymbol{X}$ do not matter in predicting $\boldsymbol{y}$, so exchangeability of observations holds under H_0. We can then use a permutation test to compute the distribution

of F for all possible permutations and compute a p-value of our observed (unpermuted) F. This makes no assumptions on the errors, e.g. normality, even though we are using the F-statistic in each permutation. In practice, even for a modest sized dataset, the number of permutations $n!$ is too large to compute a test statistic over all permutations. An approximate permutation test is based on a random sample of permutations to approximate the test statistic distribution, as done in the book.

Permutation test for association of a subset of predictors. Often times we are interested in tests like $H_0 : \beta_1 = 0$ or $H_0 : \beta_1 = \beta_2 = \cdots \beta_k = 0$. We can no longer permute all predictor values (or equivalently, response values) because the predictor coefficients in the full model are not assumed to be zero, hence are not exchangeable. In the example $H_0 : \beta_1 = 0$, the effect on y depends on the value of x_2 under the null, so swapping two values of x_2 violates the exchangeability assumption. However, we are justified in permuting values of x_1 because under the null, the effect on y does not depend on the value of x_1. Similarly, when testing $H_0 : \beta_1 = \beta_2 = \cdots \beta_k = 0$, we only permute the predictors being tested. F is still the test statistic of interest using the general formalism presented in the book, and we compute its distribution by computing F for each (partial) permutation.

Example: Permutation test comparing two models. We will repeat the book's permutation t-test using the general F-test formalism. The F-statistic of interest is computed using `anova()` between the full and reduced model. The observed value is then compared to the distribution of F, obtained by randomly sampling permutations of the full model and computing F for each permutation (by `anova()` with the reduced model). For comparison, we include the p-value computation using t as done in the book.

```
lmod <- lm(Species ~ Nearest + Scruz, gala)
lmod_red <- lm(Species ~ Nearest, gala)

nreps <- 4000
Fstat <- numeric(length = nreps)
tstat <- numeric(length = nreps)
set.seed(1)
for (i in 1:nreps) {
  lmod_perm <- lm(Species ~ Nearest + sample(Scruz), gala)
  Fstat[i] <- anova(lmod_red, lmod_perm)$F[2]
  tstat[i] <- summary(lmod_perm)$coef[3, 3]
}

mean(Fstat > anova(lmod_red, lmod)$F[2])
```

```
## [1] 0.27325
```

```
mean(abs(tstat) > abs(summary(lmod)$coef[3,3]))
```

```
## [1] 0.27325
```

Invoking the CLT when errors are not normal. Recall $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ is a linear combination of the y values, which are distributed as the errors. If the errors are normal, it follows $\widehat{\boldsymbol{\beta}}$ is a linear combination of r.v. and hence itself normal. If the errors are nonnormal, the exact distribution of $\widehat{\boldsymbol{\beta}}$ is not known in general. However, if the sample size is large enough, we can invoke the central limit theorem and approximate $\widehat{\boldsymbol{\beta}}$ as normal. Permutation tests avoid these distributional assumptions and can apply to small samples with nonnormal errors.

Example: Comparing permutation tests to parametric F-test. This example tests $H_0 : \beta_1 = \beta_2 = 0$ for a small dataset with lognormal errors. It also illustrates the mechanics of permuting a subset of predictors. First, generate the data with lognormal errors, fit full and reduced models, and compute a p-value from the parametric F test.

```
set.seed(1)
n <- 15
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
y <- x1 + x2 + x3 + rlnorm(n)

lmod <- lm(y ~ x1 + x2 + x3)
lmod_red <- lm(y ~ x3)
anova(lmod_red, lmod)$`Pr(>F)`[2]
```

```
## [1] 0.044526
```

Permuting instead:

```
nreps <- 4000
Fstat <- numeric(length = nreps)
for (i in 1:nreps) {
  rows <- sample(1:n)
  x1p <- x1[rows]
  x2p <- x2[rows]
  lmod_perm <- lm(y ~ x1p + x2p + x3)
  Fstat[i] <- anova(lmod_red, lmod_perm)$F[2]
}

mean(Fstat > anova(lmod_red, lmod)$F[2])
```

```
## [1] 0.04825
```

We observe fair agreement between methods even for a small sample size with lognormal errors. Of course to make any stronger conclusions, we should perform more simulations and possibly vary sample sizes and data generating processes.

3.4 Sampling

3.5 Confidence Intervals for β

Univariate confidence intervals. A $(1-\alpha)$ confidence interval for $\widehat{\beta}$ is the interval I such that $P(\widehat{\beta} \in I) = 1-\alpha$. Assuming normal errors, $\widehat{\beta} \sim \mathcal{N}(\beta, \sigma^2_{\widehat{\beta}})$. This suggests pivotal quantity $T = (\widehat{\beta} - \beta)/\widehat{\sigma}_{\widehat{\beta}}$ is t distributed with $l \equiv [n-(p+1)]$ d.f.[1] We can then construct an interval $I_T = [-t_{\alpha/2}(l), t_{\alpha/2}(l)]$ such that $P(T \in I_T) = 1-\alpha$, where $t_{\alpha/2}(l)$ is a quantile of the t distribution

[1] t distributed because the variance of the residuals σ^2, and hence $\sigma_{\widehat{\beta}}$, must be estimated with $[n-(p+1)]$ d.f.

with k d.f. The resultant confidence interval for $\widehat{\beta}$ satisfies

$$\left|\frac{\widehat{\beta}-\beta}{\widehat{\sigma}_{\widehat{\beta}}}\right| \leq t_{\alpha/2}(l),$$

which implies the usual $I = [\beta - \widehat{\sigma}_{\widehat{\beta}} t_{\alpha/2}, \beta + \widehat{\sigma}_{\widehat{\beta}} t_{\alpha/2}]$. Squaring both sides of the inequality,

$$\frac{(\widehat{\beta}-\beta)^2}{\widehat{\sigma}^2_{\widehat{\beta}}} \leq t^2_{\alpha/2}(l).$$

The RHS is closely related to an F distribution quantile, since the square of a $T(l)$ distribution is distributed as $F(1, l)$. This form is useful to motivate the multivariate analog below.

Multivariate confidence regions. The multivariate analog of a confidence interval for parameters $\widehat{\boldsymbol{\beta}} = (\widehat{\beta}_1, \widehat{\beta}_2, \ldots, \widehat{\beta}_k)$ is a region $R \in \mathbb{R}^k$ such that $P(\widehat{\boldsymbol{\beta}} \in R) = 1 - \alpha$. Assuming normal errors, $\widehat{\boldsymbol{\beta}}$ is multivariate normal: $\widehat{\boldsymbol{\beta}} \sim \mathcal{N}(\boldsymbol{\beta}, (\boldsymbol{X}^T\boldsymbol{X})^{-1}\sigma^2) \equiv \mathcal{N}(\boldsymbol{\beta}, \boldsymbol{\Sigma}^2)$. Roughly speaking, scalar norms are replaced with moment matrices and scalar division is replaced with matrix inversion. The analog of the univariate equality that R satisfies is then

$$(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta})^T (\boldsymbol{\Sigma}^2)^{-1} (\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \leq F^2_\alpha,$$

which is the book's expression after rearrangement, estimation of σ^2 with $\widehat{\sigma^2}$, and scaling by the constant p.

3.6 Bootstrap Confidence Intervals

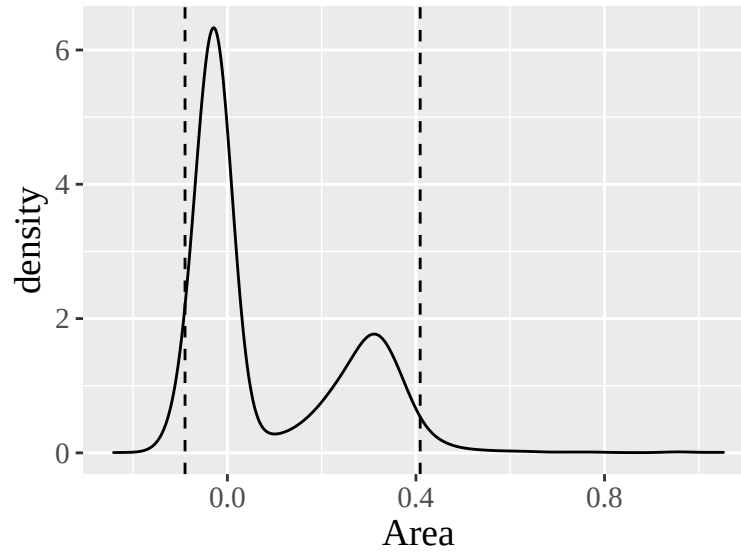
Resampling $(\boldsymbol{x}, \boldsymbol{y})$ pairs. An arguably simpler approach resamples (with replacement) directly from the data. This method is more assumption free since the predictors $\boldsymbol{x}$ are also resampled, hence assumed to be random. The book states this approach is less attractive since the predictors are assumed to be fixed in the regression model. If the model is correctly specified, the book's intervals have better coverage (but, of course, we never know if the model is correctly specified). As an example, we repeat the book's bootstrap example but resample $(\boldsymbol{x}, \boldsymbol{y})$ pairs.

```
df <- gala
lmod <- lm(Species ~ . - Endemics, df)
nboot <- 4000
coefmat <- matrix(NA, nboot, 6)

set.seed(1)
for (i in 1:nboot) {
  boot_rows <- sample(1:nrow(df), replace = TRUE)
  lmod_boot <- update(lmod, subset = boot_rows)
  coefmat[i, ] <- lmod_boot$coefficients
}

colnames(coefmat) <- c("Intercept", colnames(df[, 3:7]))
coefmat <- data.frame(coefmat)
apply(coefmat, 2, function(x) quantile(x, c(0.025,0.975)))
```

```
##       Intercept      Area Elevation Nearest    Scruz  Adjacent
## 2.5%    -30.023 -0.089849  0.049993 -4.6914 -0.62671 -0.144834
## 97.5%    35.671  0.408688  0.506284  1.7331  0.34739  0.012593
```

Most intervals qualitatively match the parametric and book's bootstrap, but `Area` has a substantially wider upper tail. Inspecting the resultant distribution shows the distribution is bimodal, unlike the book's approach in Figure 3.3.

```
library(ggplot2)
ggplot(coefmat, aes(x = Area)) +
  geom_density() +
  geom_vline(xintercept = c(-0.0898, 0.409), lty = 2)
```

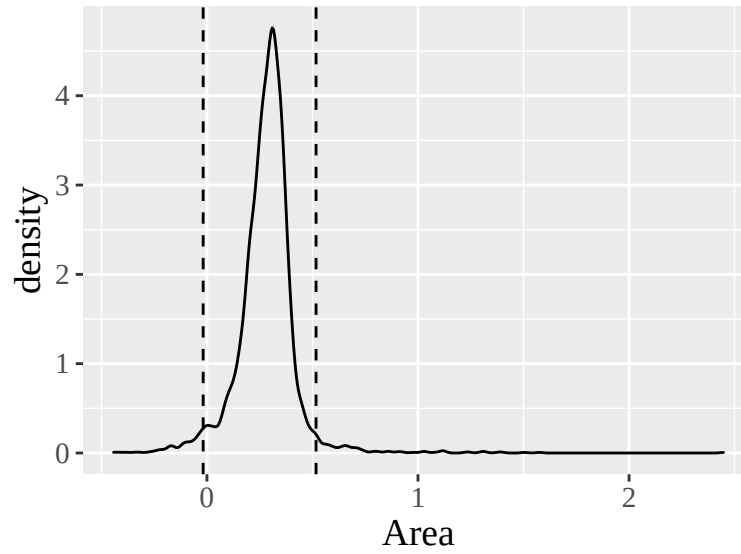


The second mode is caused by a point with high leverage that can be resampled multiply or not at all when bootstrapping:

```
sort(gala$Area, decreasing = TRUE)[1:8]
```

```
## [1] 4669.32  903.82  634.49  572.33  551.62  170.92  129.49   59.56
```

Removing it results in a unimodal distribution that is biased heavily, since none of the bootstrapped models contain it. This illustrates a limitation of this method, particularly for small samples with outliers or high leverage points.



Exercises

```
library(faraway)
library(ellipse)
```

Exercise 1: Inference using prostate cancer data. Fit `lpsa` using all other variables.

```
df <- prostate
lmod <- lm(lpsa ~ ., df)
```

(a) 90% and 95% CIs for age are below.

```
confint(lmod, level = .90)['age', ]
```

```
##         5 %        95 %
## -0.0382102 -0.0010642
```

```
confint(lmod, level = .95)['age', ]
```

```
##      2.5 %     97.5 %
## -0.0418406  0.0025663
```

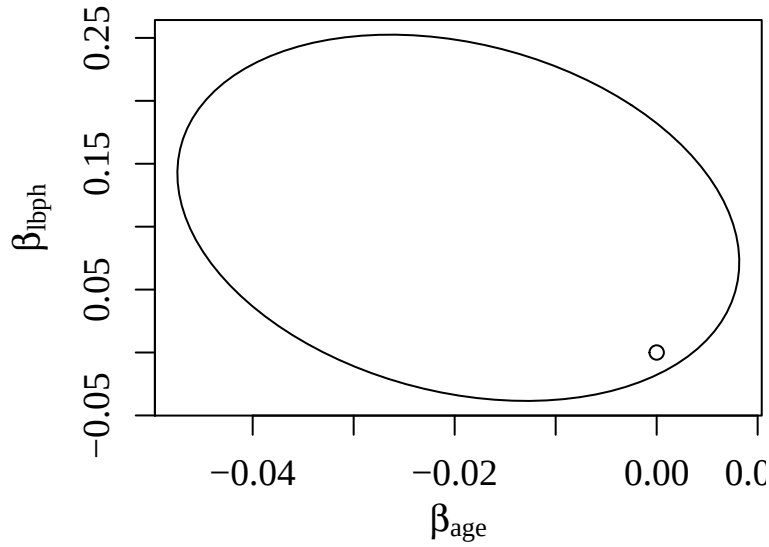
Since zero is not in the 90% CI, we can reject $\beta_{\texttt{age}} = 0$ at the $\alpha = 0.1$ level. However since zero is in the 95% CI, we can't reject $\beta_{\texttt{age}} = 0$ at the $\alpha = 0.05$ level. This implies $0.01 < p < 0.05$, verified below.

```
summary(lmod)$coefficients['age', 'Pr(>|t|)']
```

```
## [1] 0.082293
```

(b) The joint 95% confidence region for $\beta_{\texttt{age}}$ and $\beta_{\texttt{lbph}}$ is plotted below.

```
plot(
  ellipse(lmod, c('age', 'lbph')),
  type = 'l',
  xlab=expr(beta[age]),
  ylab = expr(beta[lbph])
)
points(0, 0)
```



The origin represents the joint test $H_0 : \beta_{\texttt{age}} = \beta_{\texttt{lbph}} = 0$. As the origin is contained in the region, we cannot reject H_0 at the $\alpha = 0.05$ level.

(c) The permutation test results in a p-value very similar to the parametric case in (a).

```
lmod_red <- lm(lpsa ~ . - age, df)
nperm <- 4000
Fstat <- numeric(length = nperm)

set.seed(1)
for (i in 1:nperm) {
  lmod_perm <- lm(lpsa ~ . - age + sample(age), df)
  Fstat[i] <- anova(lmod_red, lmod_perm)$F[2]
}

mean(Fstat > anova(lmod_red, lmod)$F[2])
```

```
## [1] 0.08525
```

(d) `age`, `lbph`, `lcp`, `gleason`, and `pgg45` are not significant at the $\alpha = 0.05$ level when tested individually (easily checked from `summary()` output). This suggests testing the joint five predictor hypothesis $H_0 : \beta_{\texttt{age}} = \beta_{\texttt{lbph}} = \beta_{\texttt{lcp}} = \beta_{\texttt{gleason}} = \beta_{\texttt{pgg45}} = 0$.

```
lmod_red <- lm(lpsa ~ lcavol + lweight + svi, df)
anova(lmod_red, lmod)
```

```
## Analysis of Variance Table
##
## Model 1: lpsa ~ lcavol + lweight + svi
## Model 2: lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason +
##     pgg45
##   Res.Df  RSS Df Sum of Sq    F Pr(>F)
## 1     93 47.8
## 2     88 44.2  5      3.62 1.44   0.22
```

As the p-value is relatively large, we do not have enough evidence to reject H_0 and prefer the reduced model for simplicity.

Exercise 2: Inference with changes of scale. **(a)** `H2S` and `Lactic` are statistically significant at the $\alpha = 0.05$ level using the default data.

```
df <- cheddar
lmod <- lm(taste ~ ., df)
sumary(lmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -28.877     19.735   -1.46   0.1554
## Acetic         0.328      4.460    0.07   0.9420
## H2S            3.912      1.248    3.13   0.0042
## Lactic        19.671      8.629    2.28   0.0311
##
## n = 30, p = 4, Residual SE = 10.131, R-Squared = 0.65
```

(b) Exponentiating `Acetic` and `H2S` so all variables are on their original scale,

```
df <- cheddar
lmod_org <- lm(taste ~ exp(Acetic) + exp(H2S) + Lactic, df)
sumary(lmod_org)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.90e+01   1.13e+01   -1.68    0.104
## exp(Acetic)  1.89e-02   1.56e-02    1.21    0.237
## exp(H2S)     7.67e-04   4.19e-04    1.83    0.079
## Lactic       2.50e+01   9.06e+00    2.76    0.010
##
## n = 30, p = 4, Residual SE = 11.187, R-Squared = 0.58
```

we observe only `Lactic` is statistically significant (at $\alpha = 0.05$).

(c) We can't use an F-test to compare the two models because they are not nested. Based on RSE (smaller) and R^2 (larger), the log scale model is preferable.

(d) For every unit increase in log(`H2S`), `taste` increases by the `H2S` coefficient in (a), say β. Since this is a nonlinear function of `H2S`, the change in `taste` after modifying `H2S` by 1 will depend on the value of `H2S`, say x. Specifically, the log changes by $[\log(x+1) - \log(x)]$, so `taste` would change by $\Delta(\texttt{taste}) = \beta[\log(x+1) - \log(x)]$. The same logic applies if 1 is replaced with 0.01.

(e) $f = [e^{\log(x)+0.01} - x]/x = e^{0.01} - 1$, a ratio independent of x.

Exercise 6: Comparing parametric, permutation, and bootstrap tests.

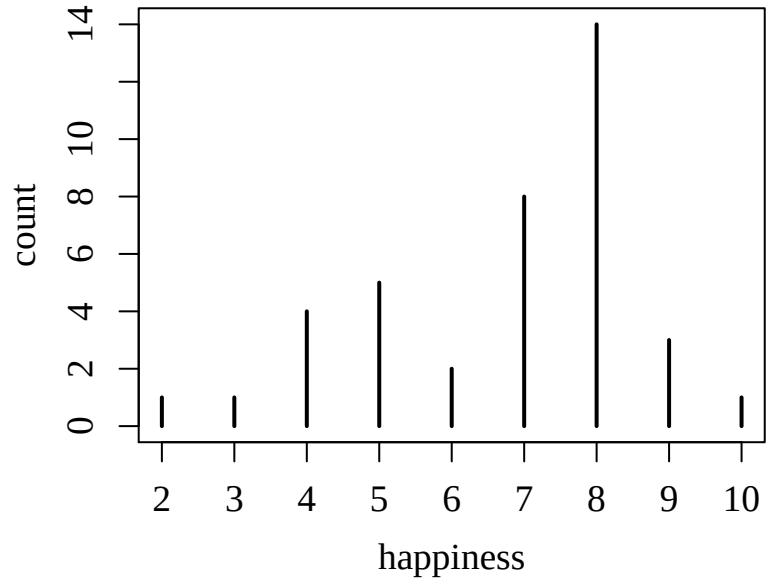
```
df <- happy
lmod <- lm(happy ~ ., df)
sumary(lmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.07208    0.85254   -0.08    0.933
## money        0.00958    0.00521    1.84    0.075
## sex         -0.14901    0.41853   -0.36    0.724
## love         1.91928    0.29545    6.50    2e-07
## work         0.47608    0.19939    2.39    0.023
##
## n = 39, p = 5, Residual SE = 1.058, R-Squared = 0.71
```

(a) `love` is the only statistically significant predictor at the 1% level.

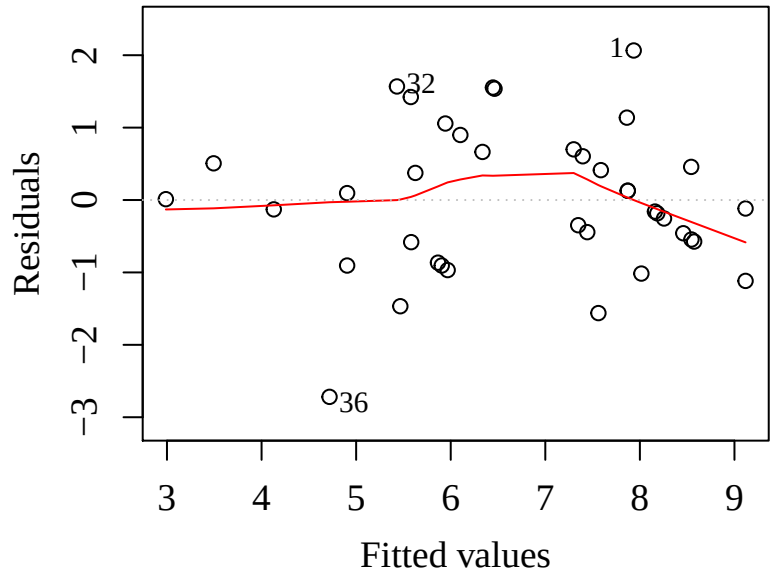
(b) The distribution of `happy` is below.

```
plot(table(df$happy), ylab = 'count', xlab = 'happiness')
```



While the response is not normally distributed, we just need the residuals (or equivalently, response conditioned on predictors) to be normally distributed to (optimally) apply a t-test. This assumption is more reasonable, evidenced in a residual plot.

```
plot(lmod, which = 1)
```



(c) A permutation test yields a similar p-value for testing $H_0 : \beta_{\texttt{money}} = 0$, which is not too surprising considering the distribution of errors.

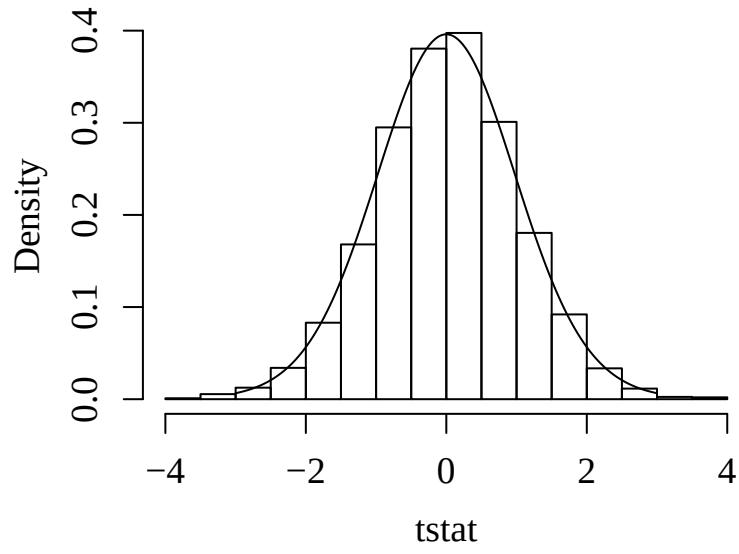
```
lmod_red <- lm(happy ~ . - money, df)
nperm <- 4000
tstat <- numeric(length = nperm)

set.seed(1)
for (i in 1:nperm) {
```

```
  lmod_perm <- lm(happy ~ . - money + sample(money), df)
  tstat[i] <- summary(lmod_perm)$coefficients['sample(money)', 't value']
}

mean(abs(tstat) > abs(summary(lmod)$coefficients['money', 't value']))
```

```
## [1] 0.07175
```

(d, e) The permuted density closely resembles the theoretical t-distribution.

```
hist(tstat, breaks = 20, freq = FALSE, main = '')  # permuted density

tgrid <- seq(-3, 3, .01)
f <- dt(tgrid, df = lmod$df.residual)
lines(tgrid, f)  # theoretical density
```



(f) Neither interval contains zero, which is inconsistent with the parametric and permutation tests. The 95% interval barely exludes zero, suggesting the p-value is just under 0.05.

```
# Fit model and extract residuals
resid <- lmod$residuals
fit_vals <- lmod$fitted.values

nboot <- 4000
beta <- numeric(length = nboot)
set.seed(1)
for (i in 1:nboot) {
  yboot <- fit_vals + sample(resid, replace = TRUE)
  lmod_boot <- update(lmod, yboot ~ .)
  beta[i] <- lmod_boot$coefficients[['money']]
}

quantile(beta, c(.05, .9))  # 90% CI
```

```
##       5%      90%
## 0.001616 0.015780
```

```
quantile(beta, c(.025, .975))  # 95% CI
```

```
##       2.5%      97.5%
## 2.7151e-05 1.9402e-02
```

Exercise 7: Complex parameter hypotheses.

(a, b) No individual predictor is significant at the 5% level. However collectively, the four predictors are significant at the 5% level, judged by the F-test.

```
df <- punting
lmod <- lm(Distance ~ RStr + LStr + RFlex + LFlex, df)
summary(lmod)
```

```
##
## Call:
## lm(formula = Distance ~ RStr + LStr + RFlex + LFlex, data = df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -23.94  -8.96  -4.44  13.52  17.02
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -79.624     65.594   -1.21     0.26
## RStr           0.512      0.486    1.05     0.32
## LStr          -0.186      0.513   -0.36     0.73
## RFlex          2.375      1.437    1.65     0.14
## LFlex         -0.528      0.826   -0.64     0.54
##
## Residual standard error: 16.3 on 8 degrees of freedom
## Multiple R-squared:  0.736,Adjusted R-squared:  0.605
## F-statistic: 5.59 on 4 and 8 DF,  p-value: 0.019
```

(c) We want to test $H_0 : \beta_{\texttt{RStr}} = \beta_{\texttt{LStr}}$. The nested model is then

```
lmod_nest <- lm(Distance ~ I(LStr + RStr) + RFlex + LFlex, df)
```

and an F-test via `anova()` yields

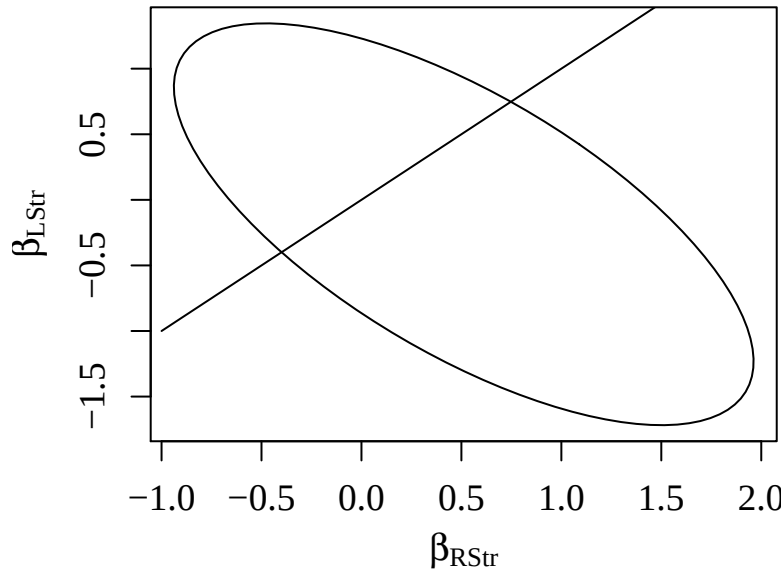
```
anova(lmod_nest, lmod)
```

```
## Analysis of Variance Table
##
## Model 1: Distance ~ I(LStr + RStr) + RFlex + LFlex
## Model 2: Distance ~ RStr + LStr + RFlex + LFlex
##   Res.Df  RSS Df Sum of Sq    F Pr(>F)
## 1      9 2287
## 2      8 2133  1       155 0.58   0.47
```

We can't reject H_0 in favor of the alternative right and left leg strengths have different effects.

(d) A 95% confidence region for $(\beta_{\texttt{RStr}}, \beta_{\texttt{LStr}})$ is below with the line $\beta_{\texttt{LStr}} = \beta_{\texttt{RStr}}$ overlaid, representing the hypothesis in (c).

```
plot(
  ellipse(lmod, c('RStr', 'LStr')),
  type = 'l',
  xlab=expr(beta[RStr]),
  ylab = expr(beta[LStr])
)

xgrid <- seq(-1, 2, .01)
lines(xgrid, xgrid)
```



Because the line is (partially) contained in the ellipse, we would not be able to reject H_0, in agreement with (c).

(e) Testing the importance of the sum of leg strengths is the same as testing $\beta_{\texttt{LStr}} = \beta_{\texttt{RStr}}$.

```
lmod2 <- lm(Distance ~ RStr + LStr, df)
lmod2_nest <- lm(Distance ~ I(RStr + LStr), df)
anova(lmod2_nest, lmod2)
```

```
## Analysis of Variance Table
##
## Model 1: Distance ~ I(RStr + LStr)
## Model 2: Distance ~ RStr + LStr
##   Res.Df  RSS Df Sum of Sq   F Pr(>F)
## 1     11 3061
## 2     10 2973  1      88.3 0.3    0.6
```

We again fail to reject H_0.

(f) Similar to (c), we can't reject $H_0 : \beta_{\texttt{RFlex}} = \beta_{\texttt{LFlex}}$.

```
lmod_nest <- lm(Distance ~ LStr + RStr + I(RFlex + LFlex), df)
anova(lmod_nest, lmod)
```

```
## Analysis of Variance Table
##
## Model 1: Distance ~ LStr + RStr + I(RFlex + LFlex)
```

```
## Model 2: Distance ~ RStr + LStr + RFlex + LFlex
##   Res.Df  RSS Df Sum of Sq    F Pr(>F)
## 1      9 2648
## 2      8 2133  1       516 1.93    0.2
```

(g) We want to test $H_0 : (\beta_{\texttt{RStr}} = \beta_{\texttt{LStr}}) \cap (\beta_{\texttt{RFlex}} = \beta_{\texttt{LFlex}})$. This requires nested model

```
lmod_nest <- lm(Distance ~ I(LStr + RStr) + I(RFlex + LFlex), df)
```

Performing the F-test, we again can't reject H_0.

```
anova(lmod_nest, lmod)
```

```
## Analysis of Variance Table
##
## Model 1: Distance ~ I(LStr + RStr) + I(RFlex + LFlex)
## Model 2: Distance ~ RStr + LStr + RFlex + LFlex
##   Res.Df  RSS Df Sum of Sq    F Pr(>F)
## 1     10 2799
## 2      8 2133  2       666 1.25   0.34
```

(g) Fitting `Hang` instead of `Distance`,

```
lmod <- lm(Hang ~ RStr + LStr + RFlex + LFlex, df)
summary(lmod)
```

```
##
## Call:
## lm(formula = Hang ~ RStr + LStr + RFlex + LFlex, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3630 -0.1353 -0.0785  0.0994  0.3589
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.22524    1.03278   -0.22     0.83
## RStr         0.00515    0.00765    0.67     0.52
## LStr         0.00770    0.00808    0.95     0.37
## RFlex        0.01940    0.02263    0.86     0.42
## LFlex        0.00461    0.01300    0.35     0.73
##
## Residual standard error: 0.257 on 8 degrees of freedom
## Multiple R-squared:  0.816,Adjusted R-squared:  0.723
## F-statistic: 8.85 on 4 and 8 DF,  p-value: 0.00492
```

we again observe no individual significance but overall significance using the F-test. We cannot compare this model to the one used in (a) because neither is a subset of the other.

Chapter 4

Prediction

```
library(faraway)
```

4.1 Confidence Intervals for Predictions

Bootstrap prediction intervals. A bootstrap prediction interval (PI) for mean response $\hat{y}_0$ of a future value $\boldsymbol{x}_0$ is a straightforward extension of parameter confidence intervals discussed in Section 3.6, since $\hat{y}_0 = \boldsymbol{x}_0^T \boldsymbol{\beta}$ is just a linear combination of parameters and easily estimated for each resample using the `predict()` function. As an example, the following produces bootstrap intervals for mean response of a 'median person'.

```
lmod <- lm(
  brozek ~ (
    age + weight + height + neck + chest + abdom + hip + thigh + knee
    + ankle + biceps + forearm + wrist
  ),
  fat
)
resids <- lmod$residuals
fit_vals <- lmod$fitted.values
x0 <- apply(model.matrix(lmod), 2, median)  # Future observation.

# Bootstrap resampling for the mean of a future observation.
set.seed(1)
nboot <- 4000
pred_vals <- numeric(length = nboot)
for (i in 1:nboot) {
  yboot <- fit_vals + sample(resids, replace = TRUE)
  lmod_boot <- update(lmod, yboot ~ .)
  pred_vals[i] <- predict(lmod_boot, newdata = data.frame(t(x0)))
}

quantile(pred_vals, c(.025, .975))  # 95% CI.
```

```
##   2.5%  97.5%
## 16.950 18.025
```

The interval is similar to the parametric one assuming normal errors:

```
predict(
  lmod, newdata = data.frame(t(x0)), interval = 'confidence', level = .95
)
```

```
##      fit    lwr    upr
## 1 17.493 16.944 18.042
```

To construct bootstrap prediction intervals for a single future observation, we must estimate $\hat{y}_0 + \varepsilon$ for each resample, which is no longer a linear combination of estimated parameters. A simple modification in the spirit of the bootstrap estimates ε by randomly selecting one of the residuals, so the updated code looks like this:

```
# Bootstrap resampling for a single future observation.
set.seed(1)
nboot <- 4000
pred_vals <- numeric(length = nboot)
for (i in 1:nboot) {
  yboot <- fit_vals + sample(resids, replace = TRUE)
  lmod_boot <- update(lmod, yboot ~ .)
  pred_vals[i] <- (
    predict(lmod_boot, newdata = data.frame(t(x0))) + sample(resids, size = 1)
  )
}

quantile(pred_vals, c(.025, .975))  # 95% PI.
```

```
##    2.5%   97.5%
##  9.7181 24.8483
```

which again is similar to the parametric PI assuming normal errors:

```
predict(lmod, newdata = data.frame(t(x0)), interval = 'prediction', level = .95)
```

```
##      fit    lwr    upr
## 1 17.493 9.6178 25.369
```

More sophisticated methods exist including using variance-adjusted residuals (see Cross Validated) or order statistics (see Wikipedia).

4.2 Predicting Body Fat

4.3 Autoregression

4.4 What Can Go Wrong with Predictions?

Exercises

```
library(faraway)
```

Exercise 1: Prediction between two different patients.

```
df <- prostate
df$svi <- factor(df$svi)
df$gleason <- factor(df$gleason)
lmod <- lm(lpsa ~ ., df)
```

(a) A new patient with predictors `x0` will have the following 95% prediction interval.

```
x0 <- data.frame(
  intercept = 1, lcavol = 1.44692, lweight = 3.62301, age = 65, lbph = .3001,
  svi = factor(0, levels = levels(df$svi)), lcp = -.79851,
  gleason = factor(7, levels = levels(df$gleason)), pgg45 = 15
)
predict(lmod, newdata = x0, interval = 'prediction', level = 0.95)
```

```
##     fit    lwr    upr
## 1 2.495 1.0698 3.9202
```

(b) Predicting on a younger patient with the same characteristics, we observe a wider interval.

```
x0$age <- 20
predict(lmod, newdata = x0, interval = 'prediction', level = 0.95)
```

```
##      fit    lwr    upr
## 1 3.4737 1.7292 5.2183
```

This is because a 20 year old patient is 'far' from a typical patient. In fact, it is a severe extrapolation evidenced by a numerical summary of ages.

```
summary(df$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    41.0    60.0    65.0    63.9    68.0    79.0
```

(c) Only `lcavol`, `lweight`, and `svi` are significant at the 5% level.

```
lmod_red <- update(lmod, . ~ lcavol + lweight + svi)

x0 <- data.frame(
  intercept = 1, lcavol = 1.44692, lweight = 3.62301,
  svi = factor(0, levels = levels(df$svi))
)
predict(lmod_red, newdata = x0, interval = 'prediction', level = 0.95)
```

```
##      fit     lwr    upr
## 1 2.3725 0.93834 3.8067
```

The resultant interval is narrower than (b), partially because age is no longer used in the model. The interval is also quite similar to (a), suggesting a large amount of information has been retained in just 3 predictors. As the reduced model does not have the extrapolation problem and is less susceptible to overfitting, we prefer it for the young patient prediction that the full model used in (b).

Exercise 2: Prediction intervals with and without transformation.

```
df <- teengamb
df$sex <- factor(df$sex)  # 0 = male, 1 = female
lmod <- lm(gamble ~ ., df)
```

(a, b) Prediction intervals for a typical and extreme male are as follows.

```
x0mean <- data.frame(
  intercept = 1, sex = factor(0, levels = levels(df$sex)),
  status = mean(df$status), income = mean(df$income), verbal = mean(df$verbal)
)
predict(lmod, newdata = x0mean, interval = 'prediction', level = 0.95)
```

```
##      fit     lwr upr
## 1 28.243 -18.515  75
```

```
x0max <- data.frame(
  intercept = 1, sex = factor(0, levels = levels(df$sex)),
  status = max(df$status), income = max(df$income), verbal = max(df$verbal)
)
predict(lmod, newdata = x0max, interval = 'prediction', level = 0.95)
```

```
##      fit    lwr    upr
## 1 71.308 17.066 125.55
```

As expected, the extreme prediction has larger intervals.

(c) Repeating the typical prediction for a square-root transformed model (but transformed back to original units after prediction):

```
lmod2 <- lm(sqrt(gamble) ~ ., df)
predict(lmod2, newdata = x0mean, interval = 'prediction', level = 0.95)**2
```

```
##      fit      lwr    upr
## 1 16.399 0.060042 69.624
```

we observe a narrower interval and a different point estimate. One major difference is that the response in original units cannot be negative.

(d) Predictions for a female using the transformed model yield nonsensical results when transformed to the original units.

```
x0fem <- data.frame(
  intercept = 1, sex = factor(1, levels = levels(df$sex)),
  status = 20, income = 1, verbal = 10
)
predict(lmod2, newdata = x0fem, interval = 'prediction', level = 0.95)**2
```

```
##      fit    lwr    upr
## 1 4.3534 47.732 7.4852
```

Examination of a numerical summary via `summary(df[df$sex == 1, ])` shows `x0fem` is an extreme set of predictor values.

Exercise 3: Comparing regression model to contigency table for a designed experiment.

(a) `snail` comes from a designed experiments where 4 measurements are made on each of $2 \times 3 = 6$ different `temp`/`humid` configurations. By default, `xtabs(water ~ temp + humid, snail)` produces a 2×3 table, where each cell is the sum of `water` for a given configuration. Dividing by 4 yields the mean `water` value per configuration.

```
xtabs(water ~ temp + humid, snail)/4
```

```
##     humid
## temp    45    75   100
##   20 72.50 81.50 97.00
##   30 69.50 78.25 97.75
```

Since $(\texttt{temp}, \texttt{humid}) = (25, 60)$ was not one of the 6 configurations, it is difficult to extrapolate and predict `water`.

(b, c) A regression model allows us to extrapolate and predict for $(\texttt{temp}, \texttt{humid}) = (25, 60)$.

```
lmod <- lm(water ~ temp + humid, snail)
x0 <- data.frame(int = 1, temp = 25, humid = 60)
predict(lmod, newdata = x0, interval = 'conf')
```

```
##      fit    lwr    upr
## 1 76.437 73.699 79.174
```

Trying a configuration studied,

```
x0 <- data.frame(int = 1, temp = 30, humid = 75)
predict(lmod, newdata = x0, interval = 'conf')
```

```
##      fit    lwr    upr
## 1 82.622 79.288 85.957
```

we observe a 95% interval for the mean that does not contain the value from the table in (a). While the regression model allows us to extrapolate values in between (and outside), it makes more assumptions than the non-parametric summary provided by the table.

(d) The regression model predicts

$$\widehat{y} = \widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \widehat{\beta}_2 x_2.$$

Any choice of (x_1, x_2) such that $\widehat{\beta}_1 x_2 + \widehat{\beta}_2 x_2 = 0$ will yield $\widehat{y} = \widehat{\beta}_0$; the choice is not unique. Two possible choices of (x_1, x_2) are below.

```
x0 <- data.frame(int = 1, temp = 0, humid = 0)
x0 <- data.frame(
  int = 1, temp = lmod$coefficients['humid'],
  humid = -lmod$coefficients['temp']
)
predict(lmod, newdata = x0)[[1]] == lmod$coefficients['(Intercept)'][[1]]
```

```
## [1] TRUE
```

(e) The regression model prediction can be solved for

$$x_2 = \frac{1}{\widehat{\beta}_2}(\widehat{y} - \widehat{\beta}_0 - \widehat{\beta}_1 x_1).$$

```
x0 <- data.frame(
  int = 1, temp = 25,
  humid = (
    80 - lmod$coefficients[['(Intercept)']] - lmod$coefficients[['temp']]*25
  ) / lmod$coefficients[['humid']]
)
x0
```

```
##   int temp  humid
## 1   1   25 67.525
```

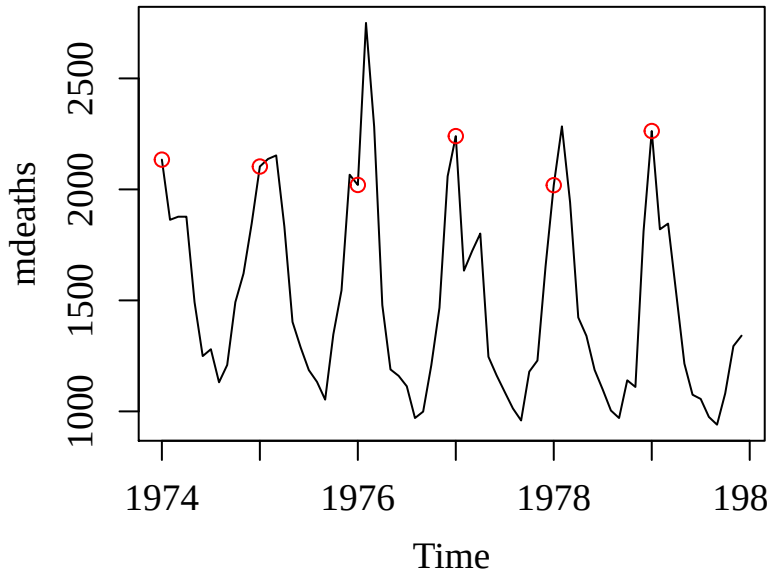
```
predict(lmod, newdata = x0)
```

```
##  1
## 80
```

Exercise 4: Time series predictions using an autoregressive model. `mdeaths` is a time-series object which indexing behavior similar to a numeric vector.

(a) A plot reveals most deaths occur near the beginning of the year.

```
plot(mdeaths)
x <- seq(1, 1+12*5, 12)
points(x%/%12 + 1974, mdeaths[x], col = 'red')  # January
```



(b) An autoregressive model is fit using previous month, year, and 13 months.

```
lagdf <- data.frame(embed(mdeaths, 14))
colnames(lagdf) <- c('y', paste0('lag', 1:13))
armod <- lm(y ~ lag1 + lag12 + lag13, lagdf)
sumary(armod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   58.199    120.736    0.48    0.632
## lag1           0.250      0.133    1.89    0.065
## lag12          0.536      0.118    4.54  3.1e-05
## lag13          0.151      0.139    1.09    0.280
```

```
##
## n = 59, p = 4, Residual SE = 238.726, R-Squared = 0.73
```

Only `lag12`, number of deaths one year ago, is significant at 5%.

(c) A prediction interval for number of deaths in January 1980 is below.

```
n <- nrow(lagdf)
x0 <- data.frame(
  lag1 = lagdf$y[n], lag12 = lagdf$lag11[n], lag13 = lagdf$lag12[n]
)
(pred <- predict(armod, x0, interval = 'pred', level = .95))
```

```
##      fit    lwr    upr
## 1 1879.6 1359.7 2399.5
```

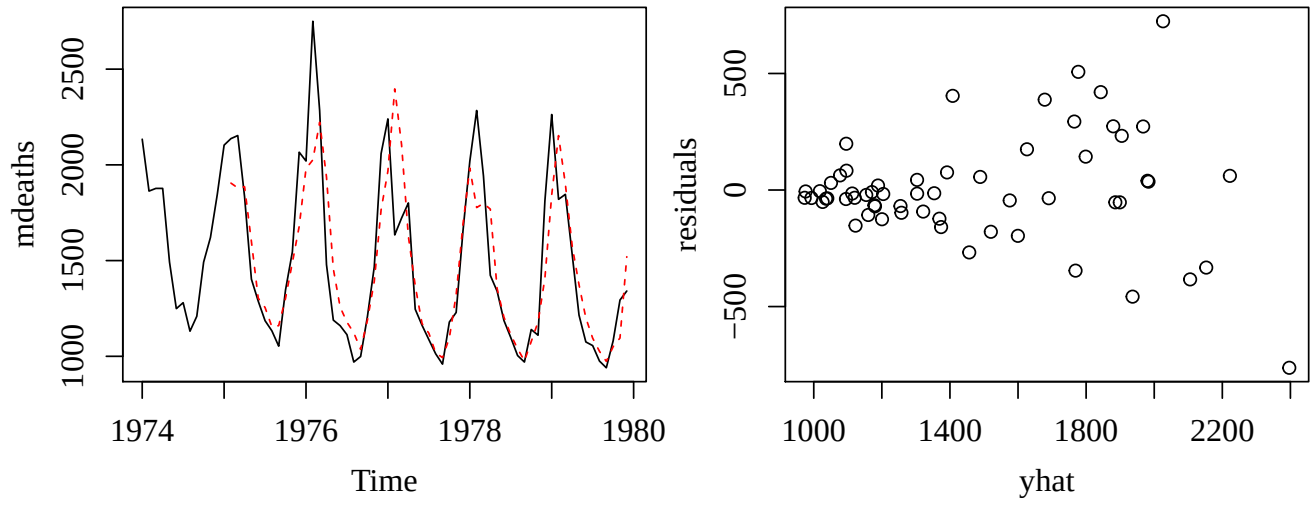
(d) Recursively iterating to produce a prediction for the next month:

```
x0 <- data.frame(
  lag1 = pred[1], lag12 = lagdf$lag10[n], lag13 = lagdf$lag11[n]
)
predict(armod, x0, interval = 'pred', level = .95)
```

```
##      fit    lwr    upr
## 1 1845.2 1345.9 2344.6
```

(e) The fit is overlayed in dotted red in the left plot.

```
yhat <- armod$fitted.values
x <- seq(13, length(mdeaths)-1, 1)
plot(mdeaths)  # Observed data
lines(x/12 + 1974, yhat, lty = 2, col = 2)  # Autoregression

plot(yhat, mdeaths[-c(1:13)] - yhat, ylab = 'residuals')  # Residual plot
```



The overall shape matches, but we notice larger deviations at death peaks near beginning of the years, as they are more variable. A residual plot on the right shows that the residual variance tends to increase with larger predictions, reducing their accuracy.

Exercise 5: Examining statistical vs. practical significance.

The full model used in the book is

```
lmod <- lm(
  brozek ~ (
    age + weight + height + neck + chest + abdom + hip + thigh + knee
    + ankle + biceps + forearm + wrist
  ),
  fat
)
```

(a) Consider a simpler reduced model comprised of easier variables to obtain:

```
lmod_red <- lm(brozek ~ age + weight + height + abdom, fat)
```

The results of an F-test

```
anova(lmod_red, lmod)
```

```
## Analysis of Variance Table
##
## Model 1: brozek ~ age + weight + height + abdom
## Model 2: brozek ~ (age + weight + height + neck + chest + abdom + hip +
##     thigh + knee + ankle + biceps + forearm + wrist)
##   Res.Df  RSS Df Sum of Sq    F Pr(>F)
## 1    247 4205
## 2    238 3785  9       420 2.93 0.0026 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

suggest that we reject the reduced model is not statistically sufficient to explain the response.

(b) Comparing predictions between the two models, however, we don't notice a practically important different in prediction intervals, despite the small p-value of the F-test.

```
x0 <- data.frame(t(apply(model.matrix(lmod), 2, median)))
predict(lmod, x0, interval = 'pred', level = .95)
```

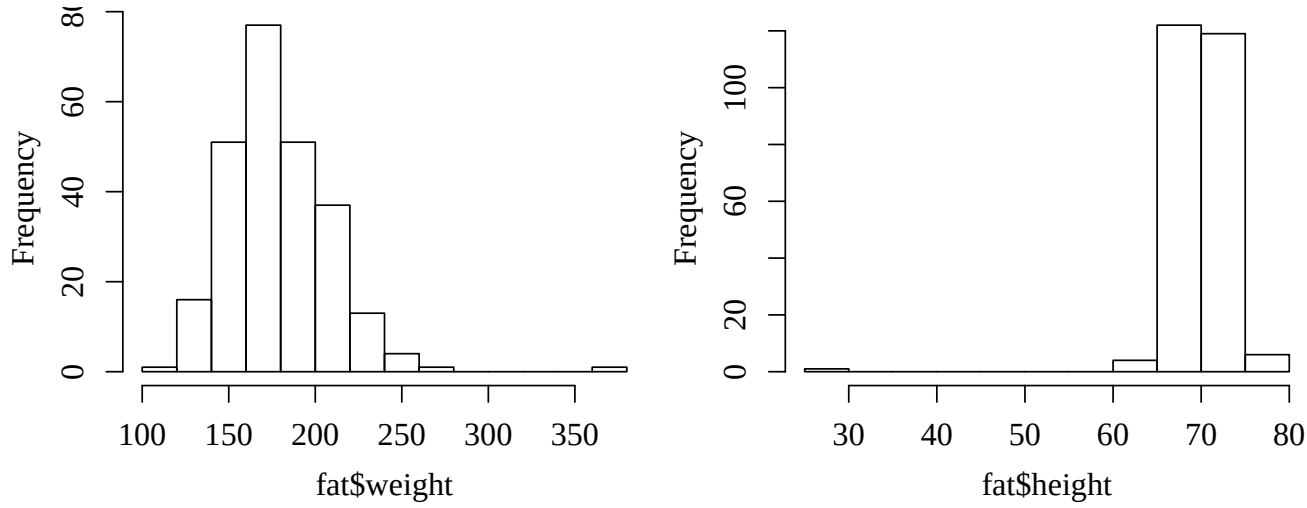
```
##      fit    lwr    upr
## 1 17.493 9.6178 25.369
```

```
predict(lmod_red, x0, interval = 'pred', level = .95)
```

```
##     fit    lwr    upr
## 1 17.84 9.6966 25.984
```

(c) Examining the distributions of height and weight are insightful to detect outliers.

```
hist(fat$weight, main = '')
hist(fat$height, main = '')
```



The weight and height outliers correspond to cases 39 and 42, respectively.

(d) Removing the two outliers in (c) and repeating (a):

```
df <- fat[-c(39, 42), ]
lmod <- lm(
  brozek ~ (
    age + weight + height + neck + chest + abdom + hip + thigh + knee
    + ankle + biceps + forearm + wrist
  ),
  df
)
lmod_red <- lm(brozek ~ age + weight + height + abdom, df)
anova(lmod_red, lmod)
```

```
## Analysis of Variance Table
##
## Model 1: brozek ~ age + weight + height + abdom
## Model 2: brozek ~ (age + weight + height + neck + chest + abdom + hip +
##     thigh + knee + ankle + biceps + forearm + wrist)
##   Res.Df  RSS Df Sum of Sq    F Pr(>F)
## 1    245 4040
## 2    236 3668  9       371 2.66  0.006 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

we would still reject the reduced model statistically. Repeating (b):

```
x0 <- data.frame(t(apply(model.matrix(lmod), 2, median)))
predict(lmod, x0, interval = 'pred', level = .95)
```

```
##      fit    lwr    upr
## 1 17.542 9.7545 25.329
```

```
predict(lmod_red, x0, interval = 'pred', level = .95)
```

```
##      fit    lwr    upr
## 1 17.903 9.8879 25.919
```

we still see no practically important difference between models. In conclusion, even after removing

outliers, the reduced model shows no practical difference despite showing a statistically significant difference to the full model.

Chapter 5

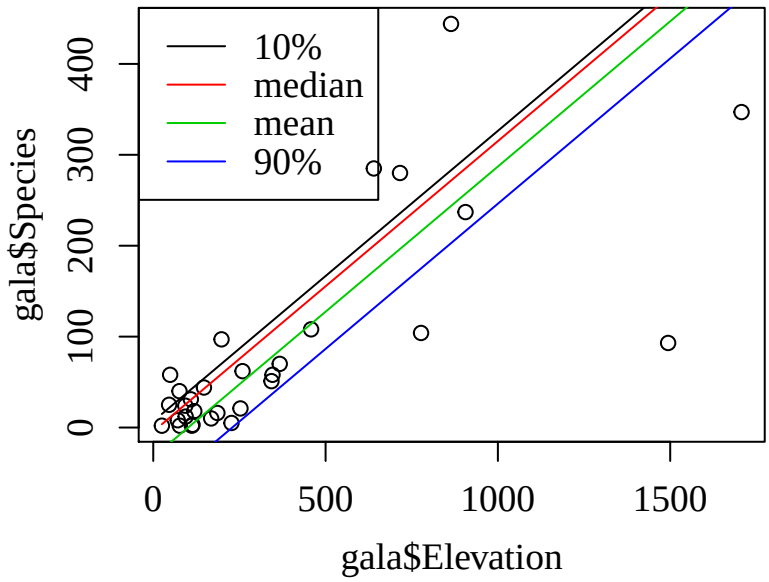
Explanation

```
library(faraway)
```

5.1 Simple Meaning

Effect plot dependence on other predictor values. For the `gala` example, the book plots the effect of `Elevation` on `Species` at the mean value of all other predictors. The following plot examines the sensitivity of the effect plot as other predictor values are used.

```
lmod <- lm(Species ~ Area + Elevation + Nearest + Scruz + Adjacent, gala)

make_data <- function(func, ...) {
  x0 <- data.frame(
    Elevation = gala$Elevation,
    Nearest = func(gala$Nearest, ...)[[1]],
    Area = func(gala$Area, ...)[[1]],
    Scruz = func(gala$Scruz, ...)[[1]],
    Adjacent = func(gala$Adjacent, ...)[[1]]
  )
  return(x0)
}

plot(gala$Elevation, gala$Species)
i <- order(gala$Elevation)
lines(gala$Elevation[i], predict(lmod, make_data(quantile, .1))[i], col = 1)
lines(gala$Elevation[i], predict(lmod, make_data(median))[i], col = 2)
lines(gala$Elevation[i], predict(lmod, make_data(mean))[i], col = 3)
lines(gala$Elevation[i], predict(lmod, make_data(quantile, .9))[i], col = 4)
legend('topleft', legend = c('10%', 'median', 'mean', '90%'), col = 1:4, lty=1)
```



The slope of the effect is unchanged, regardless of the predictor value, while the intercept is free to to shift. To see this, examine the regression prediction with effect variable x_1:

$$\widehat{y} = \widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \text{other predictors.}$$

Substituting in values of other predictors results in a constant offset to β_0. In general in the absence of interactions, modifying predictor values does not change the shape of the effect plot except by a constant offset. With interactions, the shape of the effect plot may change at different predictor values. For instance, the regression prediction could be

$$\widehat{y} = \widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \widehat{\beta}_{12} x_1 x_2 + \text{other predictors.}$$

The slope of the line is then $(\widehat{\beta}_1 + \widehat{\beta}_{12} x_2)$, which is dependent on the value of predictor x_2.

5.2 Causality

5.3 Designed Experiments

5.4 Observational Data

5.5 Matching

5.6 Covariate Adjustment

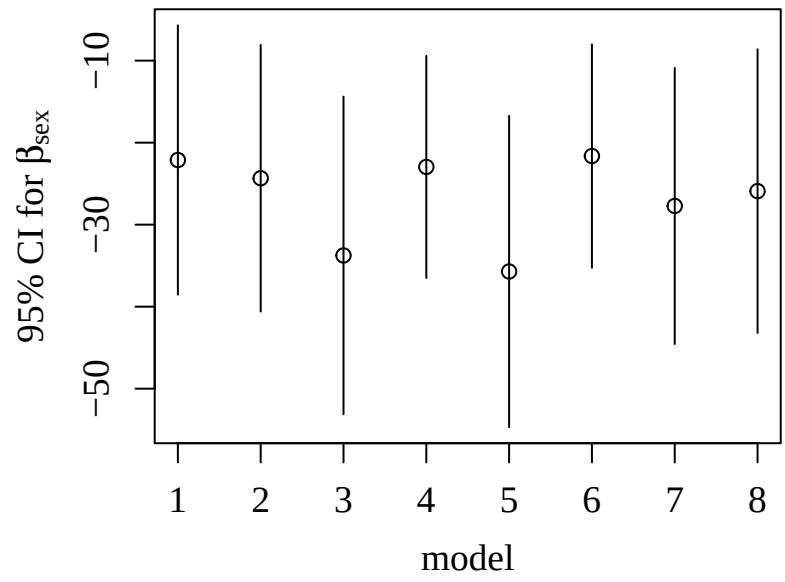
5.7 Qualitative Support for Causation

Exercises

```
library(faraway)
library(ggplot2)
library(Matching)
```

Exercise 1: Stability of effect varying models. 95% CIs for all 8 possible models with `sex` as a predictor are plotted below.

```
df <- teengamb
lmod1 <- lm(gamble ~ sex + status + income + verbal, df)
lmod2 <- lm(gamble ~ sex + status + income, df)
lmod3 <- lm(gamble ~ sex + status + verbal, df)
lmod4 <- lm(gamble ~ sex + income + verbal, df)
lmod5 <- lm(gamble ~ sex + status, df)
lmod6 <- lm(gamble ~ sex + income, df)
lmod7 <- lm(gamble ~ sex + verbal, df)
lmod8 <- lm(gamble ~ sex, df)

x <- 1:8
betas <- numeric(length = length(x))
lower <- numeric(length = length(x))
upper <- numeric(length = length(x))
for (i in x) {
  lmod <- paste0('lmod', i)
  beta <- summary(get(lmod))$coefficients['sex', 'Estimate']
  se <- summary(get(lmod))$coefficients['sex', 'Std. Error']
  betas[i] <- beta
  lower[i] <- beta - 2*se
  upper[i] <- beta + 2*se
}
plot(
  x, betas, ylim = c(min(lower), max(upper)),
  xlab = 'model', ylab = bquote(.('95% CI for' ~ beta['sex']))
)
segments(x, lower, x, upper)
```



The point estimates relative to CI widths do not vary substantially with model, suggesting the sex effect is fairly stable. Models 3 and 5 which deviate the most contain `status` but not `income`.

Exercise 2: Varying models in a designed experiment. All models will have the same point

estimate of the coefficient, but different standard errors (hence differing t-statistics and p-values). This is because the experiment has an orthogonal design (see Section 2.11 for more details). The full model has the best predictive power, judged by adjusted R^2 and RSE.

Exercise 3: Comparing matching and covariate adjustment. We are interested in the effect of `sex` on `gamble`. A simple regression with only `sex` as a predictor shows `sex` is significant.

```
df <- teengamb
sumary(lm(gamble ~ sex, df))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    29.77       5.50    5.42  2.3e-06
## sex           -25.91       8.65   -3.00   0.0044
##
## n = 47, p = 2, Residual SE = 29.094, R-Squared = 0.17
```

The estimate for $\beta_{\texttt{sex}}$ may be biased, however, due to possible confounding variables like `income`. A quick way to check this is to adjust for `income` by adding it to the model.

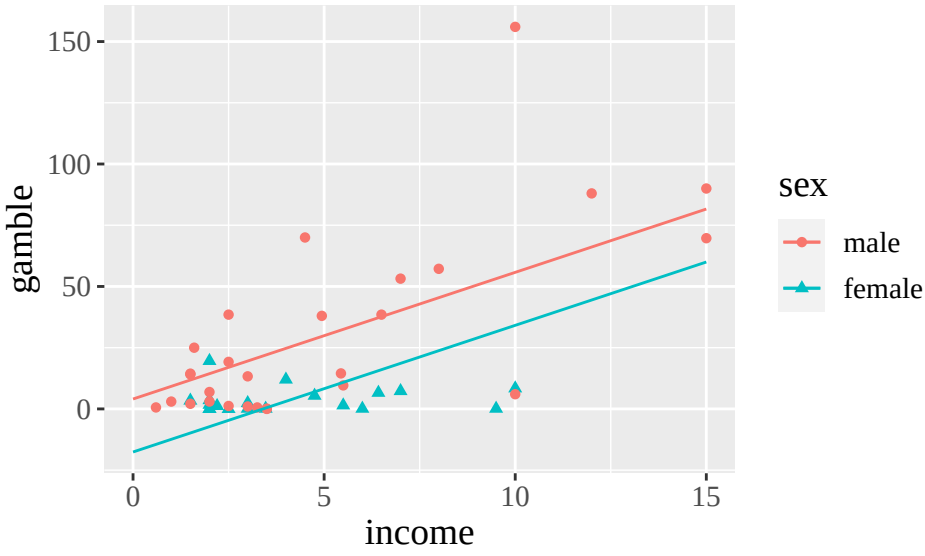
```
sumary(lm(gamble ~ sex + income, df))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.041      6.394    0.63   0.5307
## sex          -21.634      6.809   -3.18   0.0027
## income         5.172      0.951    5.44  2.2e-06
##
## n = 47, p = 3, Residual SE = 22.754, R-Squared = 0.5
```

We still observe a significant `sex` effect, suggesting the gambling behavior of males and females truly is different, even if they have the same income. We investigate this further graphically in (a)–(b). We can also deduce this by matching males and females with similar incomes, and computing their difference in `gamble`, done in (c)–(f).

(a–b) The following plots `gamble` against `income` grouped by `sex`, with regression fits overlaid.

```
df$sex <- factor(df$sex, levels = c(0, 1), labels = c('male', 'female'))
lmod <- lm(gamble ~ income + sex, df)
xmale <- data.frame(income = seq(0, 15, .1), sex = 'male')
xfemale <- data.frame(income = seq(0, 15, .1), sex = 'female')
ymale <- predict(lmod, xmale)
yfemale <- predict(lmod, xfemale)

ggplot(df, aes(x = income, y = gamble, shape = sex, color = sex)) +
  geom_point() +
  geom_line(data = cbind(xmale, ymale), aes(y = ymale)) +
  geom_line(data = cbind(xfemale, yfemale), aes(y = yfemale))
```



The vertical distance between the lines implies that on average, males gamble more than females even if they have the same income.

(c) Matching males and females with similar incomes,

```
set.seed(1)
mm <- GenMatch(
  teengamb$sex, df$income, ties = FALSE, caliper = 0.05, pop.size = 1000
)
```

we find there are

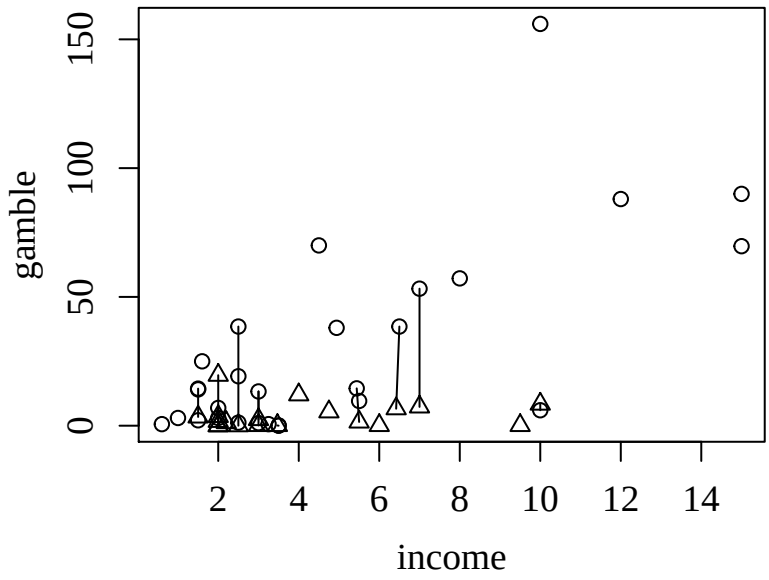
```
nrow(mm$matches)
```

```
## [1] 14
```

matches. $47 - 2 \times 14 = 19$ observations go unmatched.

(d) A plot showing the matched pairs is below.

```
plot(gamble ~ income, teengamb, pch=sex+1)
obs_female <- mm$matches[, 1]
obs_male <- mm$matches[, 2]
segments(
  df$income[obs_female], df$gamble[obs_female],
  df$income[obs_male], df$gamble[obs_male]
)
```



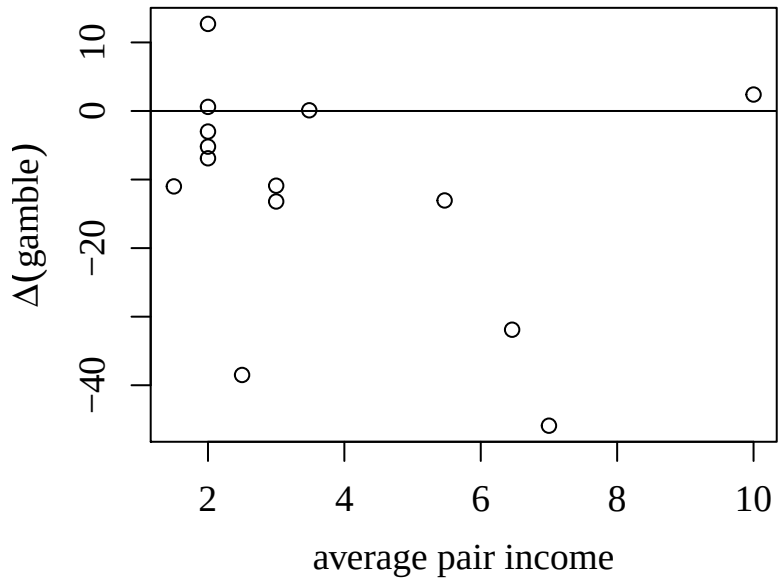
(e) A t-test on the `gamble` differences between pairs shows there is a statistically significant nonzero difference.

```
diff <- df$gamble[obs_female] - df$gamble[obs_male]
t.test(diff)
```

```
##
##  One Sample t-test
##
## data:  diff
## t = -2.66, df = 13, p-value = 0.02
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -21.2065  -2.1864
## sample estimates:
## mean of x
##   -11.696
```

(f) A qualitative plot of the differences against income shows 4/14 females gambled more than their paired males and only one female gambled substantially more.

```
plot(
  (df$income[obs_female] + df$income[obs_male]) / 2, diff,
  xlab = 'average pair income', ylab = expression(Delta(gamble))
)
abline(h = 0)
```



(g) Both the linear model and matched pair approach conclude males gamble more than females after adjusting for possible confounding by income.

Chapter 6

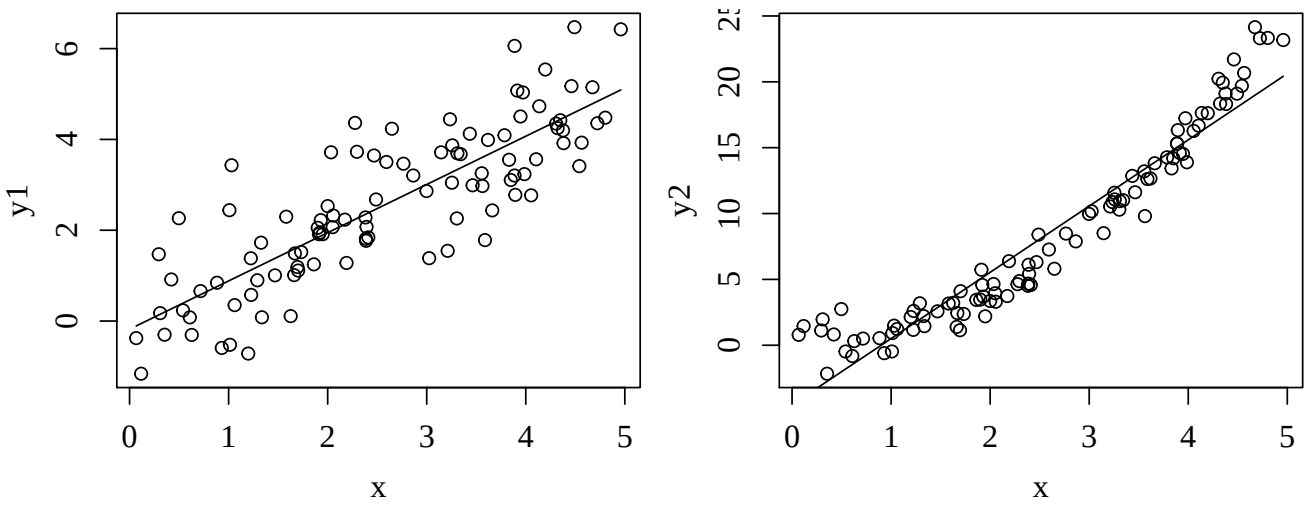
Diagnostics

```
library(faraway)
```

6.1 Checking Error Assumptions

Why residual plots against fitted values are good diagnostics. Consider simple linear regression with a single predictor x. A visual way to determine if our model is any good is to simply plot the data and overlay the regression fit. This is done for a response that is truly linear and quadratic.

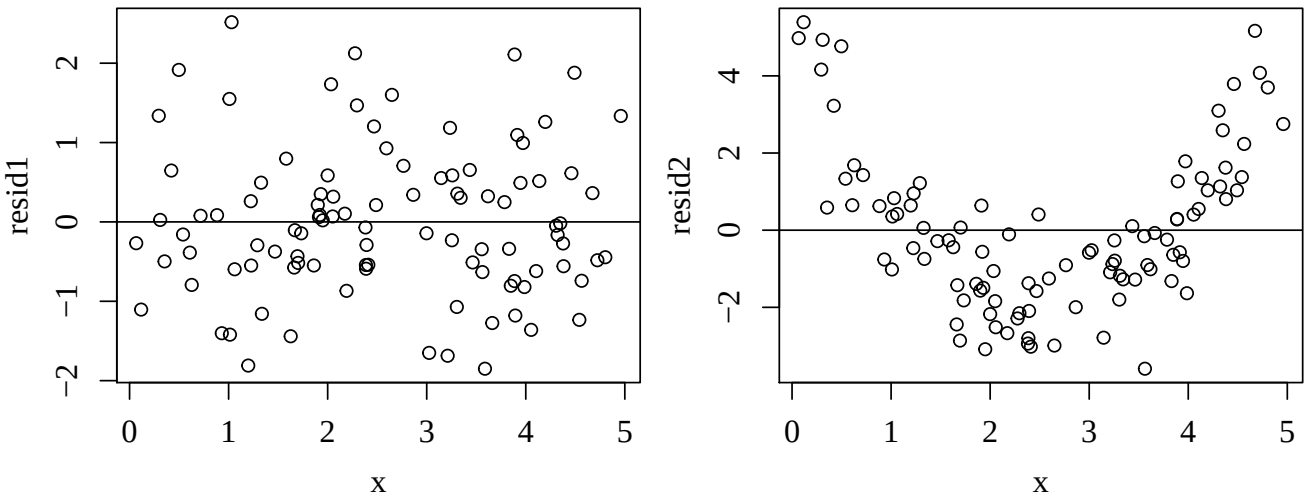
```
set.seed(1)
n <- 100
x <- runif(n, 0, 5)
y1 <- x + rnorm(n)
y2 <- x**2 + rnorm(n)
lmod1 <- lm(y1 ~ x)
lmod2 <- lm(y2 ~ x)

i <- order(x)
plot(x, y1)
lines(x[i], predict(lmod1)[i])
plot(x, y2)
lines(x[i], predict(lmod2)[i])
```



The left plot demonstrates a good fit. The data appear randomly scattered about the line, are densest near the line, and have similar spread for all values of x; these properties all imply $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is a reasonable assumption. On the contrary, the right plot shows a poor fit. The data are above the line for small and large values of x, and below the line for intermediate values. The line does not depict the trend in the data.

These observations about the regression fit are easier to see in a residual plot against x.

```
resid1 <- lmod1$residuals
resid2 <- lmod2$residuals
plot(x, resid1)
abline(h = 0)
plot(x, resid2)
abline(h = 0)
```

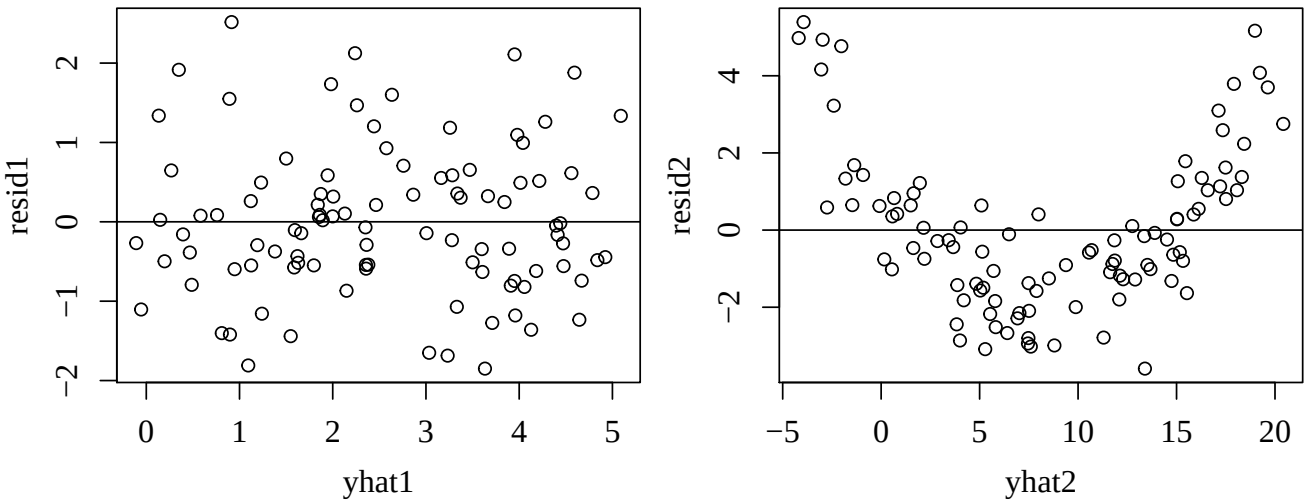


The horizontal line at zero represents perfect prediction. We would prefer the residuals to randomly scatter about this line, have a similar spread for all x, and if assuming normal errors be densest near this line. The observations made on the plots of y against x apply here as well and are easier to make since, by OLS construction, the residuals have mean zero and are uncorrelated with x. (See residual properties in Chapter 2, Exercises.) This makes nonrandom trends, e.g. parabolic, and heteroskedasticity apparent, indicating model misspecification.

Why then do we usually plot residuals plotted against the fitted values? In simple linear regression,

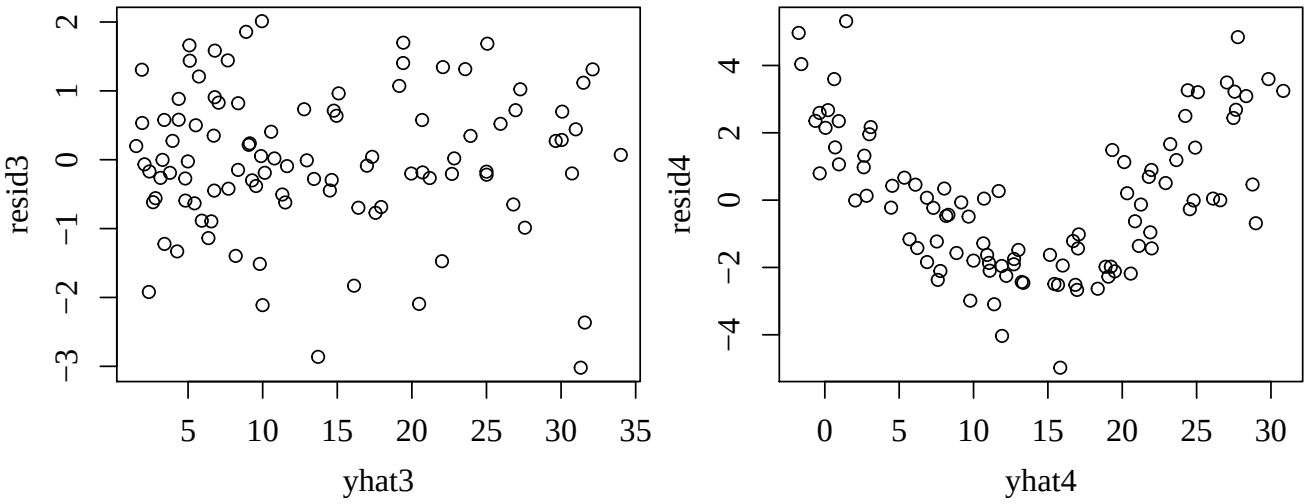
the residuals against fitted plot contains the exact same information because $\widehat{y} = \widehat{\beta}_0 + \widehat{\beta}_1 x$ is just a linear transformation, i.e. corresponds to stretching, reversing, or shifting the x axis. The following plots show this; they look identical if the x-axis labels are removed.

```
yhat1 <- lmod1$fitted.values
yhat2 <- lmod2$fitted.values
plot(yhat1, resid1)
abline(h = 0)
plot(yhat2, resid2)
abline(h = 0)
```



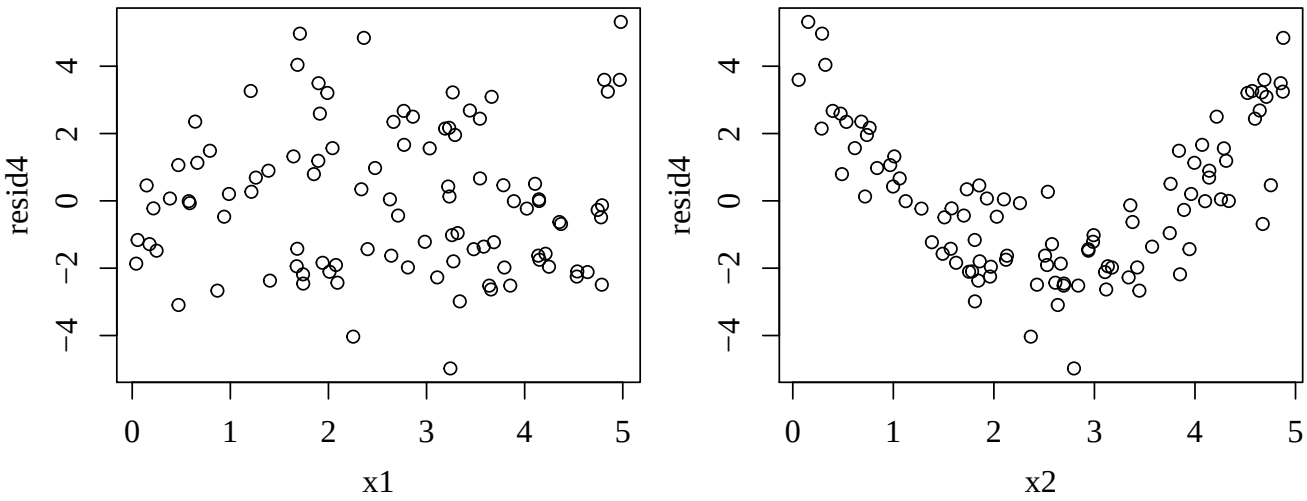
Furthermore, a residuals vs. fitted plot generalizes to multiple regression with more than one predictor. (The residuals still have zero mean and are uncorrelated with the fitted values because they are just linear combinations of predictors.) For example, consider the following 2-variable scenario where one model misses the nonlinear component. The full model yields a satisfactory plot, while the reduced model shows a clear trend.

```
x1 <- runif(n, 0, 5)
x2 <- runif(n, 0, 5)
y <- x1 + x2 + x2**2 + rnorm(n)
lmod3 <- lm(y ~ x1 + x2 + I(x2**2))
lmod4 <- lm(y ~ x1 + x2)
resid3 <- lmod3$residuals
resid4 <- lmod4$residuals
yhat3 <- lmod3$fitted.values
yhat4 <- lmod4$fitted.values
plot(yhat3, resid3)
plot(yhat4, resid4)
```



This diagnostic allows us to check for violations in our model in a single plot, even with many predictors. It can still be useful to check individual residuals vs. predictor plots to identify which predictors are problematic. For instance, the diagnostic plot of `lmod4` suggests a nonlinearity; individual residual plots against predictors identifies x_2 as the problematic variable.

```
plot(x1, resid4)
plot(x2, resid4)
```



6.1.1 Constant Variance

6.1.2 Normality

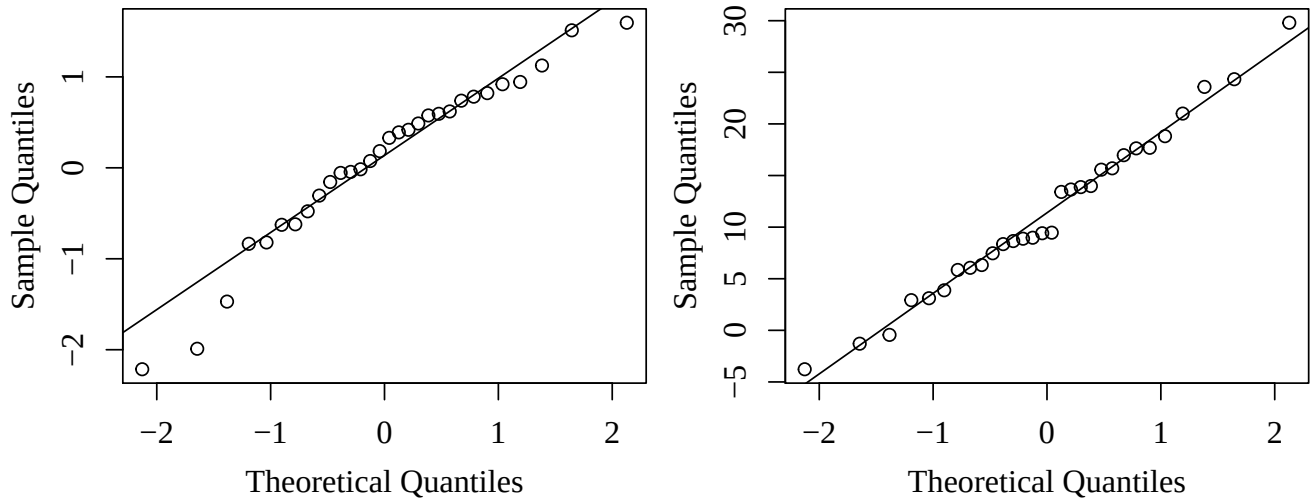
qqplot details. We want a visual diagnostic to detect if observed data $y_1, y_2, \ldots, y_n$ is normally distributed. Histograms are problematic because of arbitrary binning. A qqplot is a better alternative which computes quantiles $q_1, q_2, \ldots, q_n$ from the data and compares each q_i to the expected theoretical quantiles of a normal distribution.

Since we are computing as many quantiles as observations, the i-th quantile is just the i-th sorted value, or order statistic $y_{(i)}$. We can then compare $y_{(i)}$ to the expected theoretical order statistic $x_{(i)}$ generated from a truly normal distribution. If X and Y have the same distribution, we would expect $y_{(i)}$ to be comparable to $x_{(i)}$; indeed, a qqplot is constructed by plotting n pairs $(x_{(i)}, y_{(i)})$.

As a concrete example, suppose we use $X \sim \mathcal{N}(0, 1)$ as a reference for theoretical values. To test if

Y is also standard normal, we would expect the points in the qqplot to lie on the line $y = x$. More generally, to test if Y is normal with mean μ and variance σ^2, we would also expect the points in the qqplot to lie on a line but with slope σ and intercept μ since any other normal distribution is just a linear transformation $Y = \sigma X + \mu$. This is why the standard normal is often used as a reference. qqplots of data generated by $Y_1 \sim \mathcal{N}(0, 1)$ and $Y_2 \sim \mathcal{N}(10, 10^2)$ are plotted below.

```
set.seed(1)
n <- 30
y1 <- rnorm(n, 0)
y2 <- rnorm(n, 10, 10)
qqnorm(y1, main = ''); qqline(y1)
qqnorm(y2, main = ''); qqline(y2)
```



The plots fit lines $y_1 = x$ and $y_2 = 10x + 10$ fairly well.

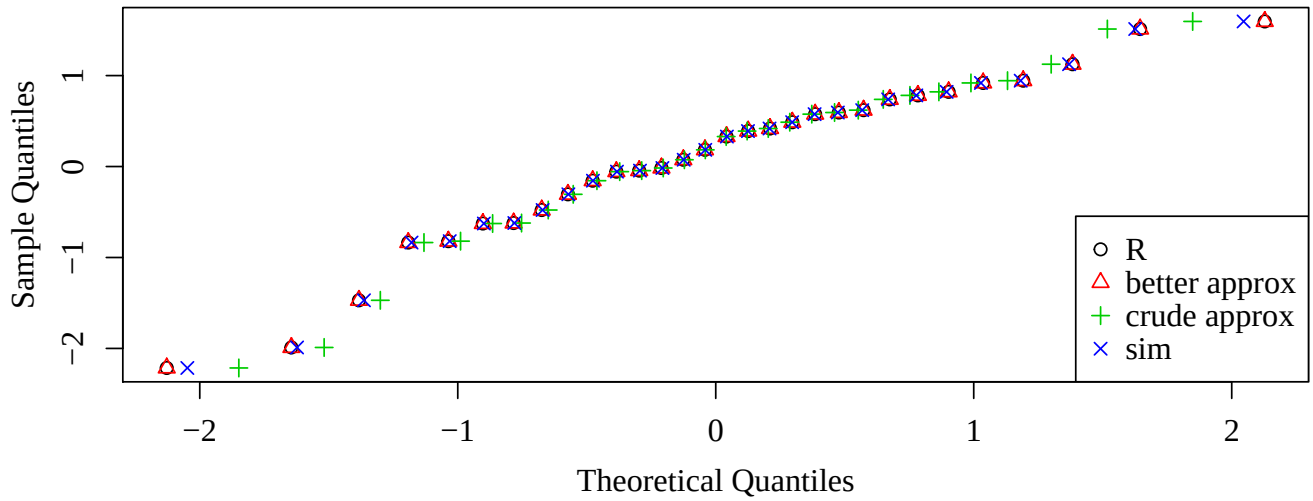
All that remains is how to compute the theoretical quantiles. An intuitively appealing, but computationally expensive, method simulates many random samples from X and records the mean of each order statistic. More analytic approaches rely on order statistics theory. The expectation $\mathbb{E}\, X_{(n)}$ can be computed directly using well known formulas for order statistic densities. A related, but more intuitive approach relies on the probability integral transformation: $\Phi(X)$ has a standard uniform distribution where $\Phi(\cdot)$ is the cumulative distribution function (of the normal distribution). It follows that we can generate random probabilities p from $\mathcal{U}(0, 1)$, sort them into $p_{(i)}$, and compute $x_{(i)} = \mathbb{E}[\Phi^{-1}(p_{(i)})]$. The book's approximation interchanges the expectation, yielding $x_{(i)} \approx \Phi^{-1}(\mathbb{E}\, p_{(i)}) = \Phi^{-1}(i/(n + 1))$. This chooses equally spaced probabilities and maps them to normal quantiles using $\Phi^{-1}(\cdot)$. By Jensen's inequality, this approximation overestimates the left tail and underestimates the right tail. A better approximation used by R is

$$x_{(i)} \approx \Phi^{-1}\left(\frac{i - a}{n + 1 - 2a}\right) \qquad \text{where } a = \begin{cases} 3/8 & \text{if } n \leq 10 \\ 1/2 & \text{if } n > 10 \end{cases}.$$

The following plot compares all three methods using data generated from $Y \sim \mathcal{N}(0, 1)$.

```
set.seed(1)
y <- rnorm(n)

nsim <- 4000
qsim_data <- array(dim = c(nsim, n))
for (i in 1:nsim) {
```

```
    qsim_data[i, ] <- sort(rnorm(n))
}
qsim <- apply(qsim_data, MARGIN = 2, FUN = mean)
qapprox_crude <- qnorm(1:n / (n + 1))
qapprox_better <- qnorm((1:n - 1/2) / (n))

qqnorm(y, main = '')
points(qapprox_better, sort(y), col = 2, pch = 2)
points(qapprox_crude, sort(y), col = 3, pch = 3)
points(qsim, sort(y), col = 4, pch = 4)
legend(
  'bottomright', legend = c('R', 'better approx', 'crude approx', 'sim'),
  col = 1:4, pch = 1:4
)
```



We see that the output of `qqplot()` is identical to the better approximation. This approximation agrees well with exact simulation results, while the crude approximation is biased especially at the tails.

One last note: the previous discussion was specifically for checking normality, but qqplots are more general. The same approach can be applied to check if Y comes from any specified distribution, where simulation can be used to compute theoretical quantiles. If the distributions are the same, the points will lie on $y = x$. However, not all distributions can be checked by referencing a standard form of that distribution as we did in the normal case. This only applies for location and scale families, of which the normal distribution belongs. Two distributions from independent datasets can be also compared by plotting their quantiles against each other done in `qqplot()`; no theoretical quantiles are needed. (If the datasets are different sizes, interpolation of quantiles must be used for the smaller dataset.)

6.1.3 Correlated Errors

6.2 Finding Unusual Observations

6.2.1 Leverage

Derivative interpretation of leverage. In terms of the hat matrix, the fitted values in matrix form are

$$\widehat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}.$$

This is equivalent to the n equations

$$\widehat{y}_i = \boldsymbol{H}_{ii}y_i + \sum_{j \neq i} \boldsymbol{H}_{ij}y_j, \qquad i = 1, 2, \ldots, n.$$

Differentiating w.r.t. y_i yields a derivative expression for the leverage:

$$\frac{\partial \widehat{y}_i}{\partial y_i} = \boldsymbol{H}_{ii} == h_i.$$

A larger leverage implies the change in $\widehat{y}_i$ corresponding to a change in y_i is larger. In other words, data points with large leverage can substantially influence the model fit.

More on Mahalanobis distance. Let's start with a simple 1D example of normally generated data.

```
set.seed(1)
n <- 100
x <- rnorm(n, mean = 0, sd = 1)
```

The mean of the data is expected to be near zero. Distance from the mean can be measured by the standard deviation (s.d.), where for the normal distribution we'd expect 3+ s.d. to be far. In the 1D case, the Mahalanobis distance is precisely the number of s.d. away from the mean.

```
xnew <- 5
mahalanobis(xnew, center = mean(x), cov = var(x))**.5
```

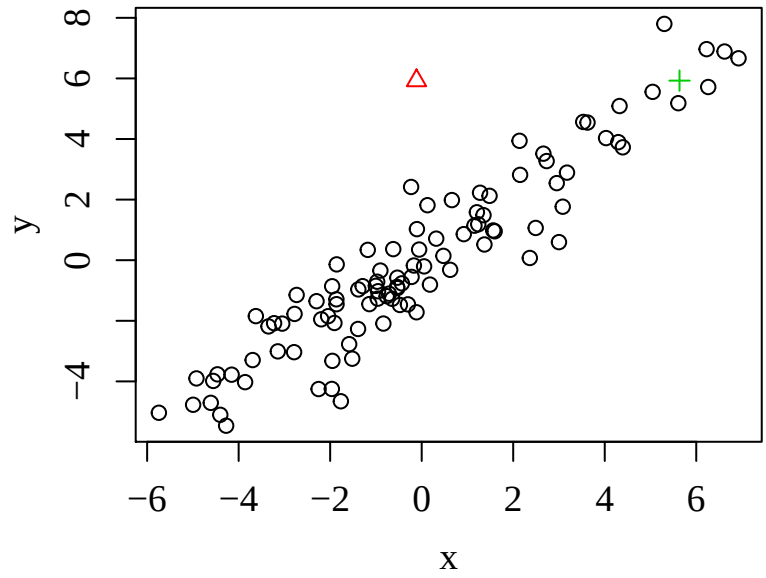
```
## [1] 5.4455
```

```
(xnew - mean(x)) / sd(x)
```

```
## [1] 5.4455
```

In higher dimensions, the Mahalanobis distance generalizes the notion of distance scaled by s.d. while accounting for possible correlations between dimensions. Consider the following correlated 2D example.

```
x <- rnorm(n, 0, 3)
y <- x + rnorm(n)
plot(x, y)
p1 <- c(mean(x), mean(y) + 2*sd(y))
p2 <- c(mean(x) + 2*sd(x), mean(y) + 2*sd(y))
points(p1[1], p1[2], col = 2, pch = 2)
points(p2[1], p2[2], col = 3, pch = 3)
```



We expect the mean, now a vector, to be at the origin. While the red triangle is at the mean of x and 2 s.d. away from y, it is more unusual than a point 2 s.d. away from the mean of x and y. This is due to correlation unaccounted for in Euclidean distance, whereas the Mahalanobis distance scales by the covariance matrix, demonstrated below.

```
M <- matrix(c(x, y), ncol = 2)
mahalanobis(p1, center = apply(M, 2, mean), cov = cov(M))**.5
```

```
## [1] 5.819
```

```
mahalanobis(p2, center = apply(M, 2, mean), cov = cov(M))**.5
```

```
## [1] 2.0312
```

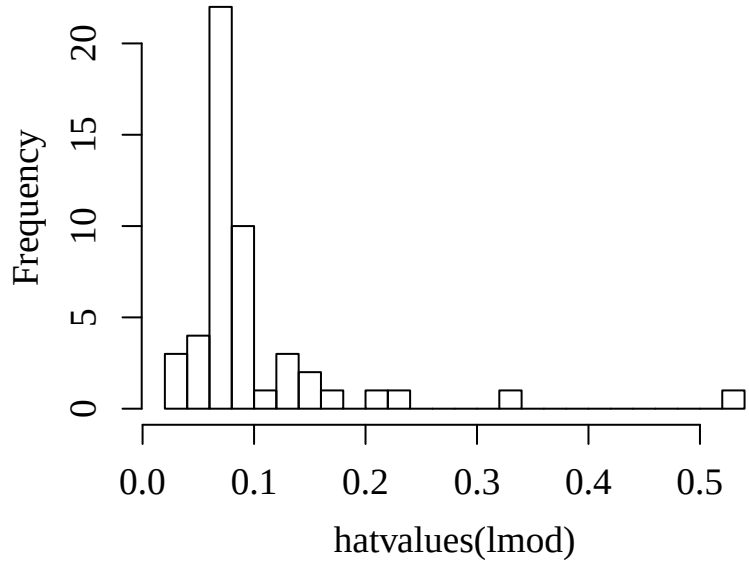
This generalizes to p dimensions, although we can no longer visualize it. Indeed the leverage, representing distance in predictor space, is almost proportional to the (squared) Mahalanobis distance. Thus, a large leverage corresponds to an unusual observation in predictor space.

Sum of leverages $\sum_{i=1}^{n} h_i = p$. The sum of leverages is equal to the trace of the hat matrix $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$ where $\boldsymbol{X}$ is a $n \times p$ design matrix. Since $\text{tr}(\boldsymbol{AB}) = \text{tr}(\boldsymbol{BA})$,

$$\text{tr}(\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T) = \text{tr}(\boldsymbol{X}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}) = \text{tr}(\boldsymbol{I}_{p\times p}) = p. \qquad \blacksquare$$

Identifying large leverages. The book uses a half-normal plot to identify unusual leverages. Since the distribution of leverages is unknown, an alternative is to simply histogram them. This still has the limitation of arbitrary binning, but can still be useful. A histogram of leverage of the book's example is below.

```
lmod <- lm(sr ~ pop15 + pop75 + dpi + ddpi, savings)
hist(hatvalues(lmod), breaks = 20, main = '')
```



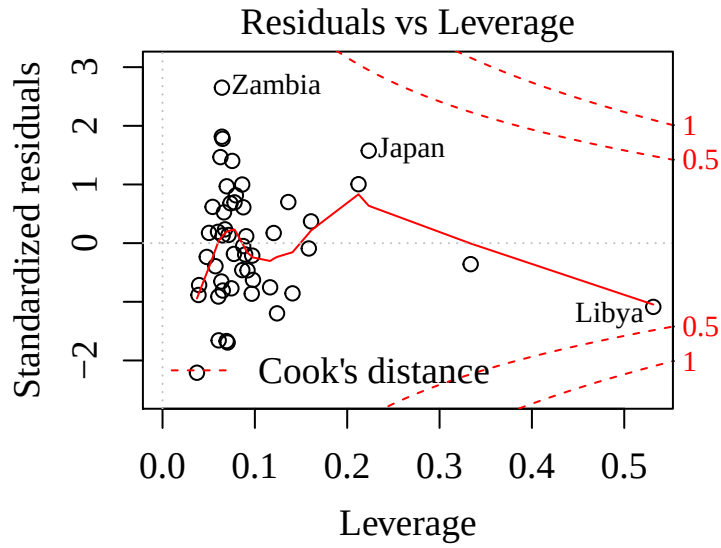
6.2.2 Outliers

Standardization and Studentization. Faraway's standardized residuals r_i, sometimes called internally Studentized residuals, are not t-distributed because the numerator and denominator are not independent. In particular, the ith residual is used to estimate the overall residual variance. Faraway's Studentized residuals t_i, sometimes called externally Studentized residuals, are t-distributed with one less d.f. because the overall residual variance is estimated from $n-1$ points, as the ith residual is excluded which also makes the numerator and denominator independent.

6.2.3 Influential Observations

Figure 6.12. The Cook's distance contours are 0.5 and 1, not 0 and 1.

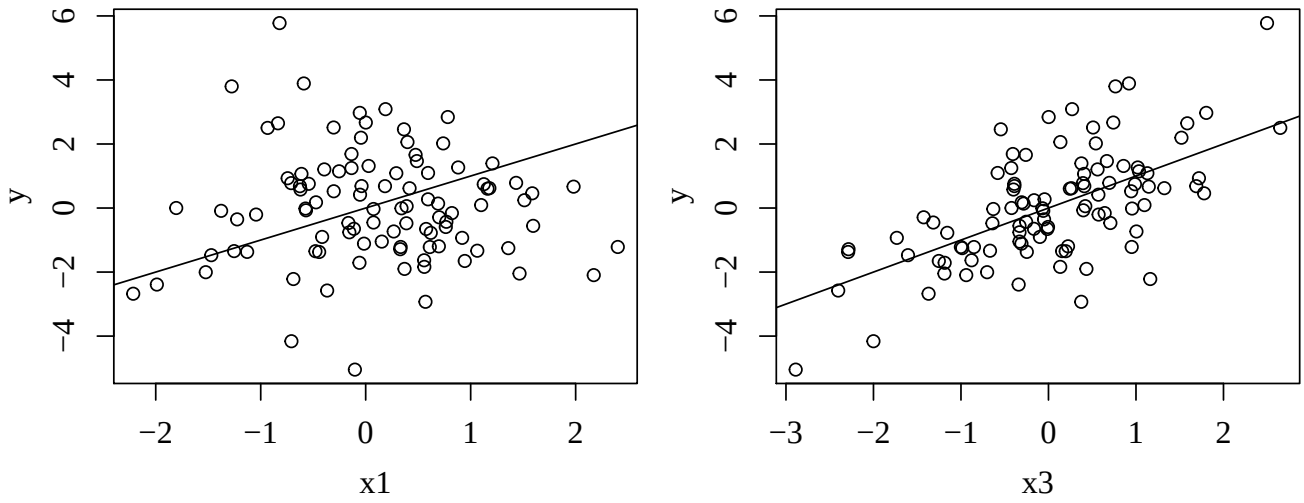
```
plot(lmod, which = 5)
```



6.3 Checking the Structure of the Model

Interpreting partial regression and residual plots. In simple linear regression with a single predictor, it is sufficient to plot y against x to check the model structure. We can see how y changes as a function of x; no partial regression or residual plots are needed. In multiple dimensions if we want to check the model structure w.r.t. x_j, a plot of y against x_j is no longer sufficient. To see this, consider a 3 predictor scenario where y is truly a linear function of x_j but x_1 and x_2 are negatively correlated. Since $\mathbb{E}\,Y = x_1 + x_2 + x_3$, we might expect a plot of y against x_j to be linear with slope 1 and intercept 0. This is the case for x_3 but not for x_1, evidenced below.

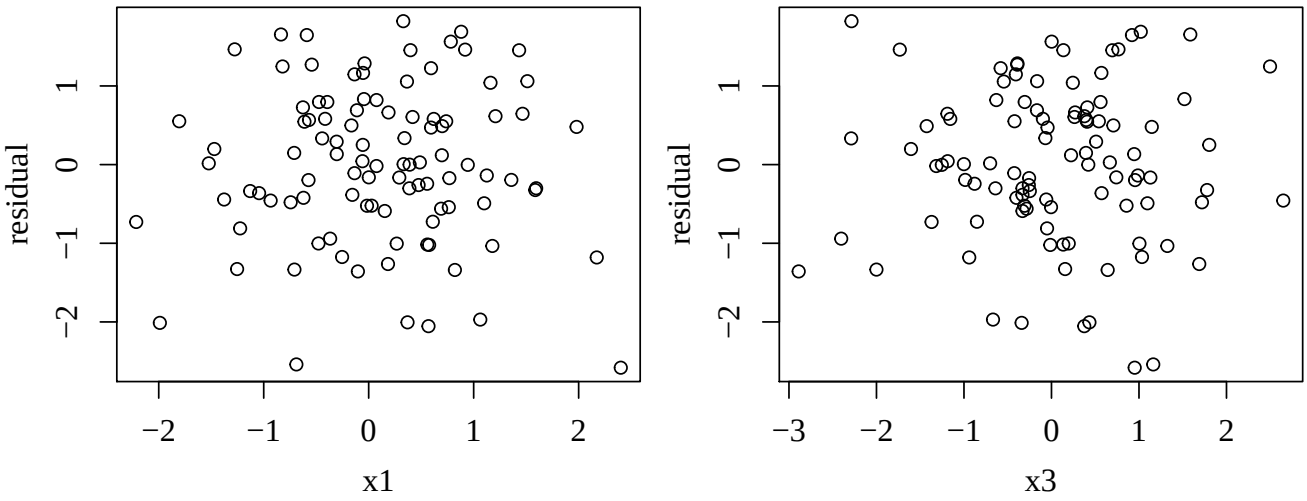
```
set.seed(1)
n <- 100
x1 <- rnorm(n); x2 <- -x1 + rnorm(n); x3 <- rnorm(n)
y <- x1 + x2 + x3 + rnorm(n)
plot(x1, y); abline(0, 1)
plot(x3, y); abline(0, 1)
```



More carefully, we expect a plot of y against x_j to be linear with slope 1 and intercept 0 if all the other predictors are held constant. This is fine for x_3, since it is independent of all other variables, but not x_1. As x_1 increases, x_2 tends to decrease, so the combined effects on y cancel out, on average. Partial regression and residual plots help account for effects of other predictors.

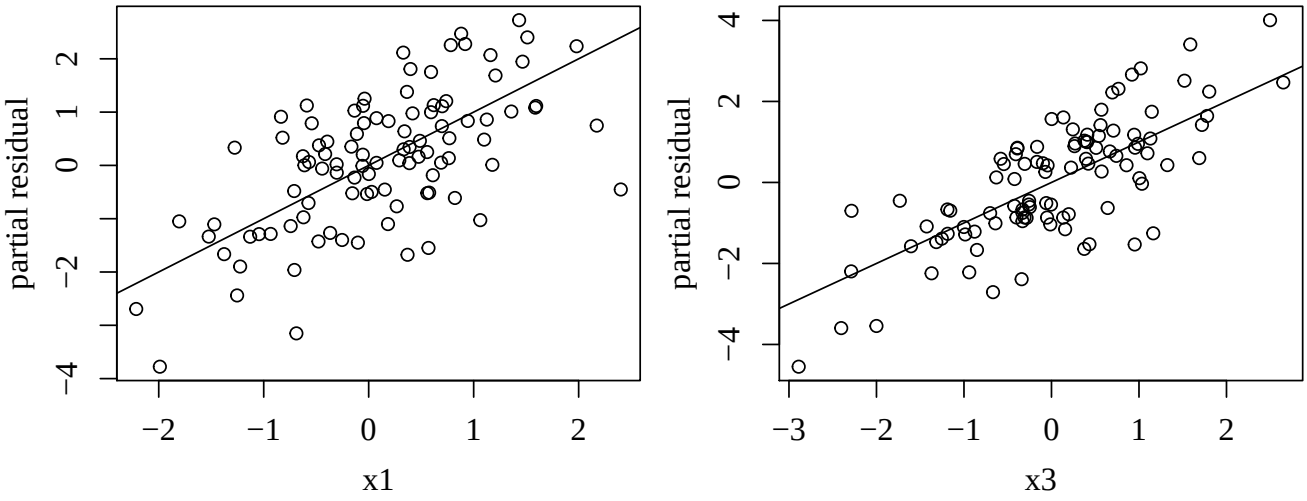
Ideally we'd like to mimic an effect plot—a plot of y against x_j, all other predictors fixed. We can do this with our regression model where predictions $\widehat{y}$ are defined for all possible predictor values, but not with the data at hand y. The best we can do is to use residuals $\widehat{\varepsilon} = y - \widehat{y}$, which subtracts out the effects of all predictors. If there is still a trend in a plot of $\widehat{\varepsilon}$ against x_j, it means the model structure could be improved w.r.t. x_j. For instance, the residual plots for x_1 and x_3 both show no trend.

```
lmod <- lm(y ~ x1 + x2 + x3)
resid <- lmod$residuals
plot(x1, resid, ylab = 'residual')
plot(x3, resid, ylab = 'residual')
```



A partial residual plot is closely related. Rather than plotting $\hat{\varepsilon}$, it plots a modified residual $(\hat{\varepsilon} + \beta_j x_j)$ so only the effects of predictors other than x_j are accounted for. These plots we expect x_j to be linear with slope 1 and intercept 0.

```
plot(x1, resid + lmod$coefficients['x1']*x1, ylab = 'partial residual')
abline(0, 1)
plot(x3, resid + lmod$coefficients['x3']*x3, ylab = 'partial residual')
abline(0, 1)
```



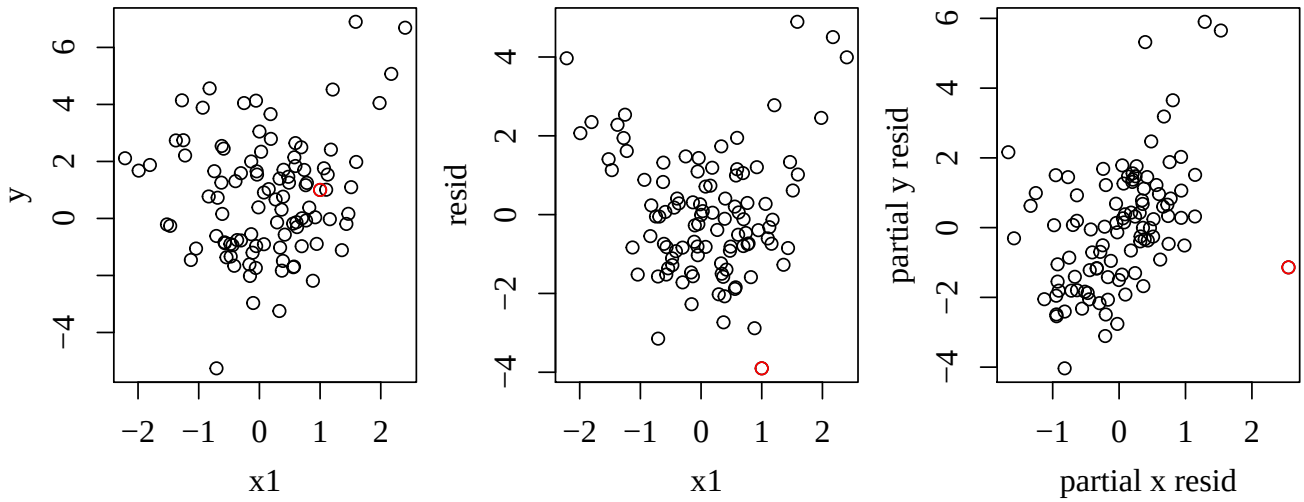
In the partial residual plot, as x_1 changes so does x_2, but the effect of x_2 on y has been accounted for.

Partial regression plots go one step further and directly account for correlations between predictors. They ask if the unexplained part of y by the other predictors is related to unique information contained in x_j not captured by the other predictors. For this reason, it can better detect outliers but is harder to detect model structure issues because the residuals of x are plotted, rather than x.

These plots and a basic scatterplot are compared after adding a quadratic x_1^2 term to the data generating process and an outlier.

```
y <- x1 + x1**2 + x2 + x3 + rnorm(n)
x1[n+1] <- 1; x2[n+1] <- 4; x3[n+1] <- 0; y[n+1] <- 1
plot(x1, y); points(x1[n+1], y[n+1], col = 2)  # Basic plot
```

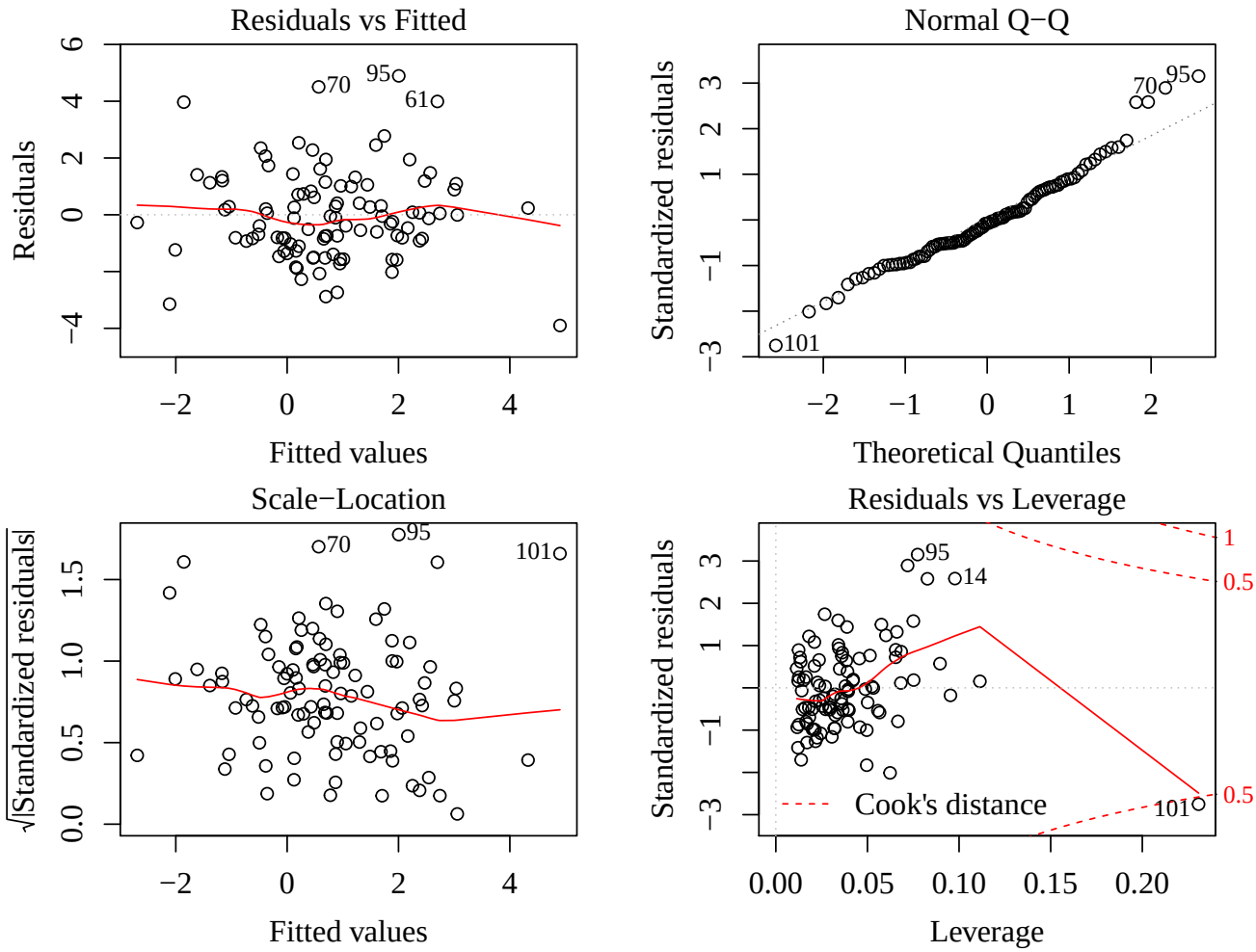
```
# Residual predictor plot
lmod <- lm(y ~ x1 + x2 + x3)
resid <- lmod$residuals
plot(x1, resid); points(x1[n+1], resid[n+1], col = 2)

# Partial residual plot
yp <- lm(y ~ x2 + x3)$residuals
xp <- lm(x1 ~ x2 + x3)$residuals
plot(xp, yp, ylab = 'partial y resid', xlab = 'partial x resid')
points(xp[n+1], yp[n+1], col = 2)
```



The basic scatterplot, unadjusted for other predictors, is not very informative. It does not capture the quadratic structure or the outlier. The residual predictor plot captures the quadratic structure best, while the partial residual plot clearly detects the outlier.

Partial plots are still worth looking at even if the residuals vs. fitted plot is unremarkable. For example, consider the default 4 plots R outputs.

```
plot(lmod)
```



The residuals vs. fitted plot does not show a strong trend, yet further examination previously with the partial plot indicated model structure issues. In this case, the remaining plots are sufficient to check for normality, homoscedasticity, and the outlier.

6.4 Discussion

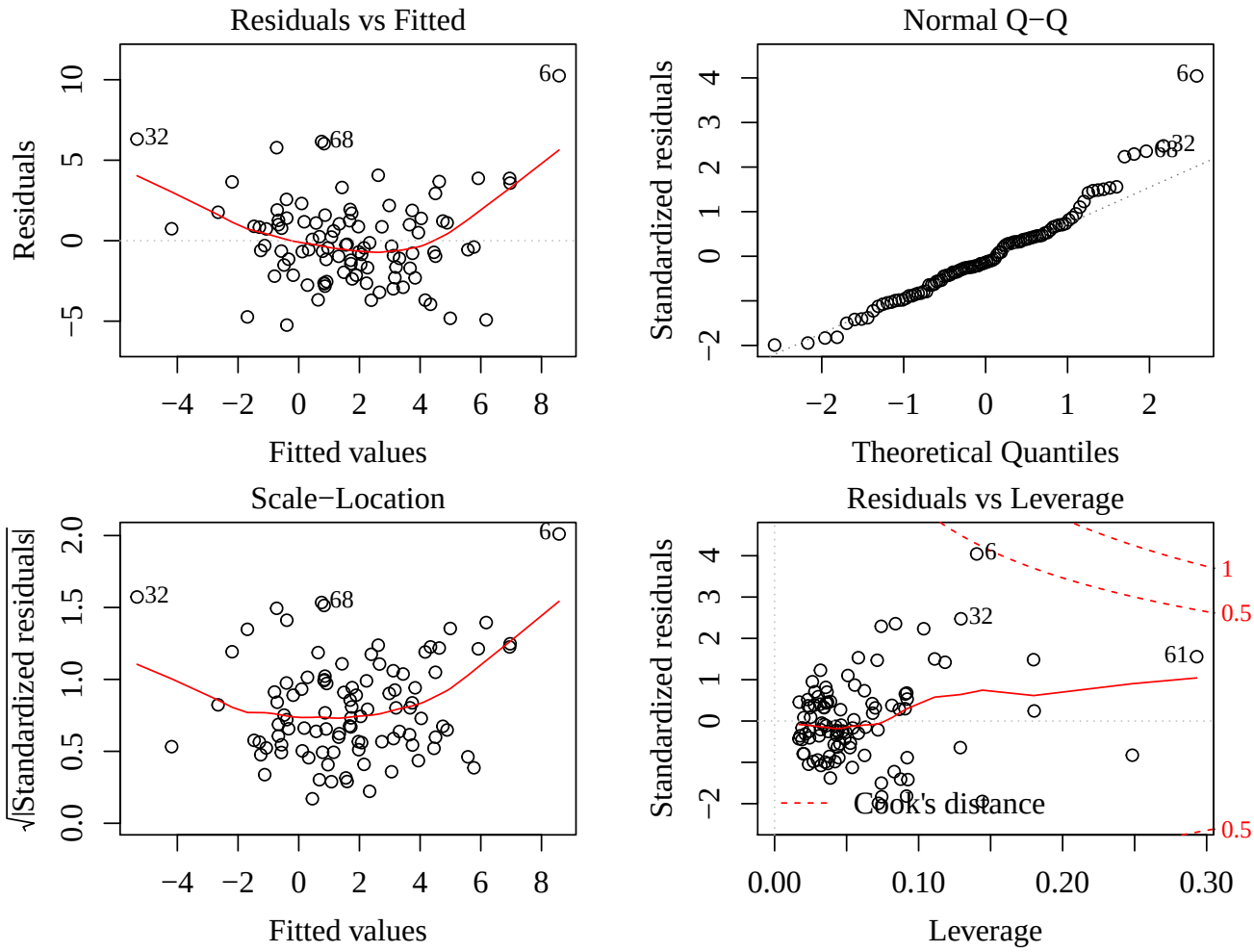
Regression diagnostics with nonlinearities and interactions: worked example. Regression coefficients can no longer be interpreted in isolation when nonlinearities, e.g. quadratic x_1^2, or interactions, e.g. $x_2 x_3$, are present. This example examines the use of familiar regression diagnostics when we know the data generating mechanism to arrive at our final model with one exception: partial residual plots are difficult to define in the presence of interactions. Suppose we have 4 predictors and response generated as follows:

```
set.seed(1)
n <- 100
x1 <- rnorm(n)
x2 <- -.5*x1 + rnorm(n)
x3 <- rnorm(n)
x4 <- rnorm(n)
y <- x1 + x1**2 + x2 + x3 + 2*x2*x3 + x4 + x4**2 + rnorm(n)
```

Further suppose from past knowledge, we expect x_1 to be quadratic, but have no knowledge of the other predictors; for simplicity, we start by assuming they are linear and additive, and plot the

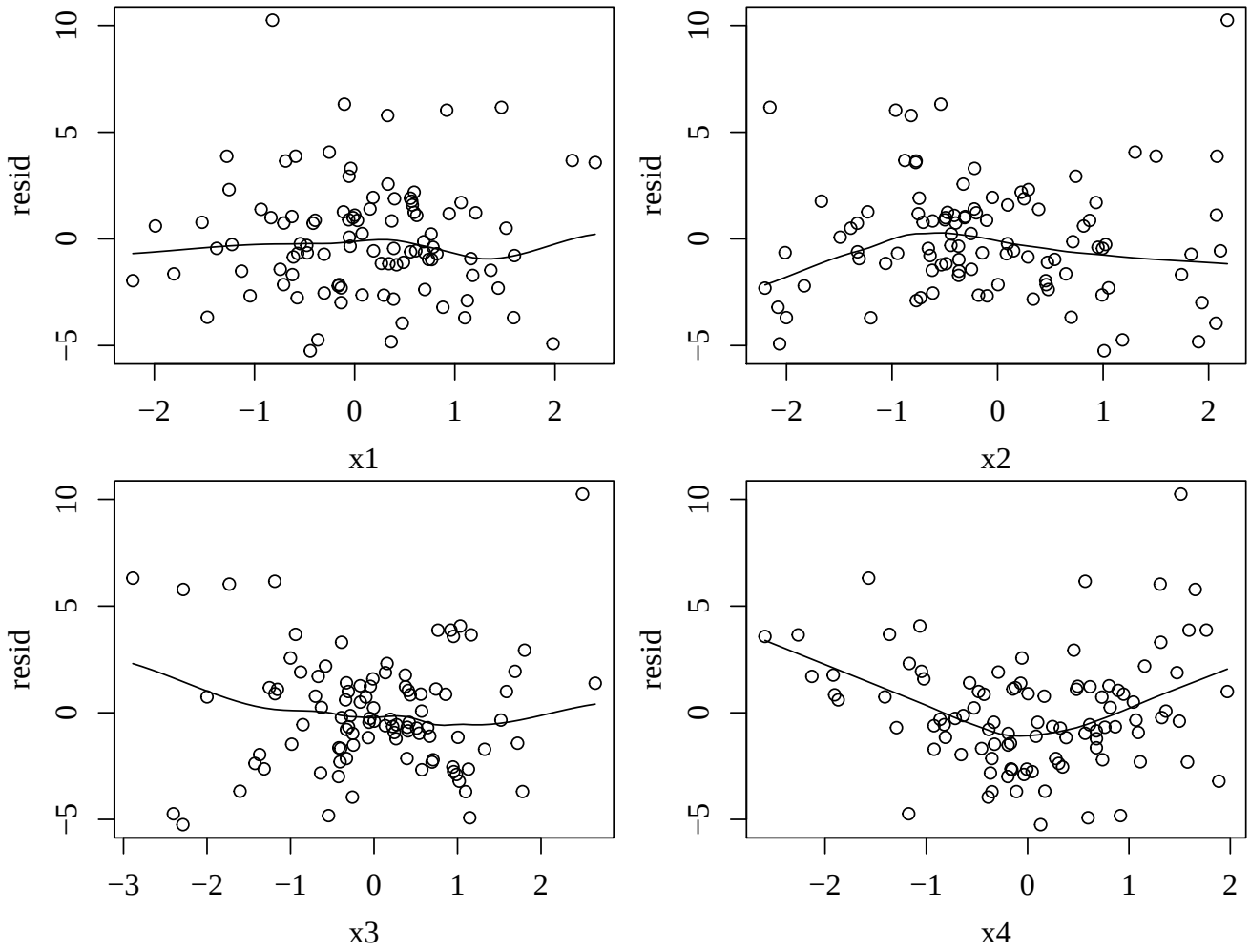
default R diagnostics.

```
lmod <- lm(y ~ x1 + I(x1**2) + x2 + x3 + x4)
plot(lmod)
```



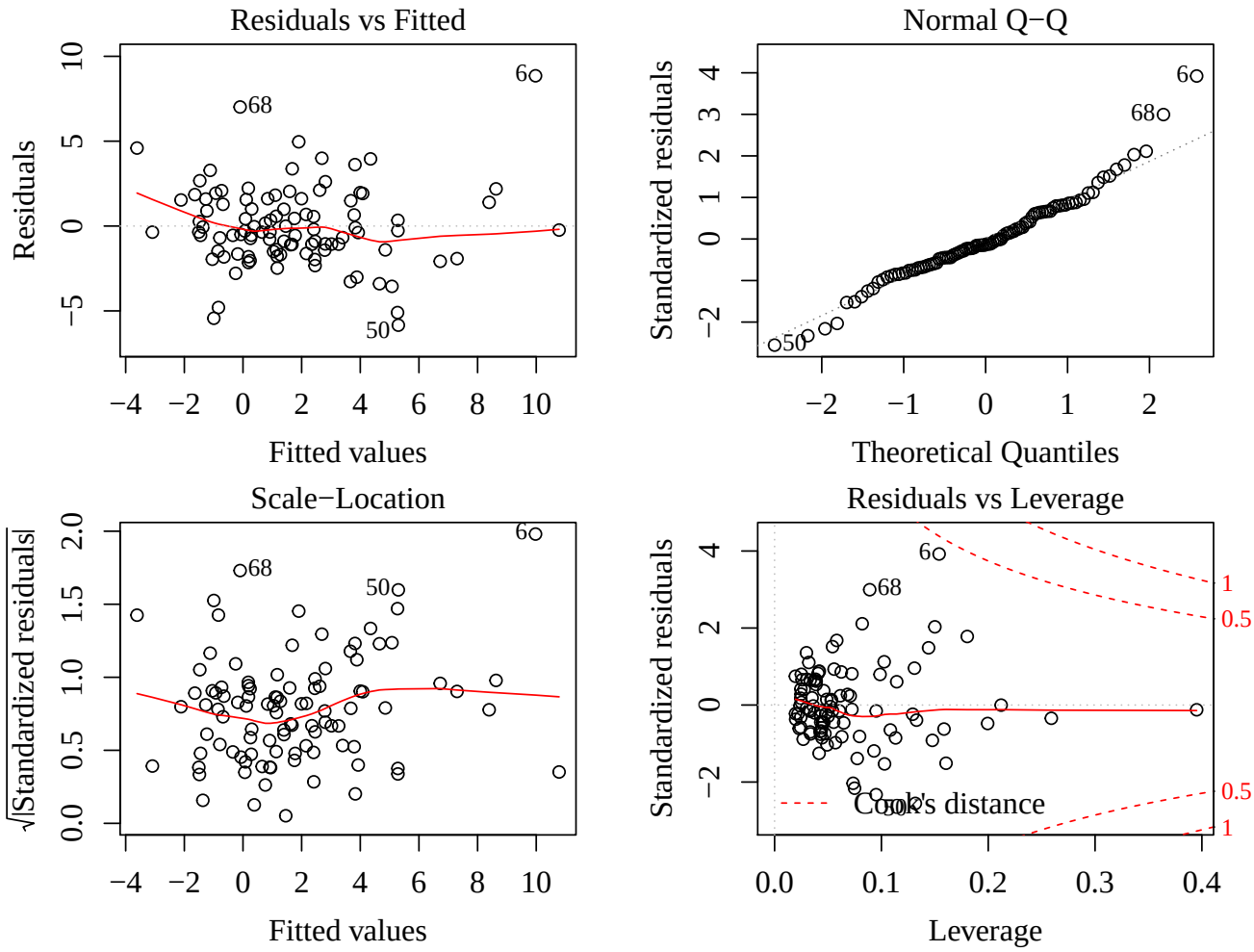
The residual vs. fitted plot shows a trend, suggesting a structural issue. The normality and homoscedasticity assumptions seem reasonable, except at the tails. We also observe a few influential points, labeled in the residuals vs. leverage plot. To address the structural issue, it is helpful to plot residuals against each predictor.

```
resid <- lmod$residuals
scatter.smooth(x1, resid)
scatter.smooth(x2, resid)
scatter.smooth(x3, resid)
scatter.smooth(x4, resid)
```



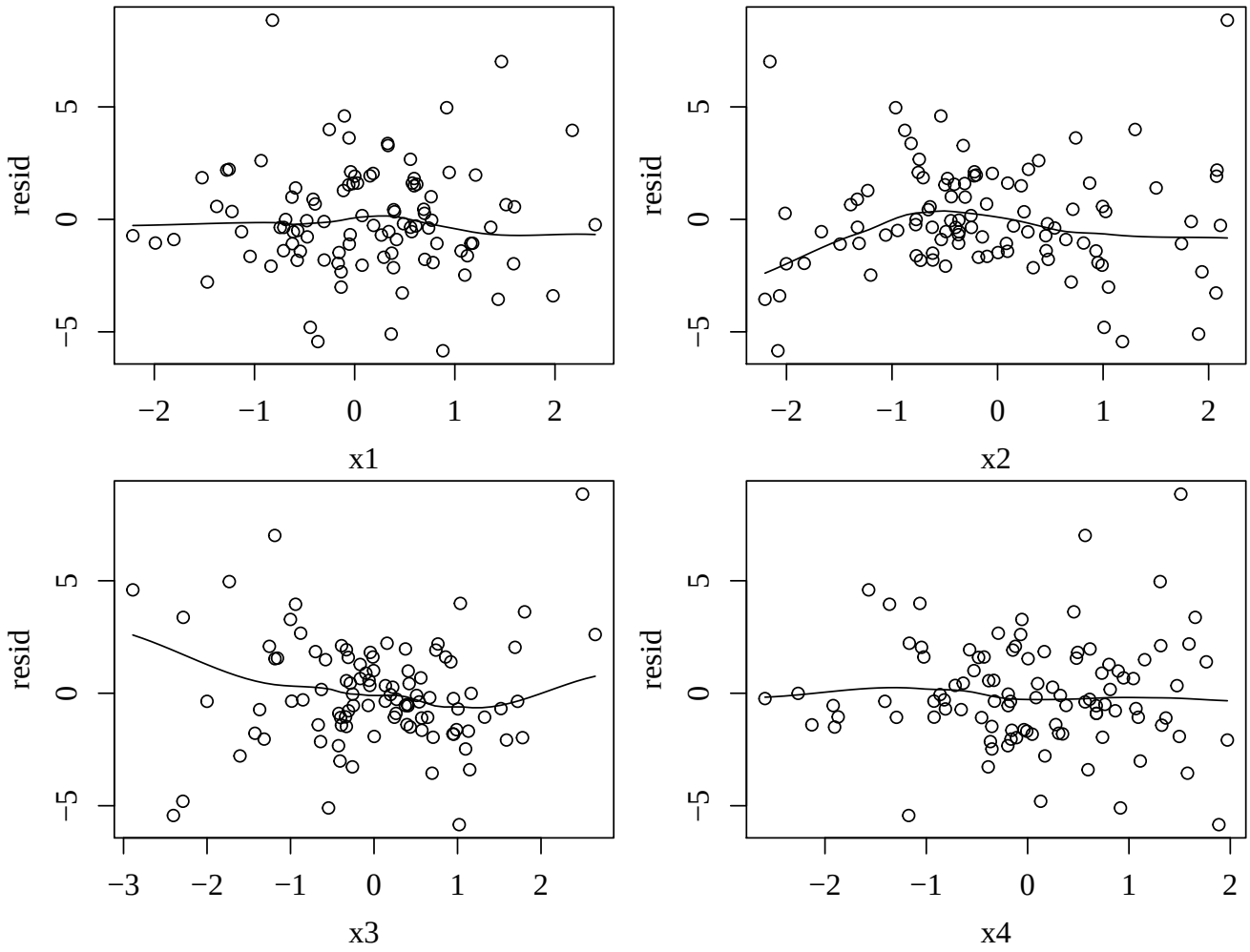
x_4 appears to be the most problematic variable, so we update our model with a quadratic x_4^2 term and examine the updated diagnostics.

```
lmod2 <- lm(y ~ x1 + I(x1**2) + x2 + x3 + x4 + I(x4**2))
plot(lmod2)
```



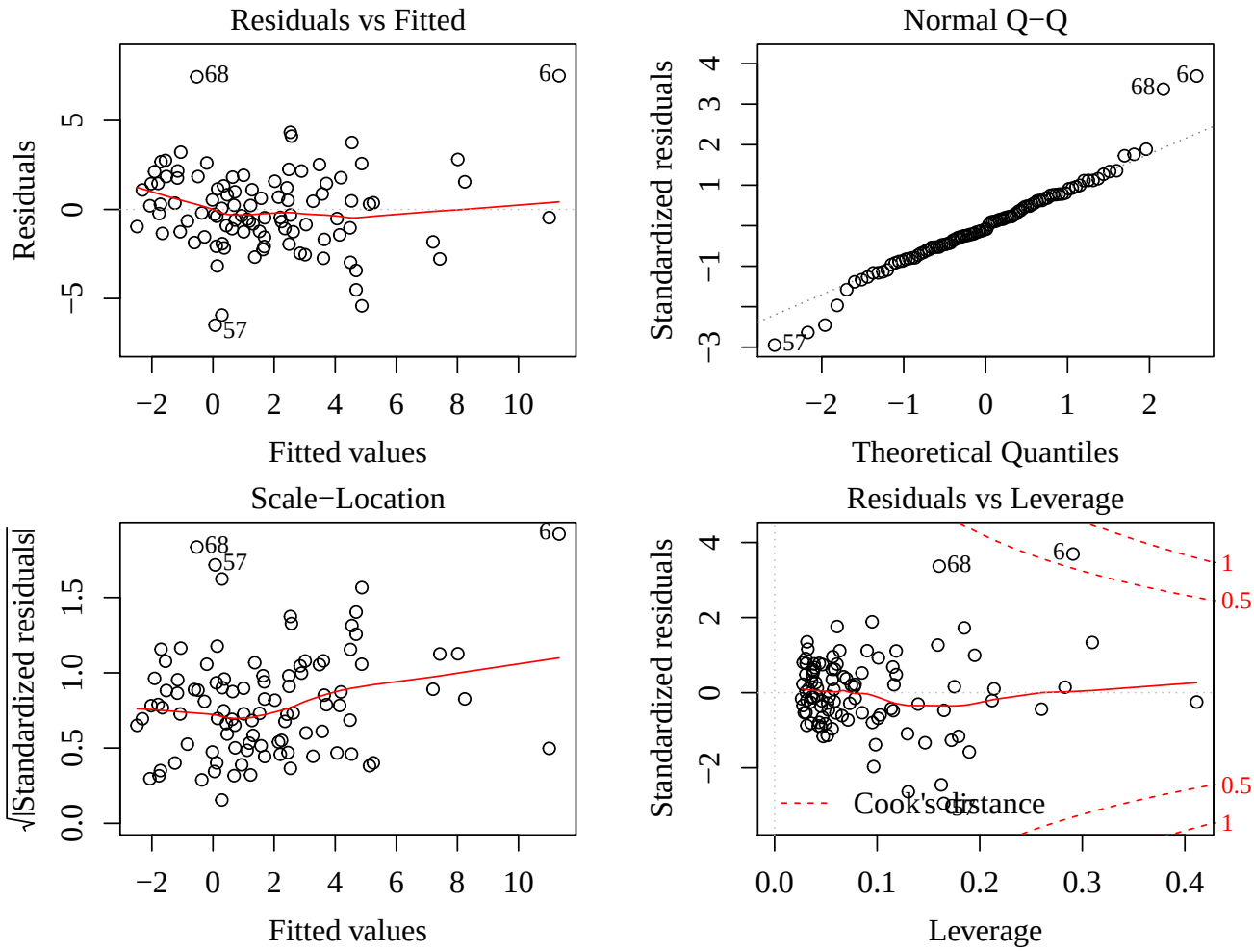
The normality and homoscedasticity assumptions appear more reasonable with this model, and many of the prior deemed influential points are no longer as influential. There still seems to be a slight trend in the residuals vs. fitted plot, so further examination of partial plots is warranted.

```
resid <- lmod2$residuals
scatter.smooth(x1, resid)
scatter.smooth(x2, resid)
scatter.smooth(x3, resid)
scatter.smooth(x4, resid)
```



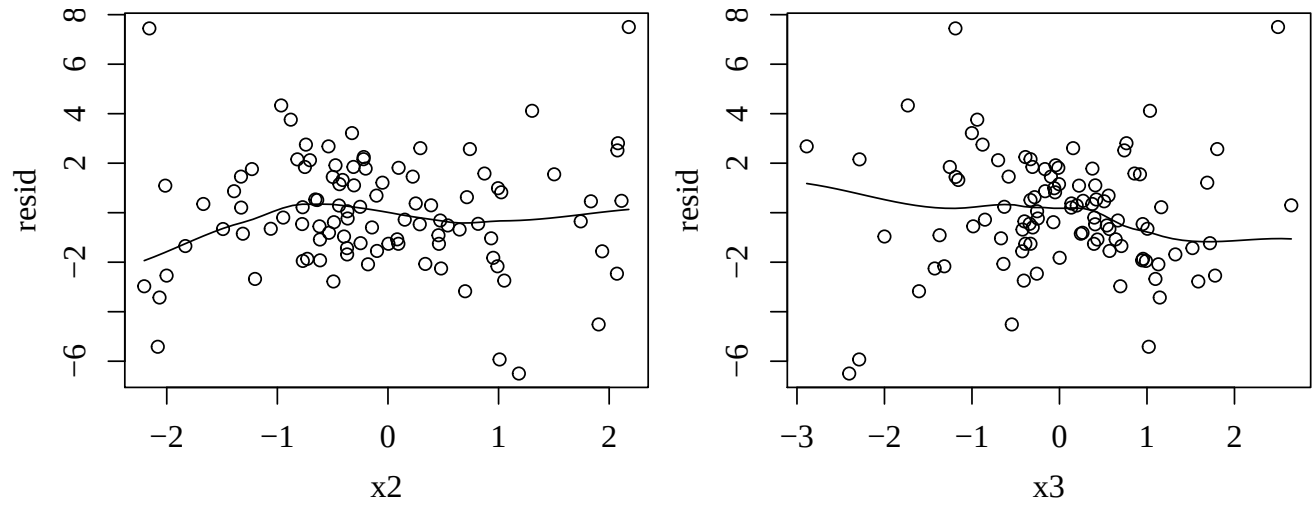
We see that only x_2 and x_3 show a trend at the tails, which prompts us to try adding two more quadratic terms.

```
lmod3 <- lm(y ~ x1 + I(x1**2) + x2 + I(x2**2) + x3 + I(x3**2) + x4 + I(x4**2))
plot(lmod3)
```



The residuals vs. fitted trend is slightly improved and otherwise there is little difference from the previous model. Examining the partial x_2 and x_3 residual plots, we still observe a small trend, but it would not be unreasonable to stop searching.

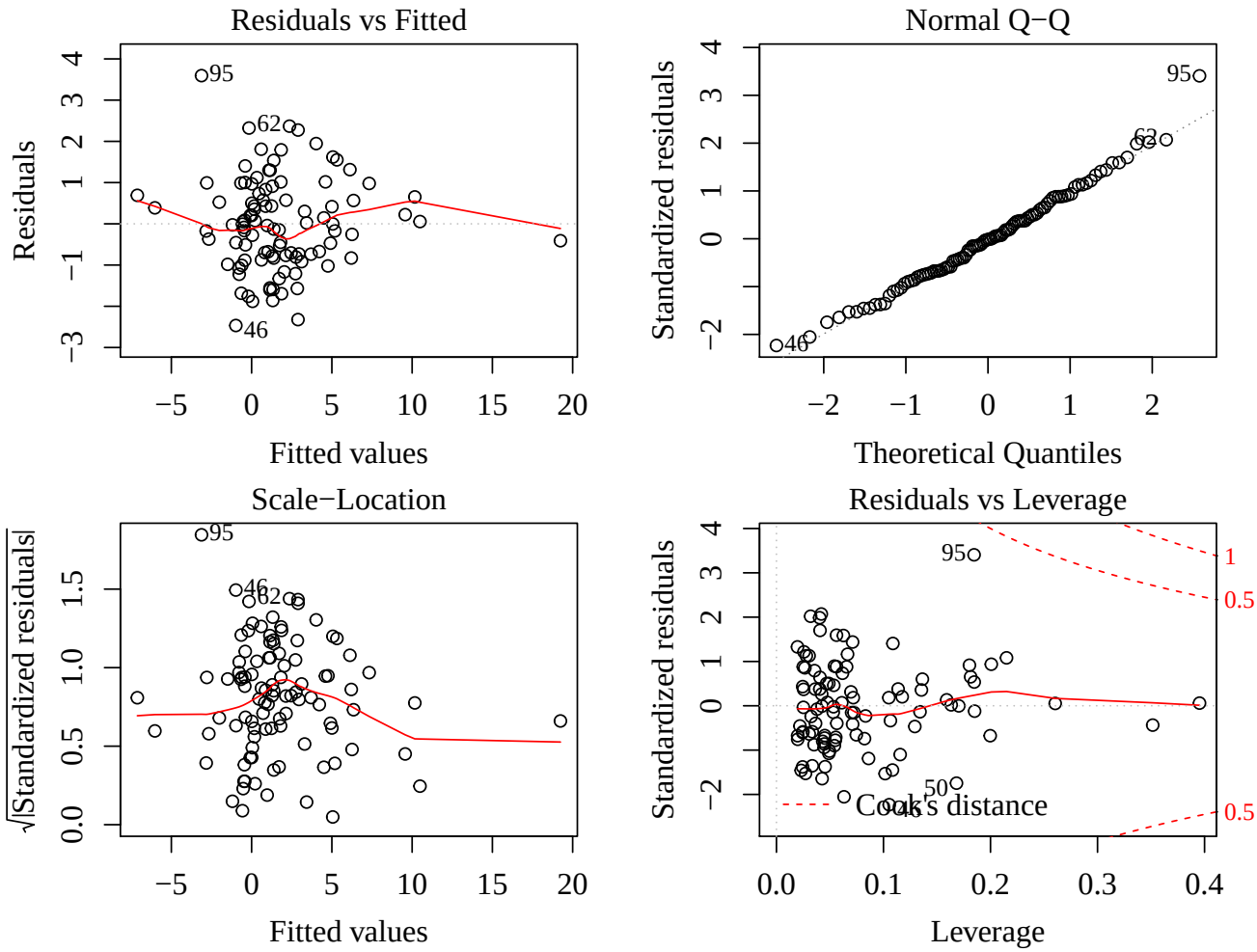
```
resid <- lmod3$residuals
scatter.smooth(x2, resid)
scatter.smooth(x3, resid)
```



However since in this instance we know there truly is an interaction term, we compare the diagnostics

of the 'true' model.

```
lmod4 <- lm(y ~ x1 + I(x1**2) + x2 + x3 + x4 + I(x4**2) + x2:x3)
plot(lmod4)
```



One might argue that the shape of the residual vs. fitted plot is even worse, but must keep in mind the substantial change in scale. If we had tried the model with the interaction, then we might prefer it on numerical diagnostics like R^2.

```
sumary(lmod3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.165      0.401   -0.41  0.68095
## x1             1.411      0.310    4.56  1.6e-05
## I(x1^2)        0.900      0.228    3.94  0.00016
## x2             1.738      0.262    6.63  2.3e-09
## I(x2^2)       -0.182      0.185   -0.98  0.32769
## x3             1.122      0.243    4.62  1.3e-05
## I(x3^2)        0.326      0.154    2.12  0.03679
## x4             1.409      0.267    5.28  8.6e-07
## I(x4^2)        0.942      0.201    4.69  9.5e-06
##
## n = 100, p = 9, Residual SE = 2.412, R-Squared = 0.58
```

```
sumary(lmod4)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1266     0.1647    0.77     0.44
## x1            1.2918     0.1450    8.91  4.6e-14
## I(x1^2)       0.9529     0.1096    8.69  1.3e-13
## x2            1.1461     0.1282    8.94  3.8e-14
## x3            1.0523     0.1150    9.15  1.4e-14
## x4            0.9932     0.1252    7.93  4.9e-12
## I(x4^2)       0.8503     0.0965    8.81  7.2e-14
## x2:x3         2.0153     0.1130   17.83  < 2e-16
##
## n = 100, p = 8, Residual SE = 1.169, R-Squared = 0.9
```

(For a more complete treatment on model selection, see Chapter 10.) Of course, in practical scenarios we do not know the true data generating mechanism, but use of graphical diagnostics to assess model structure and residual assumptions is valuable even when the structure is complex, i.e. contains nonlinearities and interactions.

Exercises

```
library(faraway)
```

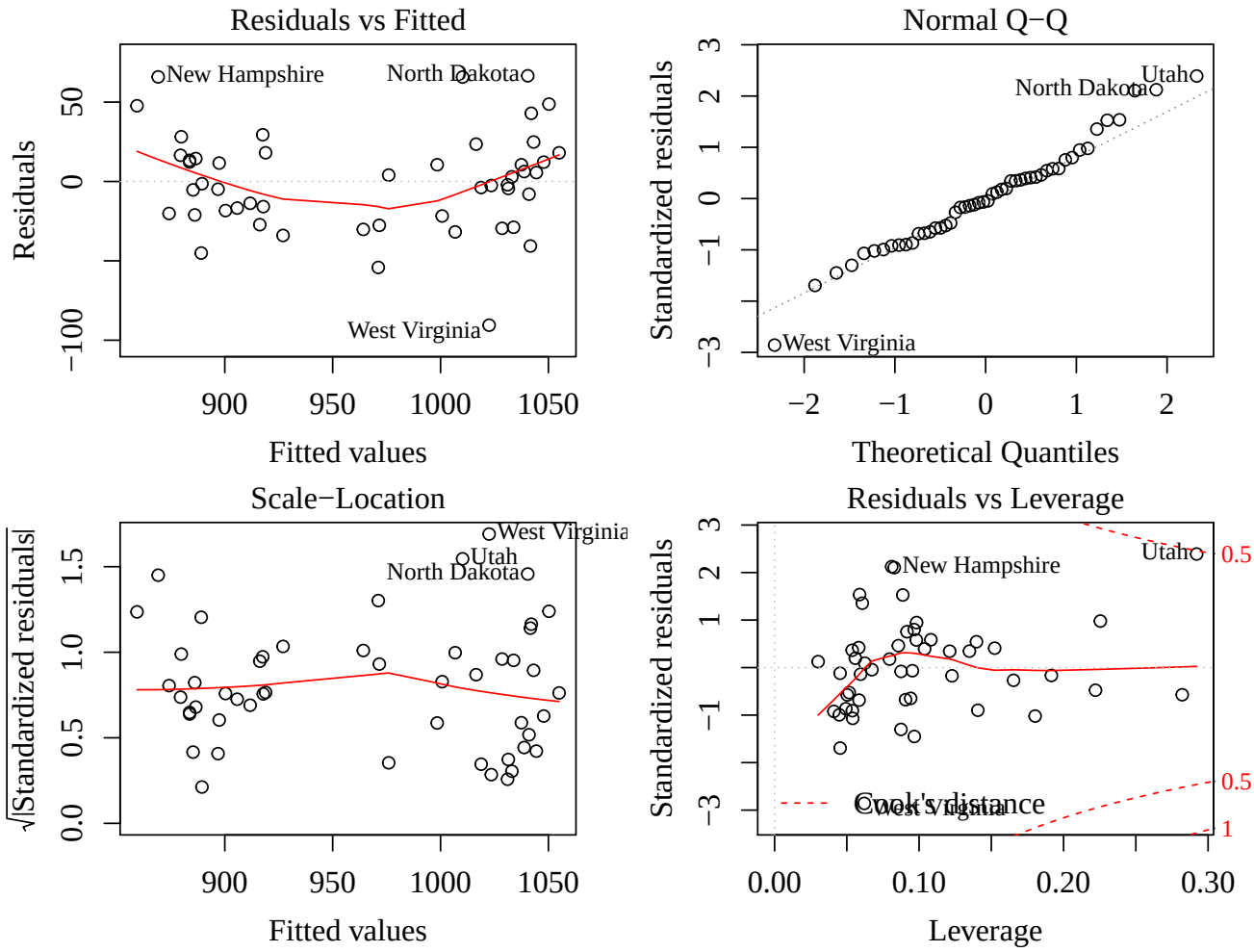
Exercise 1: Regression diagnostics using SAT data. Begin with a numerical summary of the model.

```
df <- sat
lmod <- lm(total ~ expend + ratio + salary + takers, df)
sumary(lmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1045.972     52.870   19.78  < 2e-16
## expend         4.463     10.547    0.42     0.67
## ratio         -3.624      3.215   -1.13     0.27
## salary         1.638      2.387    0.69     0.50
## takers        -2.904      0.231  -12.56  2.6e-16
##
## n = 50, p = 5, Residual SE = 32.702, R-Squared = 0.82
```

The default R diagnostic plots will be useful for most parts of the exercise.

```
plot(lmod)
```



(a) Constant variance of errors. The scale-location plot suggests the standardized residuals have nearly constant variance. There are a few labeled large residuals and a smaller spread in the 910–930 range of fitted values, but given the sample size there are no strong indicators of heteroscedasticity.

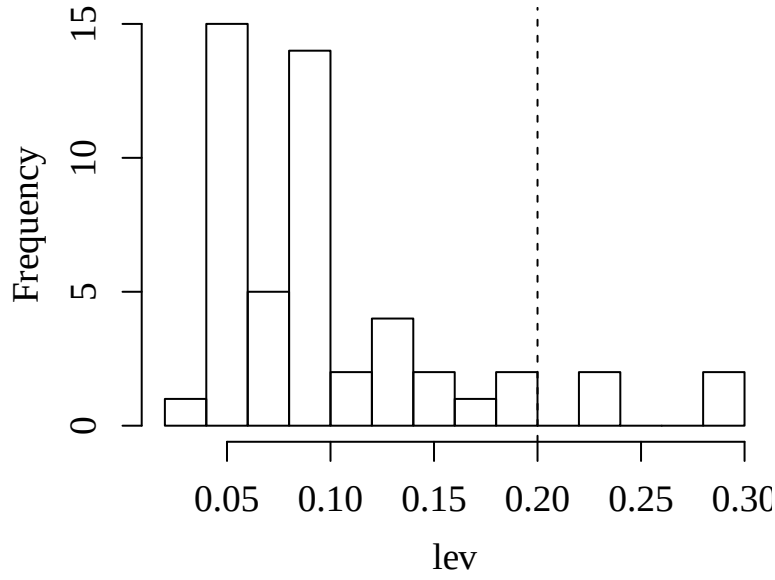
(b) Normality of errors. Most points lie near the line $y = x$ in the normal qqplot, suggesting the assumption of normal errors is not strongly violated. A formal Shapiro–Wilk test agrees.

```
shapiro.test(lmod$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  lmod$residuals
## W = 0.977, p-value = 0.43
```

(c) Large leverage points. The residuals vs. leverage plot indicates a few points have large leverage. A histogram of the leverages is more informative, where a vertical dashed line is placed at two times the mean leverage: $2\overline{h}_i = 2p/n = 0.2$

```
lev <- hatvalues(lmod)
hist(lev, breaks = 10, main = '')
abline(v = 2*mean(lev), lty = 2)
```



The following 4 states correspond to the large leverage points:

```
rownames(df)[lev > 2*mean(lev)]
```

```
## [1] "California"  "Connecticut" "New Jersey"  "Utah"
```

Fortunately, with the exception of Utah, they do not appear too influential based on the Cook's statistic. (More on this in (e).)

(d) Outliers. Based on the standardized residual plots, West Virginia has the most extreme residual and may be an outlier. We can test this formally by using the externally Studentized residuals, which are distributed as $\mathcal{T}(n-p-1=44)$. The resultant p-value

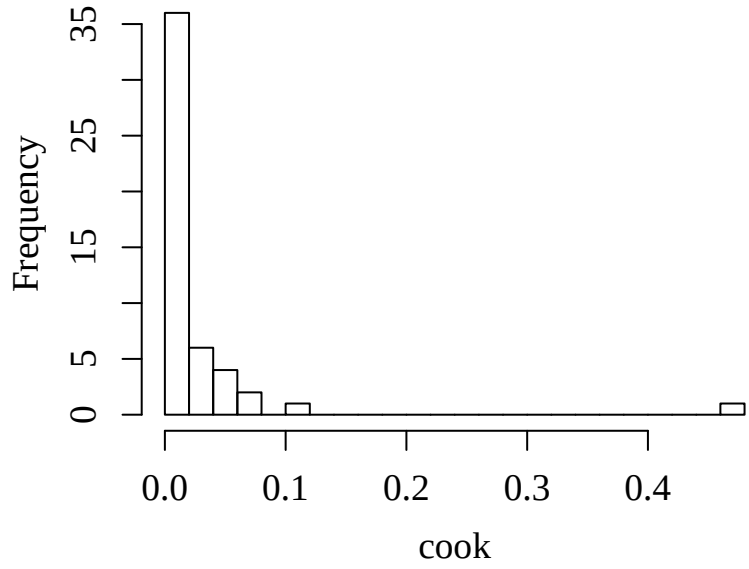
```
stud <- rstudent(lmod)
pt(-max(abs(stud)), 44)*2
```

```
## [1] 0.0031496
```

does not lead to rejection if we use the Bonferroni correction at the level $\alpha = 0.05$, since the level of a α test after correction is $\alpha/n = 0.001$.

(e) Influential points. Based on the Cook's distance of the residuals vs. leverage plot, Utah is the most influential point. Closer examination of a histogram of Cook's distances reveals Utah has the largest distance by far.

```
cook <- cooks.distance(lmod)
hist(cook, breaks = 20, main = '')
```



Let's repeat the regression excluding Utah.

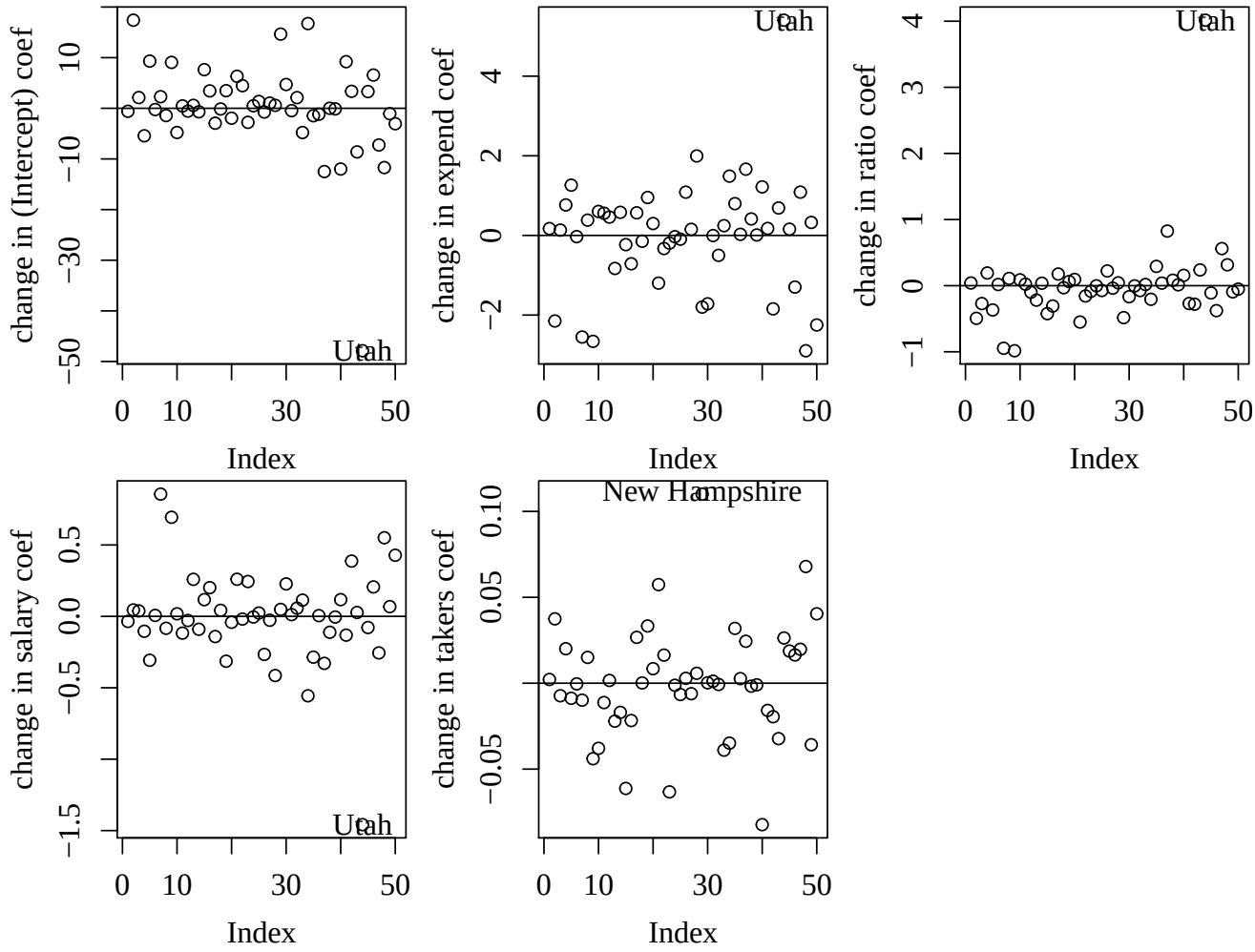
```
lmod2 <- update(lmod, subset = cook < max(cook))
sumary(lmod2)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1093.846     53.423   20.48   <2e-16
## expend        -0.943     10.192   -0.09    0.927
## ratio         -7.639      3.428   -2.23    0.031
## salary         3.096      2.328    1.33    0.190
## takers        -2.931      0.219  -13.40   <2e-16
##
## n = 49, p = 5, Residual SE = 30.901, R-Squared = 0.84
```

All the coefficients except `takers` change substantially, relative to their standard errors. The RSE and R^2 only improve slightly as `takers` seems to be the most important predictor. Combined with the null outlier test in (d), it is reasonable to include Utah in the model. The results suggest that, for some reason, Utah differs in the 3 other predictors compared to states with small `takers` values.

We may wonder if any other observation are influential, particularly on the `takers` coefficient. This is done using `dfbeta()`, which computes the change in coefficient estimates after removing an observation. This yields the following plots for each predictor, plus the intercept.

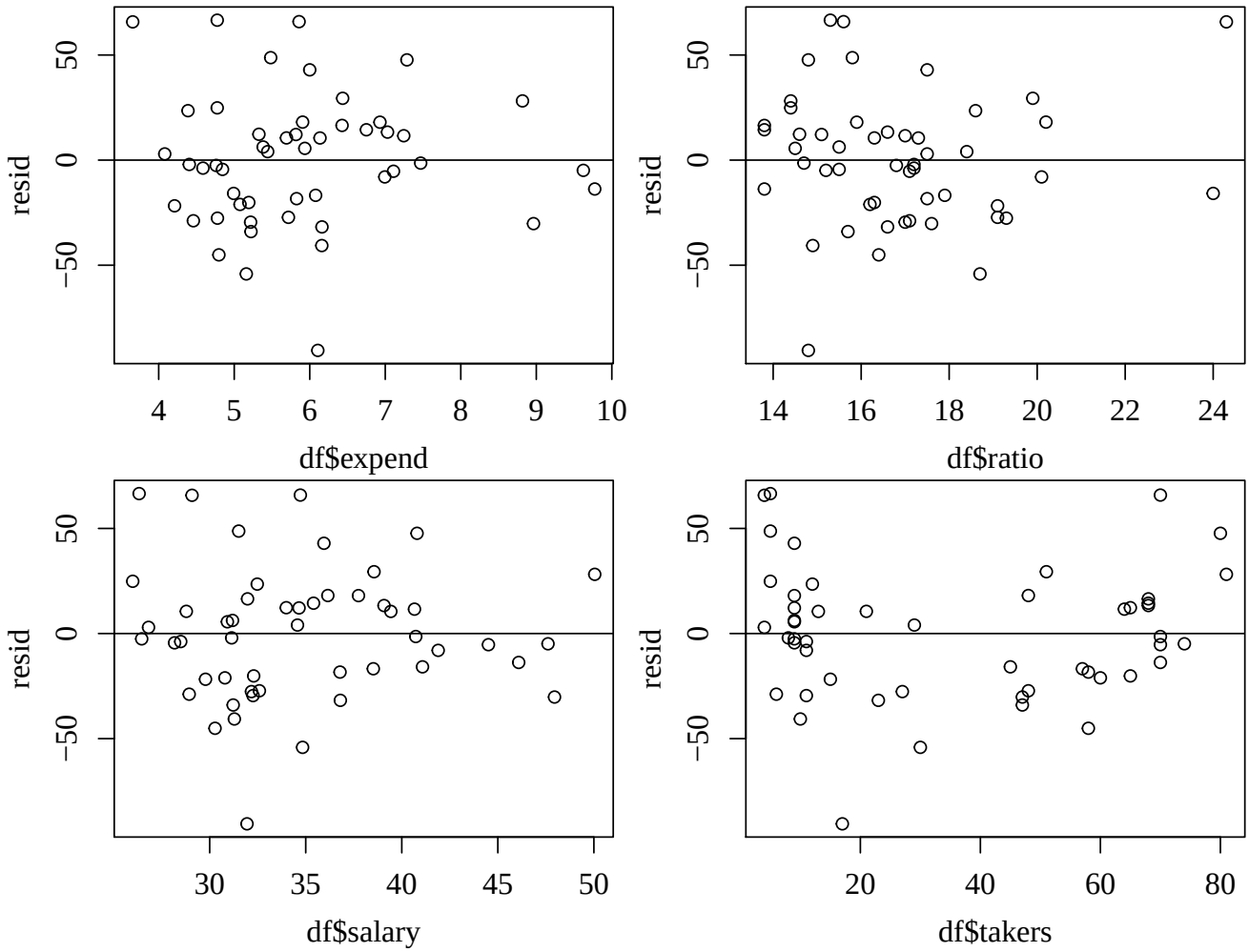
```
diffbeta <- dfbeta(lmod)
for (pred in c('(Intercept)', 'expend', 'ratio', 'salary', 'takers')) {
  plot(diffbeta[, pred], ylab = paste('change in', pred, 'coef'))
  abline(h = 0)
  largest <- names(which(abs(diffbeta[, pred]) == max(abs(diffbeta[, pred]))))
  text(
    x = which(rownames(df) == largest),
    y = diffbeta[largest, pred],
    labels = largest
  )
}
```



New Hampshire has the largest change in the `takers` coefficient, but it is small relative to the effect size. Utah causes the largest change by far in the other coefficients.

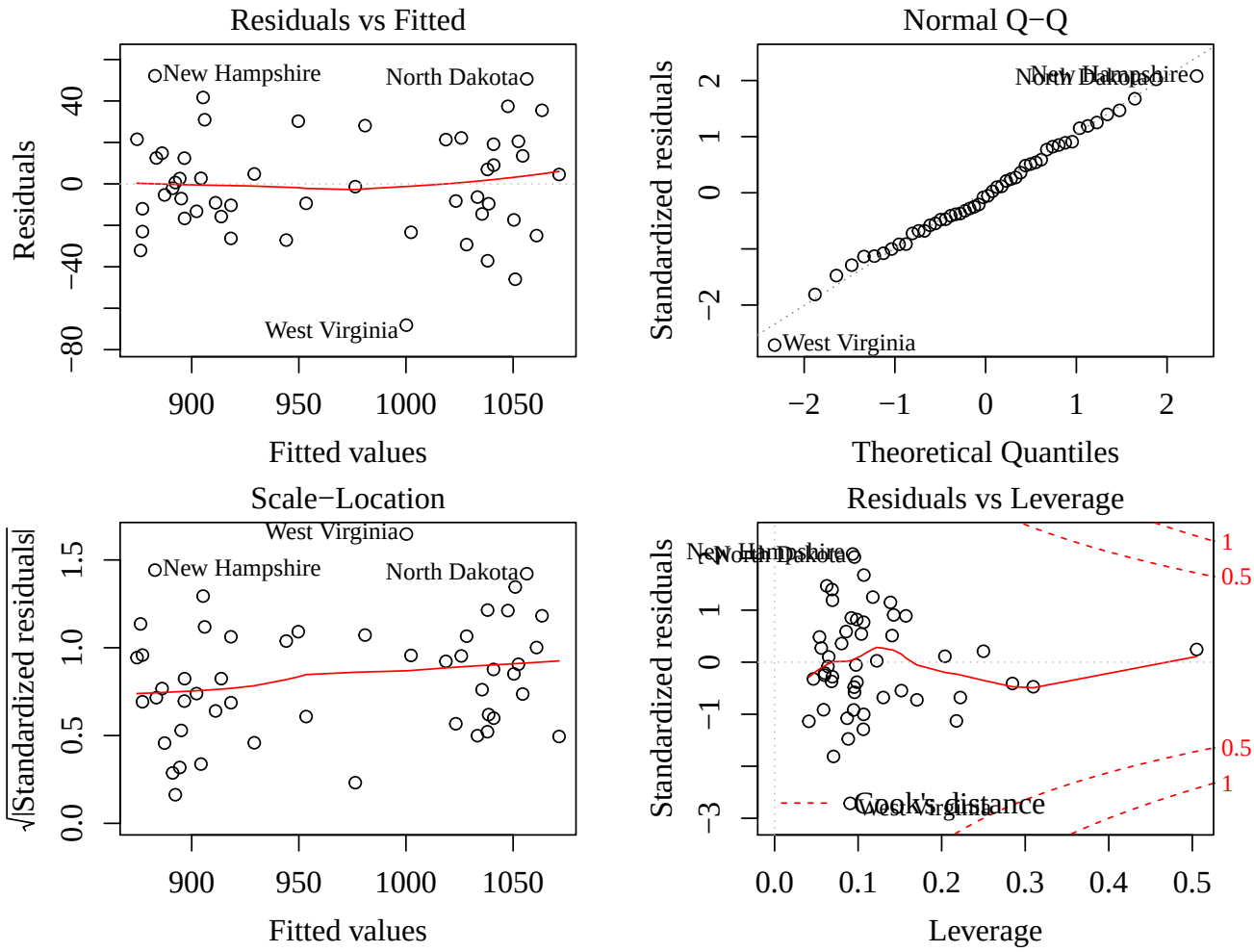
(f) Model structure. The residuals vs. fitted plot appears to have a parabolic trend. Partial plots of residuals against predictors can help us identify which predictor can be improved.

```
resid <- lmod$residuals
plot(df$expend, resid); abline(h = 0)
plot(df$ratio, resid); abline(h = 0)
plot(df$salary, resid); abline(h = 0)
plot(df$takers, resid); abline(h = 0)
```



`takers` seems to exhibit the parabolic trend, so we update our model and default diagnostic plots accordingly.

```
lmod3 <- lm(total ~ expend + ratio + salary + takers + I(takers**2), df)
plot(lmod3)
```



The trend in the residuals vs. fitted plot is greatly reduced (and also in partial plots not shown here), the normality assumption from the qqplot is improved (slightly), and the residuals vs. leverage plot indicates Utah is no longer a high influence point. The scale-location plot suggests an upward trend, perhaps indicating a difference in error variance between two groups. A format test of variance between the two groups shows the difference is not statistically significant.

```
rstan <- rstandard(lmod3)
yhat <- lmod3$fitted.values
var.test(rstan[yhat < 950], rstan[yhat >= 950])
```

```
##
##  F test to compare two variances
##
## data:  rstan[yhat < 950] and rstan[yhat >= 950]
## F = 0.621, num df = 24, denom df = 24, p-value = 0.25
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.27365 1.40918
## sample estimates:
## ratio of variances
##            0.62098
```

Overall, the diagnostics suggest the quadratic `takers` model is an improvement. Since the normality of residuals assumption is reasonable, we can formally compare the two models via F-test using

anova().

```
anova(lmod, lmod3)
```

```
## Analysis of Variance Table
##
## Model 1: total ~ expend + ratio + salary + takers
## Model 2: total ~ expend + ratio + salary + takers + I(takers^2)
##   Res.Df   RSS Df Sum of Sq    F  Pr(>F)
## 1     45 48124
## 2     44 30530  1     17594 25.4 8.6e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Indeed the formal test also suggests the quadratic model is an improvement.

Chapter 7

Problems with the Predictors

```
library(faraway)
library(simex)
```

7.1 Errors in the Predictors

Intuitive explanation for bias of $\widehat{\beta}$ toward zero. Consider the data in Figure 7.2. If we introduce more error in x, each point will shift left or right. It is simpler to consider for now points with large x, say $x > \overline{x} + \sigma_x$. Shifting right will flatten the fit line while shifting left will raise the fit line. The tendency to flatten the line, e.g. bias β toward zero, is due to asymmetrical changes in leverage; shifting right yields a higher leverage point. More generally, for any point (except $x = \overline{x}$), the direction of increasing leverage corresponds to flattening of the fit line.

Bias of $\text{var}\,\widehat{\beta}$**.** For the case of simple linear regression with one predictor, we know $\widehat{\beta}$ is biased toward zero. What about its variance? In general, there are two competing terms. Intuitively since $\widehat{\beta}$ will be smaller, we might expect less variation. However, we must also account for additional variance introduced by errors in x. Indeed, direct calculation of the estimated RSE yields an expression of the form $\widehat{\sigma} = \lambda\sigma + \kappa\beta^2$, where $0 < \kappa < \lambda < 1$. Details can be found in these lecture notes. For the `cars` data, we observe when assuming normality, the OLS estimate of $\text{var}\,\widehat{\beta}$ is biased high.

```
set.seed(1)
lmod <- lm(dist ~ speed, cars, x = TRUE)
simout <- simex(lmod, "speed", 0.5, B = 1000)
summary(simout)
```

```
## Call:
## simex(model = lmod, SIMEXvariable = "speed", measurement.error = 0.5,
##     B = 1000)
##
## Naive model:
## lm(formula = dist ~ speed, data = cars, x = TRUE)
##
## Simex variable :
##                      speed
## Measurement error :   0.5
```

```
##
##
## Number of iterations:  1000
##
## Residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -29.189  -9.488  -2.322  -0.013   9.167  42.988
##
## Coefficients:
##
## Asymptotic variance:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.926      5.649   -3.17   0.0026 **
## speed          3.956      0.406    9.75  5.7e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Jackknife variance:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.926      6.797   -2.64    0.011 *
## speed          3.956      0.418    9.46  1.5e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the jackknife estimate of variance is very similar to OLS.

7.2 Changes of Scale

7.3 Collinearity

For discussions on the different methods of detecting collinearity, see these https://stats.stackexchange.com/ threads:

- Is there a reason to prefer a specific measure of multicollinearity?
- What are the merits of different approaches to detecting collinearity?
- Standardization of variables and collinearity

Condition number intuition. The eigenvalues of $\boldsymbol{X}^T\boldsymbol{X}$ are relevant because $\boldsymbol{X}^T\boldsymbol{X}$ is almost proportional to the sample covariance matrix $\boldsymbol{\Sigma}$. It is exactly proportional when the data are rescaled to have zero mean with no intercept since when $\mathbb{E}\, x_i = 0$, then $\text{cov}(x_i, x_j) = \mathbb{E}(x_i x_j)$. Furthermore, there is a direct connection to principal component analysis (PCA): *the ith principal component is the eigenvector of* Σ *with the ith largest eigenvalue, which is proportional to the amount of variance explained.*[1]

Let's unpack the last statement. Principal components (PCs) are constructed as linear combinations of x_i. The first PC maximizes the explained variance in the data, the next PC is orthogonal to the first and maximizes the remaining explained variance in the data, and so on. If the data suffer from multicollinearity, it is expected that the first PC explains a large amount of variance compared to

[1] Indeed, due to this connection, whuber mentions in this comment that it is better to condition on a centered design matrix.

the last. Since these explained variances are proportional to the eigenvalues, we would also expect a large condition number. Intermediate condition numbers can be helpful because they measure if other PCs explain a substantial portion of the variance, even after the dominant PCs have been accounted for; this would indicate many instances of multicollinearity.

Intuition on PCA connection to eigendecomposition. In 1D, the variance is sufficient to measure the spread of data. In 2D and beyond, specifying the variance of each dimension is not enough because of correlations between variables. The (symmetric) covariance matrix $\mathbf{\Sigma}$ captures this structure; its diagonal terms are variances of variables and off-diagonal terms are covariances between variables.

For concreteness, consider 2D with variables x_1 and x_2. In general, the covariance matrix is

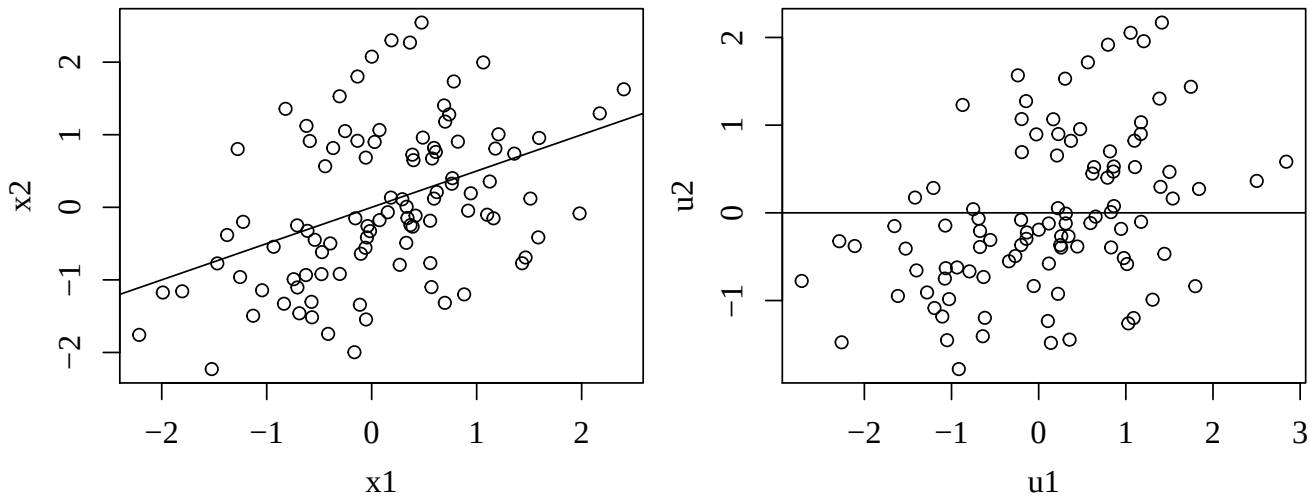
$$\mathbf{\Sigma} = \begin{bmatrix} \operatorname{var} x_1 & \operatorname{cov}(x_1, x_2) \\ \operatorname{cov}(x_2, x_1) & \operatorname{var} x_2 \end{bmatrix}.$$

All elements are nonnegative. When $\operatorname{cov}(x_1, x_2) = 0$, $\mathbf{\Sigma}$ is diagonal and the data varies most along x_i with the largest variance. This no longer holds if x_1 and x_2 are correlated. Consider the following example data.

```
set.seed(1)
n <- 100
x1 <- rnorm(n)
x2 <- .5*x1 + rnorm(n)
cov(data.frame(x1 = x1, x2 = x2))
```

```
##          x1      x2
## x1 0.80676 0.40253
## x2 0.40253 1.11837
```

```
plot(x1, x2); abline(0, .5)  # Original data

angle <- -pi/8
R <- matrix(c(cos(angle), -sin(angle), sin(angle), cos(angle)), 2, 2)
plot(as.matrix(data.frame(x1 = x1, x2 = x2)) %*% R, xlab = 'u1', ylab = 'u2')
abline(0, 0)  # Rotated data
```



In the left plot, if we projected the points onto the line, the variance of the projected data is larger than both $\operatorname{var} x_1$ and $\operatorname{var} x_2$. The projections are the values of u_1 in the right plot; the spread of u_1 is larger than x_1 and x_2. In other words, there exists a basis vector $\widehat{\boldsymbol{u}}_1$ that is a linear combination

of the x_1 and x_2 basis vectors that explains more variance. If we then construct another basis vector $\widehat{\boldsymbol{u}}_2$ orthogonal to $\widehat{\boldsymbol{u}}_1$, in this basis $\boldsymbol{\Sigma}$ is nearly diagonal:

$$\boldsymbol{\Sigma}_u \approx \begin{bmatrix} \operatorname{var} u_1 & 0 \\ 0 & \operatorname{var} u_2 \end{bmatrix}.$$

This change of basis is like a rotation where the rotated variables u_1 and u_2 have little correlation. The principal components are the basis vectors that diagonalize $\boldsymbol{\Sigma}$; roughly speaking, the covariance is rotated away into the diagonal variance, hence the first principal component explains the maximum amount of variation in the data. By definition, the vectors that define a transformation to diagonalize a matrix are the eigenvectors. The most variation corresponds to the largest scaling of the eigenvector, i.e. its eigenvalue.

Exercises

```
library(faraway)
library(simex)
```

The following function used in numerous exercises computes the condition numbers from a `lm()` object.

```
cond_num <- function(lmod) {
  X <- model.matrix(lmod)[, -1]
  e <- eigen(t(X) %*% X)
  return(sqrt(e$values[1] / e$values))
}
```

Exercise 1: Accounting for measurement error using SIMEX. Begin with a simple linear regression of eruption time on waiting time to next eruption.

```
df <- faithful
lmod <- lm(eruptions ~ waiting, df)
sumary(lmod)
```

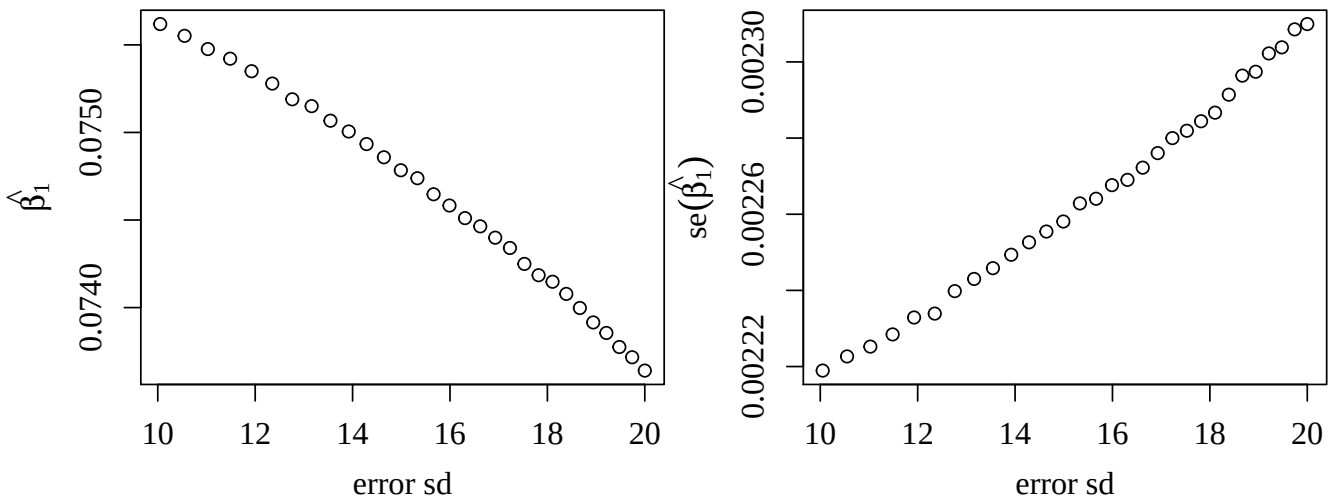
```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.87402    0.16014   -11.7   <2e-16
## waiting      0.07563    0.00222    34.1   <2e-16
##
## n = 272, p = 2, Residual SE = 0.497, R-Squared = 0.81
```

This analysis does not account for measurement error in the waiting times, so the estimate of β_1 may be smaller than an analysis accounting for error. We account for error using SIMEX. Suppose the true error in waiting time is roughly 30 seconds; further assume the error is adequately modeled by a normal distribution with mean 0 and s.d. 10 seconds. We cannot remove the true error and refit a model to obtain unbiased estimates, but we can add more error and compute $\widehat{\beta}_1$. Doing this for a range of errors, we can estimate $\widehat{\beta}_1$ and extrapolate back to zero error. It is more convenient to work with variances rather than s.d., as variances of independent r.v. are additive. Therefore adding normal error to the measurement results in a total error that is normal with mean 0 and variance sum of true and added variances. We do this for a range of s.d. ranging from 10 to 20 seconds (but evenly spaced on the variance scale) and also record the standard error $\operatorname{se}(\widehat{\beta}_1)$.

```
set.seed(1)
n <- nrow(df)
nsim <- 1000
err_var <- seq(1, 20**2 - 10**2, length.out = 30)
results <- array(dim = c(nsim, length(err_var), 2))
for (i in 1:length(err_var)) {
  var <- err_var[i] / 60  # Convert to minutes.
  for (j in 1:nsim) {
    lmodsx <- lm(eruptions ~ I(waiting + rnorm(n, 0, var**.5)), df)
    coef <- summary(lmodsx)$coefficients
    results[j, i, 1] <- coef[2, 'Estimate']
    results[j, i, 2] <- coef[2, 'Std. Error']
  }
}
betas <- apply(results[, , 1], MARGIN = 2, FUN = mean)
ses <- apply(results[, , 2], MARGIN = 2, FUN = mean)
```

Resultant plots show $\widehat{\beta}_1$ decreases as the error increases, as expected, while se$(\widehat{\beta}_1)$ is the reverse.

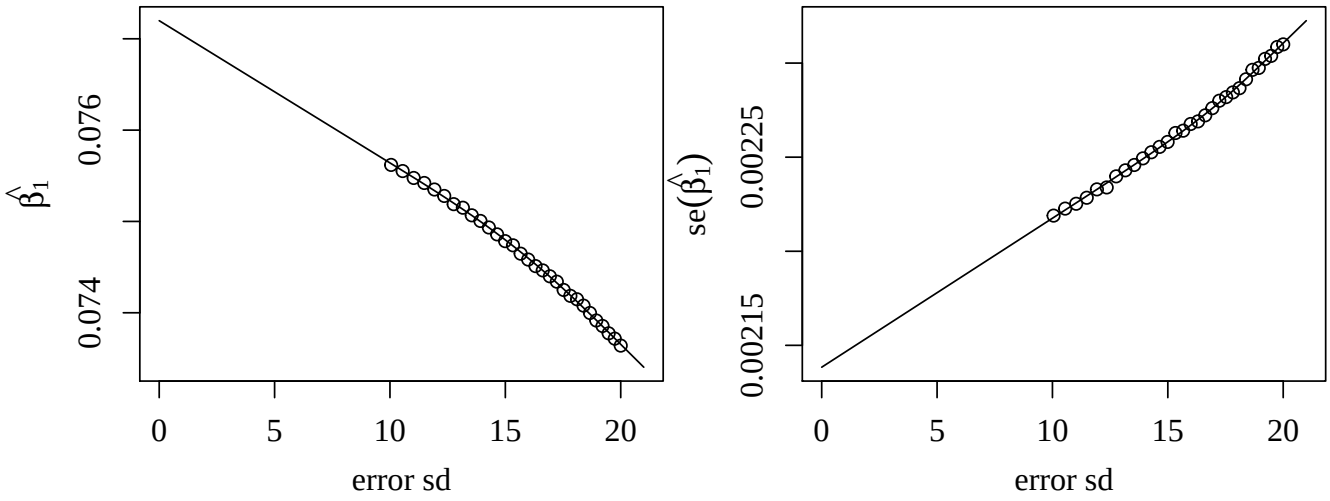
```
err <- (10**2 + err_var)**.5
plot(err, betas, xlab = 'error sd', ylab = expr(hat(beta[1])))
plot(err, ses, xlab = 'error sd', ylab = expr(se(hat(beta[1]))))
```



To extrapolate to zero error, we fit models using `lm()`. Both plot exhibits signs of nonlinearity, evidenced by a residuals vs. fitted plot of the models `lm(betas ~ err)` and `lm(ses ~ err)`. Furthermore, because the extrapolation is quite far, we use natural cubic splines instead of polynomial fits which force the tails of the fit to be linear, hence more stable. Fits are overlaid, with the scale stretched to the extrapolation point of zero measurement error.

```
library(splines)
lmodbeta <- lm(betas ~ ns(err, df = 3))
lmodse <- lm(ses ~ ns(err, df = 3))
egrid <- seq(0, 21, .01)
betahat <- predict(lmodbeta, newdata = data.frame(err = egrid))
sehat <- predict(lmodse, newdata = data.frame(err = egrid))
plot(egrid, betahat, type = 'l', xlab = 'error sd', ylab = expr(hat(beta[1])))
```

```
points(err, betas)
plot(egrid, sehat, type = 'l', xlab = 'error sd', ylab = expr(se(hat(beta[1]))))
points(err, ses)
```



The predicted estimates are

```
predict(lmodbeta, newdata = data.frame(err = 0))  # betahat
```

```
##        1
## 0.077198
```

```
predict(lmodse, newdata = data.frame(err = 0))  # se(betahat)
```

```
##         1
## 0.0021384
```

There is little change from the original model ignoring measurement error. This result is not too surprising, as the measurement error is small relative to a typical value: the median waiting time is 4560 seconds. Our estimates differ slightly from the `simex()` function, most likely because the extrapolation is heavily influenced by the linear fit on the left tail.

```
lmod <- lm(eruptions ~ waiting, df, x = TRUE)
simout <- simex(lmod, "waiting", 10/60, B=1000)
summary(simout)
```

```
## Call:
## simex(model = lmod, SIMEXvariable = "waiting", measurement.error = 10/60,
##     B = 1000)
##
## Naive model:
## lm(formula = eruptions ~ waiting, data = df, x = TRUE)
##
## Simex variable :
##                      waiting
## Measurement error : 0.16667
##
##
## Number of iterations:  1000
```

```
##
## Residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.29895 -0.37646  0.03535  0.00007  0.34898  1.19325
##
## Coefficients:
##
## Asymptotic variance:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.87561    0.13550   -13.8   <2e-16 ***
## waiting      0.07565    0.00191    39.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Jackknife variance:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.87561    0.16018   -11.7   <2e-16 ***
## waiting      0.07565    0.00222    34.1   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note. The book incorrectly uses the variance of the errors as the `measurement.error` parameter in `simex()`, but it should be the standard deviation of measurement error, as I used in this exercise.

Exercise 2: SIMEX accounting for measurement error on the response. In short, $\widehat{\beta}_1$ is still an unbiased estimator but $\text{se}(\widehat{\beta}_1)$ without SIMEX is larger as the estimate of residual variance will be larger. To see this, recall the regression model with no measurement error is

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon.$$

Supposing there is measurement error in y, this becomes

$$y_i' = \beta_0 + \beta_1 x_i + \varepsilon + \varepsilon'.$$

Assuming the measurement errors ε' are independent of ε, we can combine them into a new error term $\varepsilon'' = \varepsilon + \varepsilon'$ with larger variance $\operatorname{var}\varepsilon'' = \operatorname{var}\varepsilon + \operatorname{var}\varepsilon'$. SIMEX can extrapolate an estimate for $\widehat{\beta}_1$ where $\operatorname{var}\varepsilon' \approx 0$, hence the SIMEX estimate will have smaller s.e., observed below.

```
simout <- simex(lmod, "eruptions", 10/60, B=1000)
summary(simout)
```

```
## Call:
## simex(model = lmod, SIMEXvariable = "eruptions", measurement.error = 10/60,
##     B = 1000)
##
## Naive model:
## lm(formula = eruptions ~ waiting, data = df, x = TRUE)
##
## Simex variable :
##                      eruptions
## Measurement error :   0.16667
##
##
```

```
## Number of iterations:  1000
##
## Residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.29815 -0.37630  0.03602  0.00133  0.35079  1.19484
##
## Coefficients:
##
## Asymptotic variance:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.87228    0.13568   -13.8   <2e-16 ***
## waiting      0.07558    0.00191    39.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Jackknife variance:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.87228    0.16126   -11.6   <2e-16 ***
## waiting      0.07558    0.00223    33.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The point estimate remains very similar.

Exercise 3: Investigating collinearity in the `divusa` data.

(a) A linear model's partial tests show `femlab`, `marriage`, and `birth` are statistically significant and the direction of coefficient estimates make intuitive sense. A female in the work force has more independence, so it is plausible divorces increase as more females work. Since divorce cannot occur without marriage, it makes sense more marriages lead to more divorces. Finally, it makes sense that births tend to be correlated with successful marriages, hence more births correlate to fewer divorces.

```
df <- divusa
lmod <- lm(divorce ~ . - year, df)
sumary(lmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.4878     3.3938    0.73    0.466
## unemployed   -0.1113     0.0559   -1.99    0.051
## femlab        0.3836     0.0306   12.54  < 2e-16
## marriage      0.1187     0.0244    4.86  6.8e-06
## birth        -0.1300     0.0156   -8.33  4.0e-12
## military     -0.0267     0.0142   -1.88    0.065
##
## n = 77, p = 6, Residual SE = 1.650, R-Squared = 0.92
```

No condition number is greater than 30, so there is no evidence for severe collinearity.

```
cond_num(lmod)
```

```
## [1]  1.0000  7.4327  8.5325 13.7573 25.1508
```

(b) Examination of the VIFs agree with conclusions from the condition numbers. No VIF is greater

than 5, again suggesting there is no evidence for severe collinearity.

```
vif(lmod)
```

```
## unemployed     femlab   marriage      birth   military
##     2.2529     3.6133     2.8649     2.5855     1.2496
```

(c) Consider a model with only the significant predictors.

```
lmod_red <- lm(divorce ~ femlab + marriage + birth, df)
sumary(lmod_red)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.5455     2.2125   -0.70     0.49
## femlab        0.4134     0.0227   18.17  < 2e-16
## marriage      0.1261     0.0220    5.73  2.1e-07
## birth        -0.1163     0.0141   -8.23  5.1e-12
##
## n = 77, p = 4, Residual SE = 1.695, R-Squared = 0.91
```

The overall R^2 is similar and the estimates and their standard errors are comparable. Examination of the condition numbers and VIFs show the collinearity is reduced somewhat.

```
cond_num(lmod_red)
```

```
## [1]  1.000  7.432 13.660
```

```
vif(lmod_red)
```

```
##   femlab marriage    birth
##   1.8934   2.2019   2.0085
```

Exercise 8: Investigating collinearity with outliers.

(a) The `fat` dataset aims to estimate body fat accurately from easier body measurements, for instance height, weight, and abdomen circumference.

```
lmod <- lm(
  brozek ~ (
    age + weight + height + neck + chest + abdom + hip + thigh + knee
    + ankle + biceps + forearm + wrist
  ),
  fat
)
```

We expect many bodily measurements to be correlated, so we check for collinearity by examining the condition numbers

```
cond_num(lmod)
```

```
##  [1]   1.000  17.471  25.305  58.606  83.591 100.632 137.897 175.286
##  [9] 192.614 213.007 228.157 268.206 555.671
```

and variance inflation factors

```
vif(lmod)
```

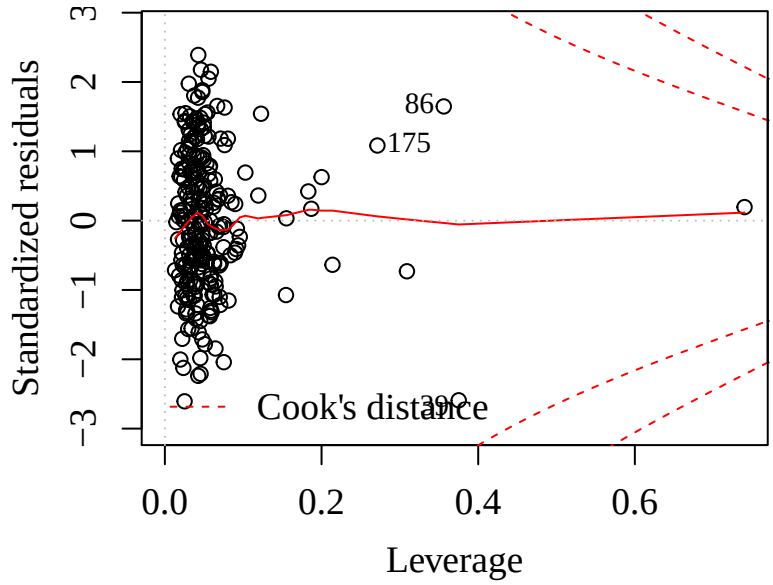
```
##     age  weight  height    neck   chest   abdom     hip   thigh
```

```
##  2.2505 33.5093  1.6746  4.3245  9.4609 11.7671 14.7965  7.7779
##    knee   ankle  biceps forearm   wrist
##  4.6121  1.9080  3.6197  2.1925  3.3775
```

Both metrics suggest collinearity is a major issue. The condition numbers contain many large gaps, suggesting multiple collinear linear combinations.

(b) Recall from Exercise 4.5(c), cases 39 and 42 have abnormal values of weight and height, respectively. Using diagnostic plots from Chapter 6, we can identify these values from the residuals vs. leverage plot.

```
plot(lmod, which = 5)
```



The largest leverage point (unlabeled) corresponds to case 42, while case 39 has the largest Cook's distance. Removing these points and recomputing collinearity diagnostics:

```
lmod2 <- update(lmod, subset = -c(39, 42))
cond_num(lmod2)
```

```
##  [1]   1.000  18.398  26.215  61.532  91.076 114.448 148.725 178.809
##  [9] 202.087 211.784 240.695 276.350 554.798
```

```
vif(lmod2)
```

```
##     age  weight  height    neck   chest   abdom     hip   thigh
##  2.2782 45.2988  3.4396  3.9789 10.7125 11.9676 12.1462  7.1537
##    knee   ankle  biceps forearm   wrist
##  4.4418  1.8103  3.4095  2.4229  3.2637
```

suggest collinearity is just as severe.

(c) To avoid collinearity issues, we will fit a simplified model reintroducing the two outliers.

```
lmod_red <- lm(brozek ~ age + weight + height, fat)
cond_num(lmod_red)
```

```
## [1]  1.000 13.512 22.673
```

```
vif(lmod_red)
```

```
##    age weight height
## 1.0323 1.1070 1.1405
```

The diagnostics indicate collinearity is no longer mch of a concern. However, examination of the model R^2 shows the fit suffers noticeably.

```
summary(lmod2)$r.squared
```

```
## [1] 0.75033
```

```
summary(lmod_red)$r.squared
```

```
## [1] 0.52363
```

Detailed examination of `lmod2` summary shows `abdom` is the most significant variable with a large effect size, so a better model that does not suffer from collinearity (which we will not use for the later parts) replaces `weight` with `abdom` (which is strongly correlated with `weight`).

```
lmod_red2 <- lm(brozek ~ age + abdom + height, fat)
cond_num(lmod_red2)
```

```
## [1]  1.000 10.751 18.255
```

```
vif(lmod_red2)
```

```
##    age  abdom height
## 1.0991 1.0750 1.0489
```

```
summary(lmod2)$r.squared
```

```
## [1] 0.75033
```

```
summary(lmod_red2)$r.squared
```

```
## [1] 0.69358
```

(d) A prediction interval on the reduced model for a median person is below.

```
x0 <- data.frame(
  intercept = 1,
  age = median(fat$age),
  weight = median(fat$weight),
  height = median(fat$height)
)
predict(lmod_red, newdata = x0, interval = 'pred', level = .95)
```

```
##      fit    lwr    upr
## 1 18.281 7.6596 28.903
```

For comparison with (e) and (f), we quantify how unusual `x0` is by appending it to the data and comparing its leverage to the mean leverage, using the rule of thumb that a point with 2 times the leverage may be unusual.

```
leverages <- function(X) {
  # Compute leverages given design matrix.
  H <- X %*% (solve(t(X) %*% X)) %*% t(X)
  return(diag(H))
```

```
}

X <- model.matrix(lmod_red); colnames(X)[1] <- 'intercept'
X <- as.matrix(rbind(X, x0))
levs <- leverages(X)
2*mean(levs)
```

```
## [1] 0.031621
```

```
levs[nrow(fat)+1]
```

```
##      1100
## 0.0040765
```

As expected, a median observation does not have unusual leverage.

(e) The interval width is very similar and the leverage is still reasonably small.

```
x0 <- data.frame(intercept = 1, age = 40, weight = 200, height = 73)
predict(lmod_red, newdata = x0, interval = 'pred', level = .95)
```

```
##      fit    lwr    upr
## 1 20.479 9.8378 31.119
```

```
X <- model.matrix(lmod_red); colnames(X)[1] <- 'intercept'
X <- as.matrix(rbind(X, x0))
levs <- leverages(X)
2*mean(levs)
```

```
## [1] 0.031621
```

```
levs[nrow(fat)+1]
```

```
##      1100
## 0.0076373
```

(f) The interval width still is similar, although this interval is misleading as bodyfat cannot be negative.

```
x0 <- data.frame(intercept = 1, age = 40, weight = 130, height = 73)
predict(lmod_red, newdata = x0, interval = 'pred', level = .95)
```

```
##      fit     lwr    upr
## 1 7.6174 -3.1011 18.336
```

This point has higher leverage, making the prediction more unstable.

```
X <- model.matrix(lmod_red); colnames(X)[1] <- 'intercept'
X <- as.matrix(rbind(X, x0))
levs <- leverages(X)
2*mean(levs)
```

```
## [1] 0.031621
```

```
levs[nrow(fat)+1]
```

```
##     1100
```

```
## 0.021978
```

While its weight and height alone are not unusual, because of the positive correlation between weight and height, the given pair is a farther extrapolation. (To see this, a scatterplot of `height` against `weight` is sufficient.)

Chapter 8

Problems with the Error

8.1 Generalized Least Squares

8.2 Weighted Least Squares

8.3 Testing for Lack of Fit

Comparing a model to a saturated one using anova(). The saturated model fitting means of y to each group x yields the smallest residual sum of squares (RSS) for models containing only x or functions of x. As such, any model containing only x is nested in the sense that it will not have a better RSS. Thus, `anova()` can be used to compare models even though there is no obvious nested structure, e.g. $y \sim x$ is obviously nested to $y \sim x + x^2$. For more discussion, see AdamO's answer on stats.stackexchange.

8.4 Robust Regression

8.4.1 M-Estimation

8.4.2 Least Trimmed Squares

Which method to use and why? Consider a linear regression model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where $\boldsymbol{y}$ and $\boldsymbol{X}$ are the data (response and predictors), $\boldsymbol{\beta}$ are the parameters we wish to estimate to determine the relationship between $\boldsymbol{y}$ and $\boldsymbol{X}$, and $\boldsymbol{\varepsilon}$ are random variable errors representing all unaccounted effects. When the true errors $\boldsymbol{\varepsilon}$ are independent and identically distributed (i.i.d., i.e. uncorrelated with equal variance), **ordinary least squares** (OLS) for the parameter estimates $\widehat{\boldsymbol{\beta}}$ is the best linear unbiased estimate (BLUE) by the Gauss–Markov theorem. The OLS solution is $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$. If $\boldsymbol{\varepsilon}$ are normally distributed, $\boldsymbol{y}$ are normally distributed so $\widehat{\boldsymbol{\beta}}$ are normally distributed, and we can proceed with parametric inference.[1] These assumptions on the errors can

[1] If the errors are not normally distributed, $\widehat{\boldsymbol{\beta}}$ are asymptotically normal. However, what constitutes a large enough sample size is difficult to determine so nonparametric estimates like the bootstrap should be used for comparison.

be checked using diagnostics discussed in Chapter 6. Under mild deviations, OLS estimates are still acceptable. In fact, it is straightforward to check $\widehat{\boldsymbol{\beta}}$ is unbiased as long as the assumption $\mathbb{E}\,\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}$ holds, without assuming anything about the errors:

$$\mathbb{E}\,\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T \underbrace{\mathbb{E}\,\boldsymbol{y}}_{=\boldsymbol{X}\boldsymbol{\beta}} = \boldsymbol{\beta}.$$

However, **generalized least squares** (GLS) may be preferred, especially when the i.i.d. assumptions are violated strongly; while the OLS estimate is also unbiased, its variance may be larger.

In OLS, the i.i.d. error assumption implies the covariance matrix of the errors

$$\operatorname{cov}(\boldsymbol{\varepsilon}) \equiv \boldsymbol{\Sigma} = \sigma^2\boldsymbol{I},$$

where $\sigma^2 = \operatorname{var}\varepsilon_i$, $i = 1, 2, \ldots, n$, and $\boldsymbol{I}$ is the identity matrix. In general with no assumptions on the error, $\boldsymbol{\Sigma}$ is any symmetric, positive semi-definite matrix. Supposing $\boldsymbol{\Sigma}$ is known, GLS decomposes $\boldsymbol{\Sigma} = \boldsymbol{S}\boldsymbol{S}^T$ and transforms the regression model into

$$\boldsymbol{y}' = \boldsymbol{X}'\boldsymbol{\beta} + \boldsymbol{\varepsilon}'$$

where the primes indicate left matrix multiplication by $\boldsymbol{S}^{-1}$. In this transformed regression model, the covariance matrix of the residuals becomes the identity:

$$\operatorname{cov}\boldsymbol{\varepsilon}' = \operatorname{cov}(\boldsymbol{S}^{-1}\boldsymbol{\varepsilon}) = \boldsymbol{S}^{-1}(\operatorname{cov}\boldsymbol{\varepsilon})(\boldsymbol{S}^T)^{-1} = \boldsymbol{S}^{-1}\underbrace{(\boldsymbol{\Sigma})}_{=\boldsymbol{S}\boldsymbol{S}^T}(\boldsymbol{S}^T)^{-1} = \boldsymbol{I}.$$

Thus, we can obtain the BLUE for $\boldsymbol{\beta}$ by applying OLS on the transformed regression model. Explicitly, $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{y}$ and $\operatorname{var}\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{\Sigma}\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}$; $\widehat{\boldsymbol{\beta}}$ is unbiased with smallest variance among all other linear estimators. In the case of multivariate normal errors $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$, the GLS estimates $\widehat{\boldsymbol{\beta}}$, i.e. OLS estimates on the transformed regression model, are normally distributed and we can again proceed with parametric inference. (Footnote 1 still applies.)

So why not always use GLS? The problem is that the covariance structure of the residuals $\boldsymbol{\Sigma}$ is rarely known and is difficult to estimate from the data because there are over $n^2/2$ elements of $\boldsymbol{\Sigma}$ that require estimation.[2] In practice, GLS requires structural assumptions on the form of $\boldsymbol{\Sigma}$ to drastically reduce the number of parameters to estimate, for instance the book's example modeling serial correlations in time series data. An implementable version of GLS that estimates $\boldsymbol{\Sigma}$ is known as feasible generalized least squares which no longer has the optimal properties GLS does. Indeed for finite samples, OLS may be preferable even when the errors are known to not be i.i.d., see Wikipedia and stats.stackexchange for more discussion.

A useful compromise between OLS and GLS is known as **weighted least squares** (WLS). WLS assumes the errors are independent but can have unequal variance, i.e. heteroscedasticity. This is a special case of GLS where $\boldsymbol{\Sigma}$ is diagonal with n possibly distinct variances to estimate, closely related to the so called regression weights. To see the effect of weighted regression, in the GLS framework $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$, so $\boldsymbol{S} = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ and $\boldsymbol{S}^{-1} = \operatorname{diag}(1/\sigma_1, 1/\sigma_2, \ldots, 1/\sigma_n)$. It is straightforward to verify the transformed regression model $\boldsymbol{y}' = \boldsymbol{X}'\boldsymbol{\beta} + \boldsymbol{\varepsilon}'$ in component form is

$$y_i/\sigma_i = (\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_2 x_2)/\sigma_i + \varepsilon_i/\sigma_i, \qquad i = 1, 2, \ldots, n.$$

[2] Diagnostics discussed in Chapter 6 for an OLS regression can indicate violations to the i.i.d. error assumptions, but it is difficult to learn how the errors are generated.

It follows that least squares seeks to minimize the sum of the squared transformed errors

$$\sum_{i=1}^{n}(\varepsilon_i/\sigma_i)^2 = \sum_{i=1}^{n}\frac{1}{\sigma_i^2}\left(y_i - \sum_{j=0}^{p}\beta_j x_j\right)^2,$$

where $x_0 = 1$. OLS weighs all observations $i = 1, 2, \ldots, n$ equally. In WLS, large variance observations have less influence over the fit (less weight) while small variance observations have more influence over the fit (more weight). The weights are formally defined as $w_i = 1/\sigma_i^2$ so that $\boldsymbol{\Sigma} = \operatorname{diag}(1/w_1, 1/w_2, \ldots, 1/w_n)$. If the variances and thus weights are known, then from the GLS formalism discussed previously, the $\widehat{\boldsymbol{\beta}}$ estimates are BLUE. In practice, the n variances must be estimated. We can reduce the number of parameters to estimate by modeling the variances as done in the book:

$$\operatorname{var}\varepsilon_i = (\gamma_0 + \gamma_1 x_1^{\delta_1} + \gamma_2 x_2^{\delta_2} + \cdots + \gamma_p x_p^{\delta_p})^2.$$

The weights are then $w_i = 1/\widehat{\operatorname{var}\varepsilon_i}$. This is a nonlinear regression model. Coefficients in front of the x_j are needed when $p > 1$ because they cannot be absorbed into γ_0 by scaling by a constant. As we do not directly observe $\operatorname{var}\varepsilon_i$, we use an estimate of the variance of the ith error as the response values in the nonlinear regression. One sensible estimate uses the squared residuals $\widehat{\varepsilon}_i^2$ from an OLS fit. This is because $\mathbb{E}\,\widehat{\varepsilon}_i^2 = \operatorname{var}\widehat{\varepsilon}_i + (\mathbb{E}\,\widehat{\varepsilon}_i)^2 = \operatorname{var}\widehat{\varepsilon}_i$, since OLS residuals have zero mean. Other more robust estimates accounting for outliers and high-leverage points can be used, e.g. standardized residuals. See gung's answer on stats.stackexchange for more discussion.

An alternate way to address unequal variance (assuming independence of errors) without pre-specifying the weights uses OLS with **robust standard errors**, often referred to as a sandwich estimate of $\operatorname{var}\widehat{\boldsymbol{\beta}}$ or heteroscedasticity-consistent standard errors. We showed earlier that $\widehat{\boldsymbol{\beta}}$ obtained from OLS is unbiased, but the variance obtained may not be accurate since $\operatorname{var}\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{\Sigma}\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}$. The sandwich estimate uses the residuals $\widehat{\varepsilon}_i$ of an OLS fit to estimate the covariance matrix $\widehat{\boldsymbol{\Sigma}} = \operatorname{diag}(\widehat{\varepsilon}_1^2, \widehat{\varepsilon}_2^2, \ldots, \widehat{\varepsilon}_n^2)$. The parameter estimate variances are then given by

$$\widehat{\operatorname{var}\widehat{\boldsymbol{\beta}}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\widehat{\boldsymbol{\Sigma}}\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}.$$

While robust standard errors may lead to better estimates of $\operatorname{var}\widehat{\boldsymbol{\beta}}$, they do not transform the model so that estimates are BLUE nor do they do not indicate where the model is problematic. In contrast, when using WLS estimates are BLUE, assuming the correct transformation is specified, and one can examine the usual diagnostics on the transformed regression model. Finally, note that robust standard errors are computed after the model is fit, rather than before as in WLS. Consequently, we could apply WLS with robust standard errors, but this is not necessarily better. Applying both corrections estimates more parameters, and as such the corrected standard errors may have less bias but more variance; for further discussion, see stats.stackexchange.

We should also mention **transformation of the response**, as discussed in Section 6.1.1, as a way to account for heteroscedasticity. Similar to WLS, the goal is to find a transformation so that the transformed data meets the assumptions of OLS, where estimates are BLUE. While it can be hard to find such a variance-stabilizing transformation, this method has the additional benefit that non-normal errors can be transformed to normality, e.g. a log transformation for log-normal errors. Transformation of the response can be used in conjunction with WLS and robust standard errors, i.e. transform y, run a weighted regression, and obtain robust standard errors. Again, more complexity is not necessarily better.

One final way to obtain accurate standard errors with potential heteroscedasticity is to use the **bootstrap** on OLS estimates of $\widehat{\boldsymbol{\beta}}$. This method is sensible because the OLS estimates are unbiased and the bootstrap estimates standard errors without making assumptions on the errors. The

bootstrap is also useful for inference when the normality assumption is suspect after correcting for heteroscedasticity.

So far we've addressed how to extend OLS when the errors are not independent and do not have equal variance. In practice, we'll also observe outliers and high influence points that can drastically affect the fit, and thus accuracy of estimates. This is where **robust regression** is useful. Robust regression applies to both OLS and GLS, since GLS is an OLS applied to transformed data, assuming such a transformation can be estimated (which, as discussed previously, is a difficult in general). For example, a WLS aims to remove heteroscedasticity by weighting, but after doing so may still contain some unusual observations warranting use of robust regression of $\boldsymbol{y}'$ against $\boldsymbol{x}'$.

M-estimation modifies the function of residuals to be minimized, for instance minimizing the sum of absolute values of errors rather than sum of squares. This effectively introduces a weighted regression; if a WLS was already used to address heteroscedasticity with weights w_i, M-estimation introduces another set of weights v_i so that the overall weight is the product $w_i v_i$. Huber's method is intuitively appealing as it only weights observations deemed unusual, based on a robust estimate of the residual standard error. It can be shown that M-estimates, like OLS estimates, are asymptotically normally distributed (see Wikipedia) allowing for parametric inference given a large enough sample. As a nonparametric alternative, we can repeatedly apply robust regression to bootstrapped samples.

One limitation of M-estimation is that it does not downweight large leverage points. This is seen in the two book examples, wear the M-estimation yielded very similar fits to OLS. **Least trimmed squares** addresses this limitation and can identify clusters of outliers by finding the subset of the data minimizing the sum of squares. Here, the bootstrap is necessary for conducting inference as there is no theoretical foundation for the distribution of $\widehat{\boldsymbol{\beta}}$ based on the 'best' subset of observed data.

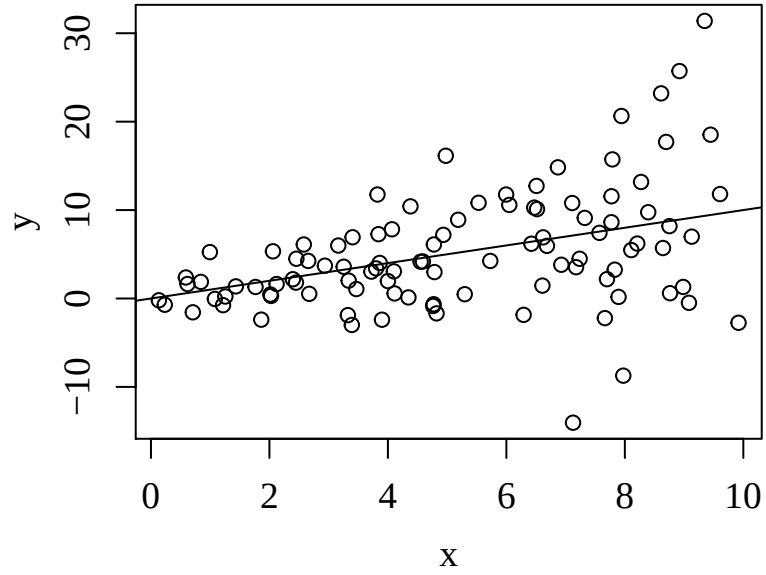
For other types of robust regression, see Wikipedia.

Case study: heteroscedastic data.

```
library(faraway)
library(nlme)  # For GLS.
library(sandwich)  # For robust standard errors.
library(MASS)  # For robust regression.
```

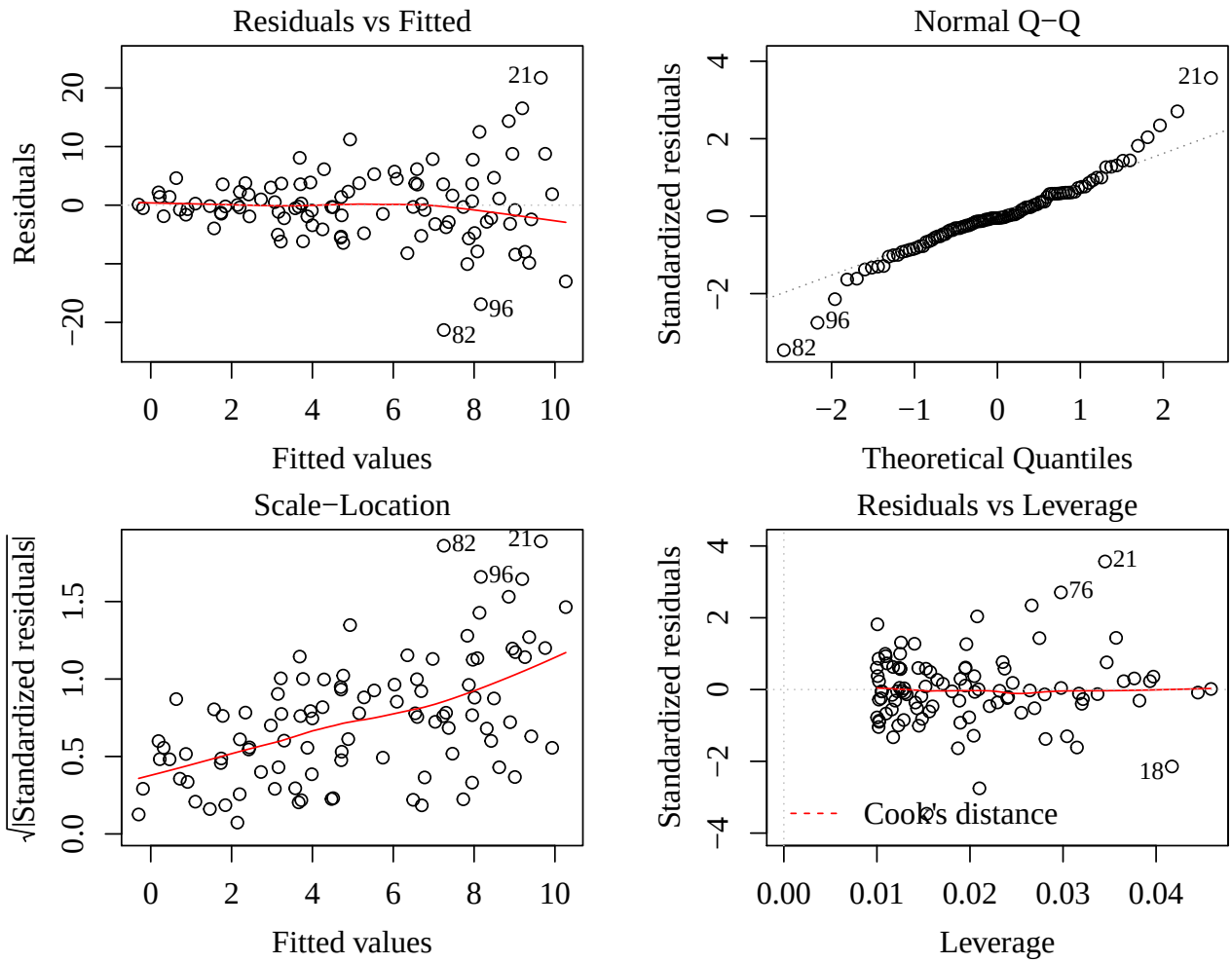
Suppose the data is generated by $y = x + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, 1 + x^2)$. A scatterplot of the data shows strong heteroscedasticity is present, where the true $\mathbb{E}\, y = x$ is overlaid.

```
set.seed(1)
n <- 100
x <- runif(n, 0, 10)
epsvar <- x**2
y <- x + rnorm(n) + rnorm(n, 0, epsvar**.5)
plot(x, y)
abline(0, 1)
```



We begin with a simple OLS model and its diagnostic plots.

```
lmod <- lm(y ~ x)
plot(lmod)
```

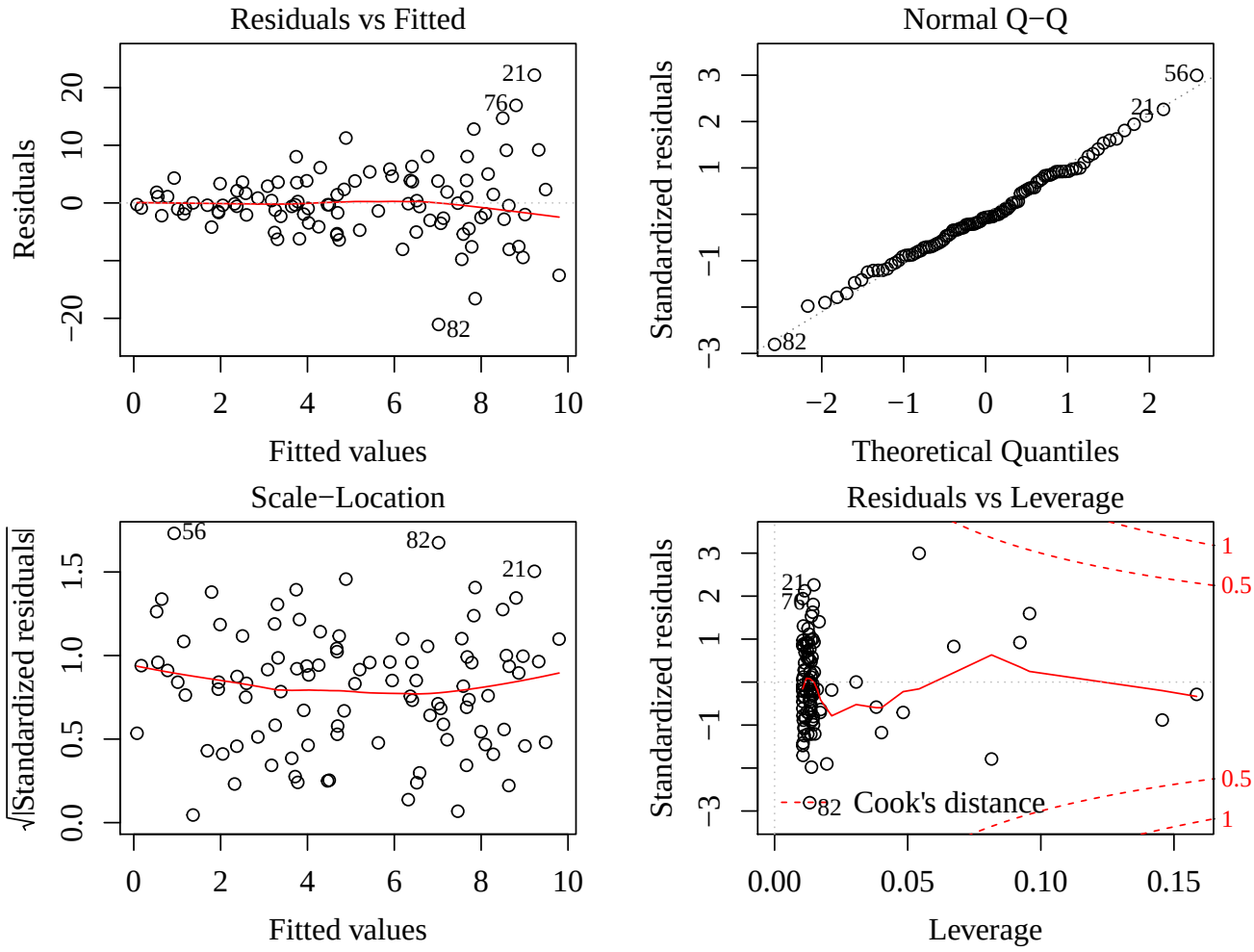


Signs of heteroscedasticity are indicated in all plots, although other OLS assumptions don't appear strongly violated.

1. **Residuals vs Fitted.** $\mathbb{E}\,\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}$ appears reasonable, but it is difficult to tell due to the fanning out of residuals, indicating heteroscedasticity.
2. **Normal Q-Q.** The residuals are not normally distributed with constant variance particularly at the tails, likely due to heteroscedasticity.
3. **Scale-Location.** The increasing trend line indicates heteroscedasticity is a clear problem.
4. **Residuals vs Leverage.** There are no high leverage points and therefore no strongly influential points, measured by Cook's distance, despite a few large residuals (again, likely due to heteroscedasticity).

Since we know the data generating process, the best case scenario weights are $w_i = 1/\sigma_i^2 = 1/(1+x_i^2)$. This weighted least squares regression with the same diagnostic plots is next.

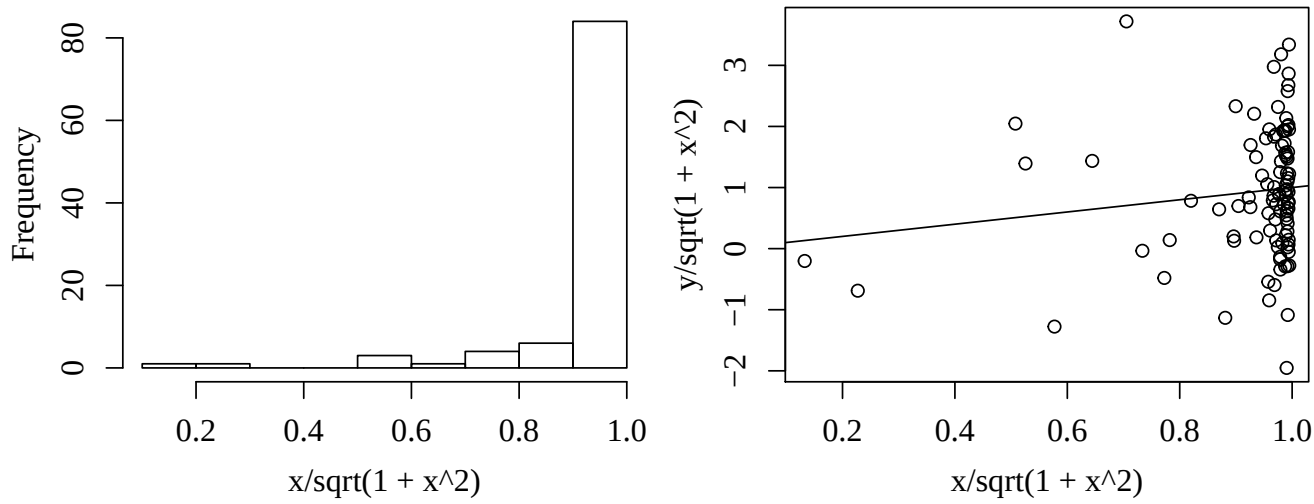
```
wlmod1 <- lm(y ~ x, weights = 1/(1 + epsvar))
plot(wlmod1)
```



1. **Residuals vs Fitted.** There is little change. This makes sense because OLS is also unbiased, while WLS adjusts the variance of estimates.
2. **Normal Q-Q.** The standardized residuals now appear normally distributed.
3. **Scale-Location.** There is little trend. Heteroscedasticity is far less problematic.
4. **Residuals vs Leverage.** The magnitude of standardized residuals is less, but many points

now have high leverage and thus more influence. The increase in leverage can be attributed to weighting. $x \in [0, 10]$ is transformed to $x' \in [0, 1]$ via $x' = x/\sqrt{1+x^2}$. As x' quickly saturates to 1 near $x \approx 2$, most values of x' are close to one. The high leverage points are those with small x (and thus x'); the distribution x' demonstrates this, along with a scatter plot of y' against x'.

```
hist(x / sqrt(1+x**2), main = '')
plot(x / sqrt(1+x**2), y / sqrt(1+x**2))
abline(0, 1)
```



While there are more influential points in the WLS, addressing heteroscedasticity may be worth it. Model summaries reveal tighter standard errors.

```
sumary(lmod)
```

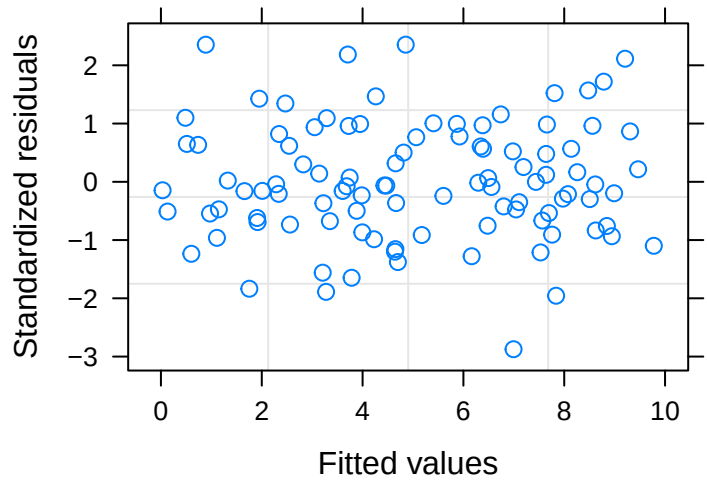
```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.444      1.356   -0.33     0.74
## x              1.080      0.233    4.64  1.1e-05
##
## n = 100, p = 2, Residual SE = 6.202, R-Squared = 0.18
```

```
sumary(wlmod1)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.0589     0.4364   -0.14     0.89
## x             0.9935     0.1568    6.34  7.2e-09
##
## n = 100, p = 2, Residual SE = 1.050, R-Squared = 0.29
```

In practice, we do not know the data generating process; the prior WLS was the best case scenario. We compare a feasible WLS where the error variances are estimated using `gls()` from the `nlme` package.

```
wlmod2 <- gls(y ~ x, weight = varConstPower(1, form = ~ x))
plot(wlmod2)
```



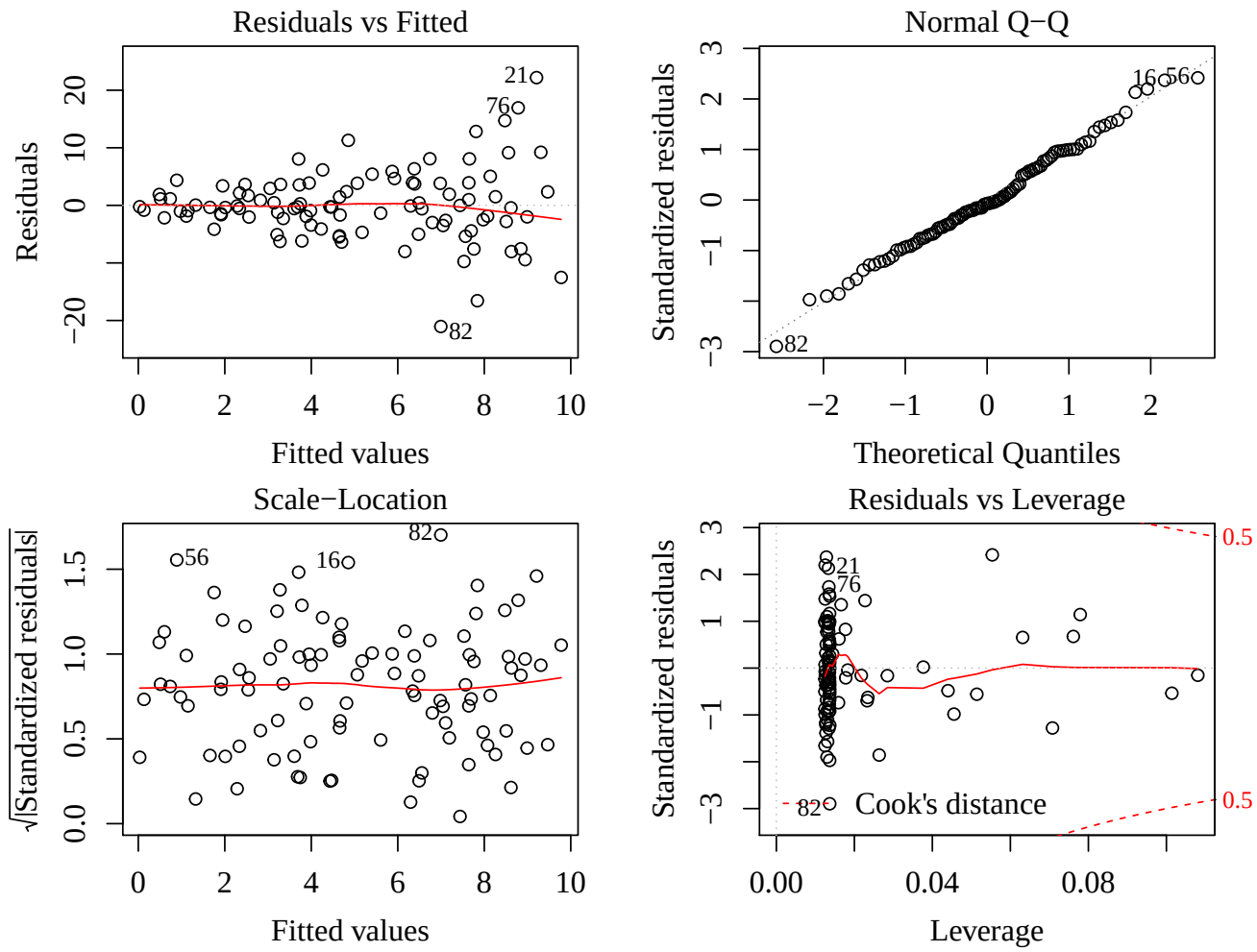
Unfortunately, only the standardized residuals against fitted values plot is shown by default. While it appears heteroscedasticity has been addressed, we cannot examine the other model assumptions in detail. A workaround uses the variance estimation fit to construct the estimated weights, and use `lm()` as previously.

```
summary(wlmod2)
```

```
## Generalized least squares fit by REML
##   Model: y ~ x
##   Data: NULL
##       AIC    BIC  logLik
##   605.15 618.07 -297.57
##
## Variance function:
##  Structure: Constant plus power of variance covariate
##  Formula: ~x
##  Parameter estimates:
##  const  power
## 7.0096 1.6320
##
## Coefficients:
##                Value Std.Error t-value p-value
## (Intercept) -0.10091   0.55361 -0.1823  0.8557
## x            0.99587   0.17002  5.8575  0.0000
##
##  Correlation:
##   (Intr)
## x -0.776
##
## Standardized residuals:
##       Min        Q1       Med        Q3       Max
## -2.873848 -0.667525 -0.070433  0.679313  2.354325
##
## Residual standard error: 0.2313
## Degrees of freedom: 100 total; 98 residual
```

Note that the point estimates are similar, with standard errors in between the OLS and optimal WLS case. The diagnostic plots are generated next.

```
weights2 <- 1 / (7.01 + x**1.63)**2
wlmod2 <- lm(y ~ x, weights = weights2)
plot(wlmod2)
```



They are qualitatively similar to the observations on `wlmod1`.

Next, we see the effect of robust standard errors using the `sandwich()` function from the `sandwich` package. We compute them on both the OLS and optimal WLS fits.

```
diag(sandwich(lmod))**.5
```

```
## (Intercept)           x
##     0.96455     0.26061
```

```
diag(sandwich(wlmod1))**.5
```

```
## (Intercept)           x
##     0.50476     0.15775
```

Depending on the seed, i.e. data drawn, the robust standard error estimates can be larger or smaller. However in general, they differ more for the OLS estimates rather than WLS as we might expect from the strong heteroskedasticity. A more rigorous analysis relies on Monte Carlo simulation of

many data generated from the same process. Before that, we generate estimates and their standard errors using OLS on repeatedly bootstrapped samples.

```
boot_se <- function(lmod, nboot) {
  resid <- lmod$residuals
  yhat <- lmod$fitted.values
  betas <- array(dim = c(nboot, 2))
  x <- lmod$model[, 'x']
  for (i in 1:nboot) {
    yboot <- yhat + sample(resid, replace = TRUE)
    lmod_boot <- lm(yboot ~ x)
    betas[i, 1] <- lmod_boot$coefficients[1]
    betas[i, 2] <- lmod_boot$coefficients[2]
  }
  return(apply(betas, 2, sd))
}

boot_se(lmod, nboot = 1000)
```

```
## [1] 1.33203 0.23343
```

The standard errors are similar to the original OLS.

We've seen different results between various methods (OLS, WLS, robust estimates, bootstrap) for this particular seed and may wonder: which method is the best? Coverage probabilities of confidence intervals provide one useful characterization, particularly for β_x since we often test $H_0 : \beta_x = 0$. If the smaller standard errors of WLS yield smaller coverage probabilities, then they are not necessarily better. The coverage probabilities of OLS and WLS, both with and without robust estimates, are compared for intervals constructed with confidence level 95%.

```
is_covered <- function(lmod, robust = FALSE, alpha = .05) {
  s <- summary(lmod)
  beta <- s$coefficients['x', 1]
  if (robust) {
    se <- sandwich(lmod)['x', 'x']**.5
  } else {
    se <- s$coefficients['x', 2]
  }
  q <- qt(1 - alpha/2, lmod$df.residual)
  return((1 > beta - q*se) & (1 < beta + q*se))
}

nsim <- 10000
counter <- rep(0, 4)
names(counter) <- c('OLS', 'OLSS', 'WLS', 'WLSS')
for (i in 1:nsim) {
  x <- runif(n, 0, 10)
  y <- x + rnorm(n) + rnorm(n, 0, x)
  ols_mod <- lm(y ~ x)
  wls_mod1 <- lm(y ~ x, weight = 1/(1+x**2))

  if (is_covered(ols_mod)) {
```

```
    counter[1] <- counter[1] + 1
  }
  if (is_covered(ols_mod, robust = TRUE)) {
    counter[2] <- counter[2] + 1
  }
  if (is_covered(wls_mod1)) {
    counter[3] <- counter[3] + 1
  }
  if (is_covered(wls_mod1, robust = TRUE)) {
    counter[4] <- counter[4] + 1
  }
}
counter/nsim  # Coverage probabilities.
```

```
##    OLS   OLSS    WLS   WLSS
## 0.9236 0.9395 0.9518 0.9460
```

We observe that OLS intervals do not meet the nominal 95% level, even after adjustment from robust estimates. On the other hand, WLS is satisfactory with slightly reduced performance after adjustment from robust estimates. We conclude that WLS results in the narrowest and most accurate intervals.

Another useful characterization is a confidence interval for the standard error of β_x. Since we know the data generating process, we can compute the 'true' standard errors for OLS and WLS using Monte Carlo simulation.

```
nsim <- 10000
betas <- array(dim = c(nsim, 2), dimnames = list(NULL, c('OLS', 'WLS')))
for (i in 1:nsim) {
  x <- runif(n, 0, 10)
  y <- x + rnorm(n) + rnorm(n, 0, x)
  betas[i, 1] <- lm(y ~ x)$coefficients['x']
  betas[i, 2] <- lm(y ~ x, weights = 1/(1 + x**2))$coefficients['x']
}
(se <- apply(betas, 2, sd))
```

```
##     OLS     WLS
## 0.22126 0.14419
```

We compare 6 methods: OLS, optimal WLS (OWSL), feasible WLS (FWLS), OLS with sandwich estimates (OLSS), optimal WLS with sandwich estimates (WLSS), and the bootstrap.

```
nsim <- 1000
betases <- array(
  dim = c(nsim, 6),
  dimnames = list(
    NULL, c('OLS', 'OWLS', 'FWLS', 'OLSS', 'OWLSS', 'boot')
  )
)
for (i in 1:nsim) {
  x <- runif(n, 0, 10)
  y <- x + rnorm(n) + rnorm(n, 0, x)
```

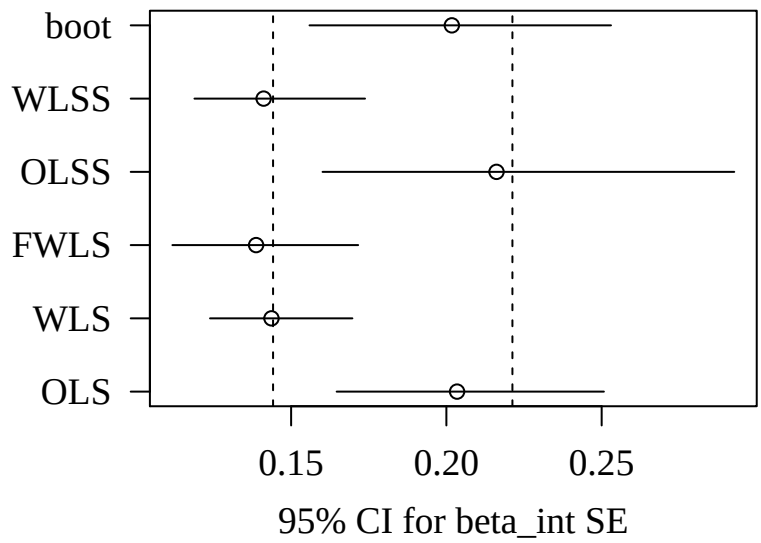
```
  ols_mod <- lm(y ~ x)
  wls_mod1 <- lm(y ~ x, weight = 1/(1+x**2))
  wls_mod2 <- gls(y ~ x, weight = varConstPower(1, form = ~ x))
  betases[i, 1] <- summary(ols_mod)$coefficients['x', 2]
  betases[i, 2] <- summary(wls_mod1)$coefficients['x', 2]
  betases[i, 3] <- summary(wls_mod2)$tTable['x', 2]
  betases[i, 4] <- sandwich(ols_mod)['x', 'x']**.5
  betases[i, 5] <- sandwich(wls_mod1)['x', 'x']**.5
  betases[i, 6] <- boot_se(ols_mod, nboot = 200)[2]
}
(mean <- apply(betases, 2, mean))
```

```
##      OLS     OWLS     FWLS     OLSS    OWLSS     boot
## 0.20344 0.14367 0.13873 0.21614 0.14115 0.20176
```

```
ci <- apply(betases, 2, function(x) quantile(x, c(.025, .975)))
```

Means and 95% confidence intervals for each method are plotted next.

```
plot(mean, 1:6, xlim = c(min(ci[1, ]), max(ci[2, ])),
     xlab = '95% CI for beta_int SE', ylab = '', yaxt = 'n')
axis(
  2, at = 1:6, labels = c('OLS', 'WLS', 'FWLS', 'OLSS', 'WLSS', 'boot'), las = 1
)
abline(v = se[1], lty = 2); abline(v = se[2], lty = 2)
segments(ci[1, ], 1:6, ci[2, ], 1:6)
```

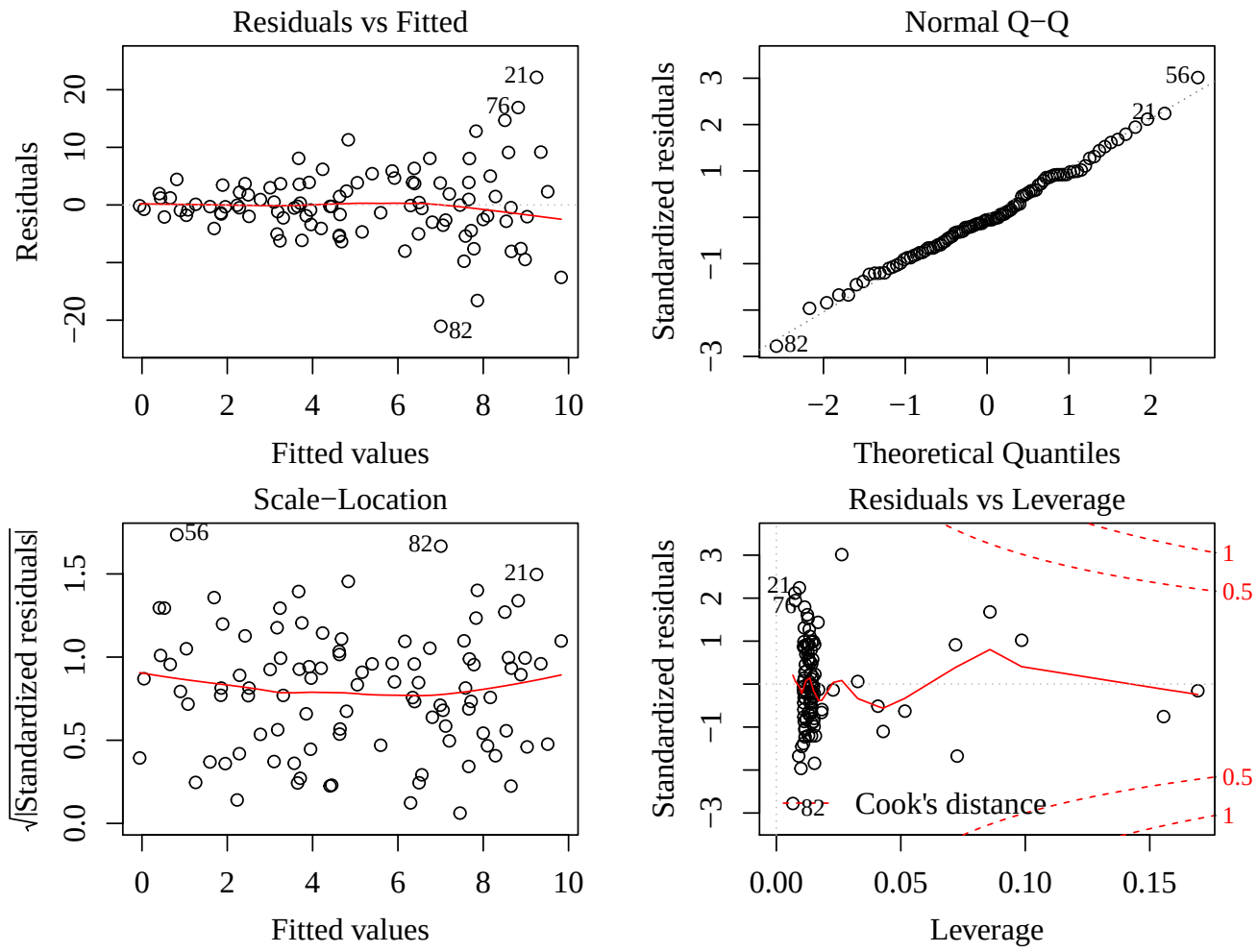


The improvement in standard errors for WLS is substantial, even when the weights are estimated in FWLS, especially because they maintain nominal coverage probabilities when used to construct confidence intervals. The robust estimates reduce the bias of OLS standard error estimates, but also increase their variance (wider interval). As a result, they improve coverage probabilities but are not enough to maintain nominal coverage. Bootstrap estimates are comparable to OLS. Again, WLS or FWLS is preferred.

Finally, we explore robust regression, in particular Huber's M-estimation. We saw from the earlier diagnostics that WLS contained a few large leverage and potentially influential points.

Consequentally, we fit a robust regression using `rlm()` from the `MASS` package.

```
set.seed(1)  # Generate the original data.
n <- 100
x <- runif(n, 0, 10)
epsvar <- x**2
y <- x + rnorm(n) + rnorm(n, 0, epsvar**.5)
rlmod <- rlm(y ~ x, weights = 1/(1 + epsvar))
plot(rlmod)
```



The diagnostic plots are qualitatively similar to the original WLS, as is the model summary:

```
summary(rlmod)
```

```
##
## Call: rlm(formula = y ~ x, weights = 1/(1 + epsvar))
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9262 -0.6936 -0.0588  0.7783  3.1413
##
## Coefficients:
##             Value  Std. Error t value
## (Intercept) -0.189  0.434     -0.435
## x            1.010  0.156      6.477
```

```
##
## Residual standard error: 1.06 on 98 degrees of freedom
```

As there is little discrepancy, we conclude robust regression is not necessary and WLS or FWLS is the best method.

Exercises

```
library(faraway)  # For data and sumary().
library(nlme)  # For generalized least squares gls().
```

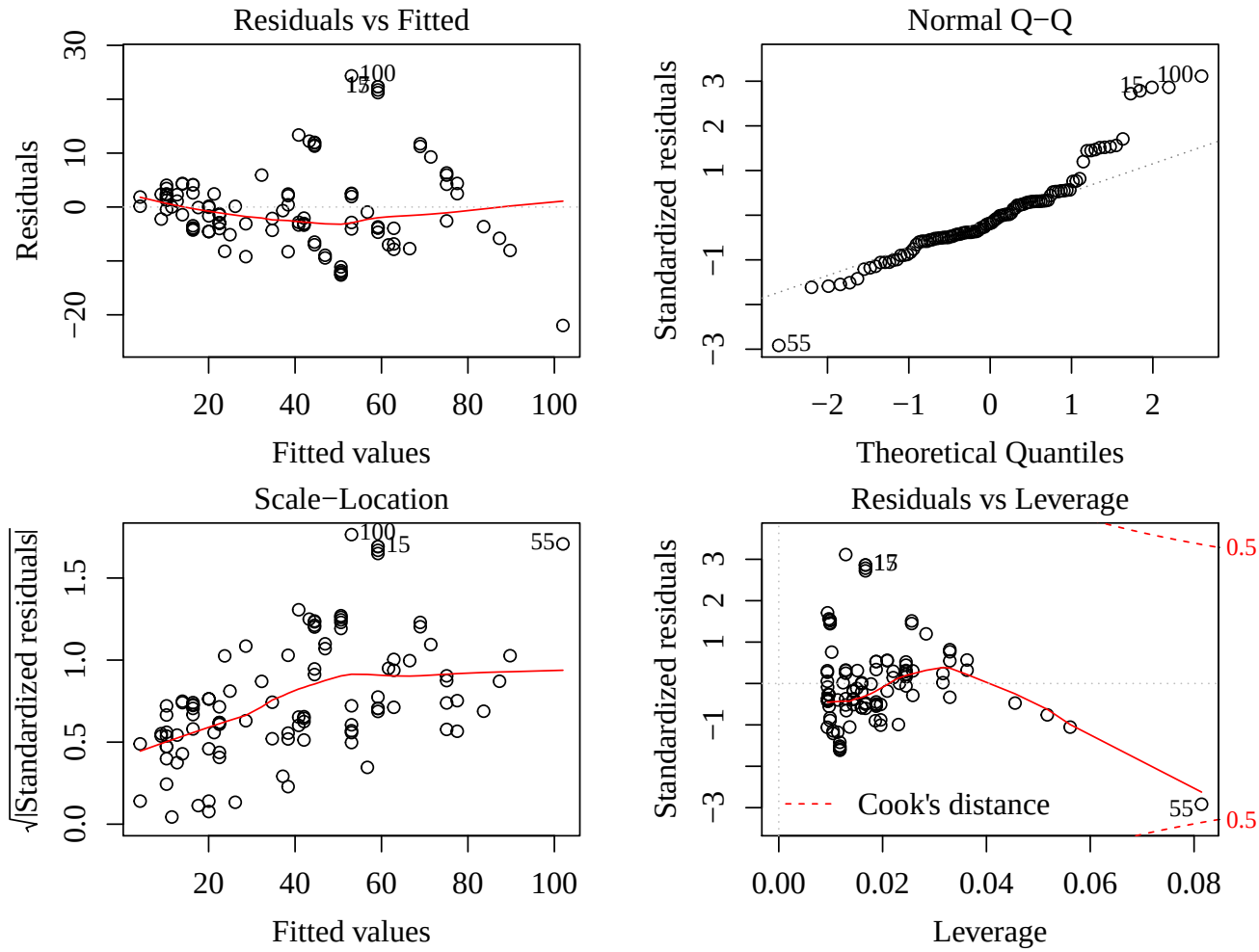
Exercise 1: Multiple methods to address heteroscedasticity.

(a) We fit a linear model `Lab ~ Field` and plot diagnostics.

```
df <- pipeline
lmod <- lm(Lab ~ Field, df)
sumary(lmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.9675     1.5748   -1.25     0.21
## Field         1.2230     0.0411   29.78   <2e-16
##
## n = 107, p = 2, Residual SE = 7.865, R-Squared = 0.89
```

```
plot(lmod)
```



The residuals vs. fitted plot suggests a linear model may not be sufficient and demonstrates heteroscedasticity, the latter being further evidenced in the scale-location plot. The qqplot shows the normality assumption is suspect at the tails, but this may be due to heteroscedasticity. Finally, the residuals vs. leverage plot indicates observation 55 should be examined more closely.

(b) To address heteroscedasticity, we estimate variances (and therefore weights) by grouping the observations into a range of similar values. Assuming members of each group are similar, this improves variance estimates obtained from (squared) residuals. Because the similarity assumption may not be valid, we can improve estimates for each member by modeling the grouped variances as a function of the grouped predictor means, then using this model to predict the ith variance given ith predictor value.

```
i <- order(pipeline$Field)
npipe <- pipeline[i,]
ff <- gl(12,9)[-108]
meanfield <- unlist(lapply(split(npipe$Field,ff),mean))
varlab <- unlist(lapply(split(npipe$Lab,ff),var))
```

Log-transforming the model $\operatorname{var}\texttt{Lab} = a_0\texttt{Field}^{a_1}$ yields approximate linearized model

$$\log(\operatorname{var}\texttt{Lab}) = \log(a_0) + a_1 \log(\texttt{Field}).$$

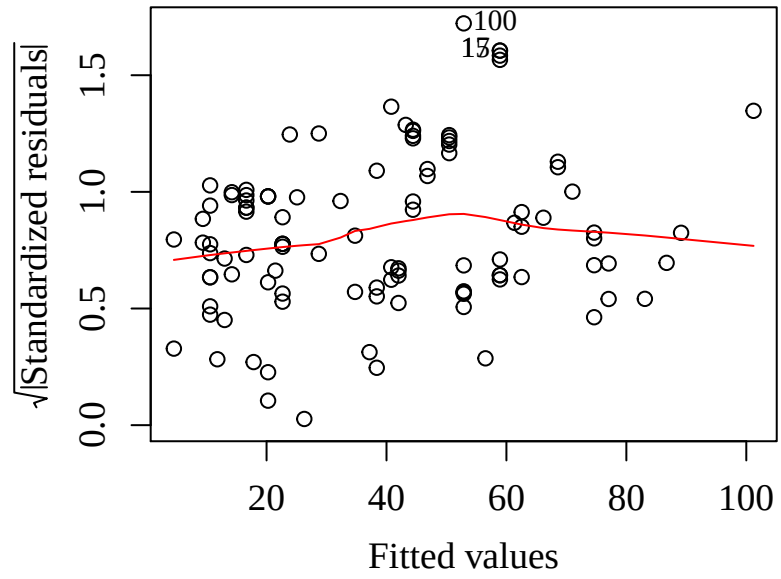
(Approximate because in order to have a linear model with additive errors, the original model must have mean 1 multiplicative errors.) This allows us to estimate a_0' and a_1' and compute weights $w_i = 1/(e^{a_0'}\texttt{Field}_i^{a_1'})$, where primes indicate coefficients from the linearized model.

```
linearizedmod <- lm(log(varlab) ~ log(meanfield))
a <- linearizedmod$coefficients
wlmod <- lm(Lab ~ Field, df, weights = 1/(exp(a[1])*(df$Field**a[2])))
sumary(wlmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.4944     0.9071   -1.65      0.1
## Field         1.2083     0.0349   34.64    <2e-16
##
## n = 107, p = 2, Residual SE = 1.169, R-Squared = 0.92
```

The model summary does not change substantially. The scale-location plot does improve substantially, suggesting the modeled weights are an improvement.

```
plot(wlmod, which = 3)
```



An alternate weighting scheme using `gls()` from the `nlme` package, as done in the book, yields similar results.

```
wlmod2 <- gls(Lab ~ Field, df, weights = varConstPower(1, form = ~ Field))
summary(wlmod2)$tTable
```

```
##                Value Std.Error t-value    p-value
## (Intercept) -0.8054  0.597833 -1.3472 1.8082e-01
## Field        1.1764  0.034018 34.5808 3.3421e-59
```

(c) Of the three transformations listed acting on the response only, the square root transformation best improves the scale-location plot. This is reasonable since, from the scale-location plot in (a), the variance increases until a threshold of y, then stabilizes. Log and inverse transformations destabilize the variance past the threshold.

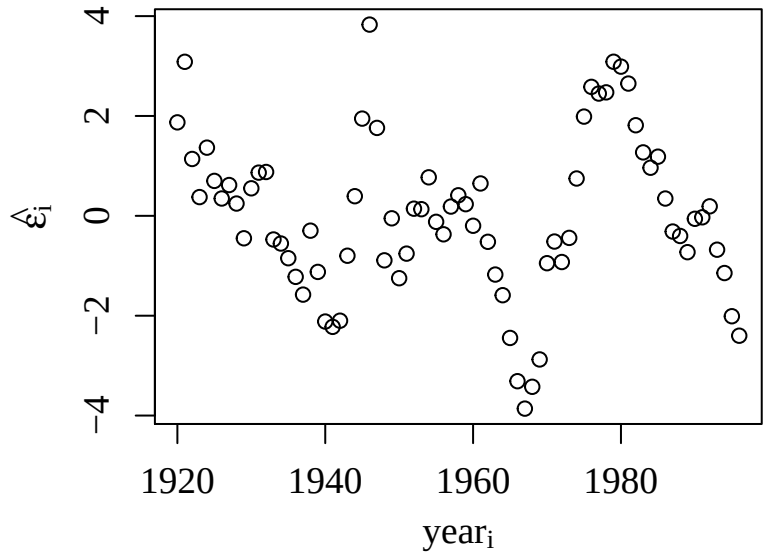
Exercise 2: Correlated errors.

```
df <- divusa
lmod <- lm(divorce ~ unemployed + femlab + marriage + birth + military, df)
```

(a) Correlation among errors can be detected by the residuals vs. fitted plot in the usual `plot(lmod)` diagnostics. Since we are dealing with temporal data, we can get a better picture by plotting a

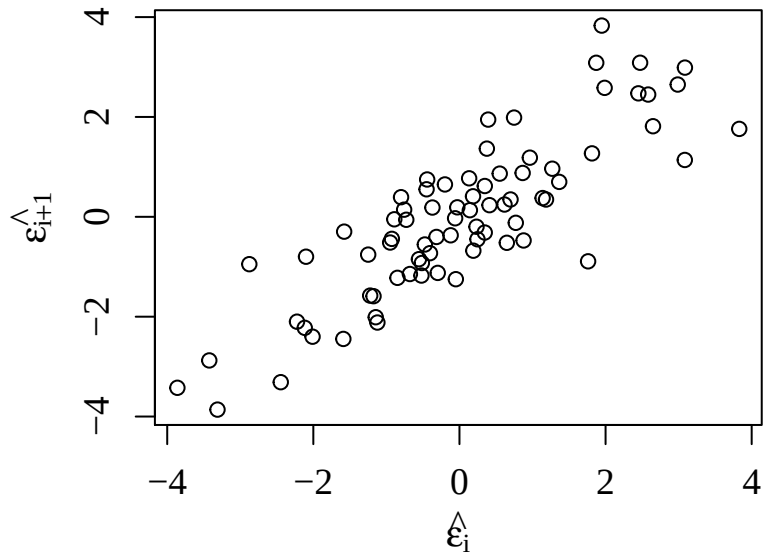
time series of residuals.

```
resid <- lmod$residuals
plot(
  df$year, resid,
  xlab = expression(year[i]), ylab = expression(hat(epsilon[i]))
)
```



The oscillatory patterns are clear indicators of correlated errors; they tend to increase and decrease together. This can be checked more carefully by plotting the $(i+1)$th residual against the ith residual.

```
n <- nrow(df)
plot(
  resid[1:n-1], resid[2:n],
  xlab = expression(hat(epsilon[i])), ylab = expression(hat(epsilon[i+1]))
)
```



The increasing trend suggests serial correlation, on average.

(b) We account for serial correlation using generalized least squares, particularly `gls()` with a

`corAR1()` correlation structure. We fit using maximum likelihood (rather than restricted) so the estimated variances are stable.

```
glmod <- gls(
  divorce ~ unemployed + femlab + marriage + birth + military, df,
  correlation = corAR1(form = ~ year), method = 'ML'
)
intervals(glmod, which = 'var-cov')
```

```
## Approximate 95% confidence intervals
##
##  Correlation structure:
##       lower    est.   upper
## Phi 0.65281 0.97155 0.99802
## attr(,"label")
## [1] "Correlation structure:"
##
##  Residual standard error:
##    lower     est.    upper
##  0.79744  2.90766 10.60206
```

We see strong, significant evidence for serial correlation; we would certainly reject the null of no serial correlation $H_0 : \phi = 0$. Coefficient estimates for `unemployed`, `marriage`, `birth`, and `military` change substantially. The statistical significance of `femlab`, `marriage`, and `birth` also change substantially.

```
summary(glmod)$tTable
```

```
##                 Value Std.Error t-value    p-value
## (Intercept) -7.059682  5.547193 -1.2727 2.0729e-01
## unemployed   0.107643  0.045915  2.3444 2.1860e-02
## femlab       0.312085  0.095151  3.2799 1.6110e-03
## marriage     0.164326  0.022897  7.1768 5.5611e-10
## birth       -0.049909  0.022012 -2.2673 2.6415e-02
## military     0.017946  0.014271  1.2575 2.1268e-01
```

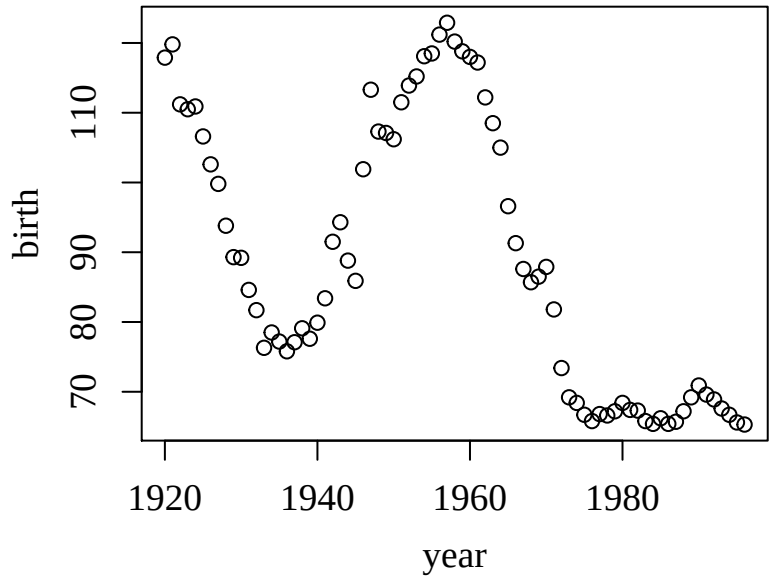
```
sumary(lmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.4878     3.3938    0.73    0.466
## unemployed   -0.1113     0.0559   -1.99    0.051
## femlab        0.3836     0.0306   12.54  < 2e-16
## marriage      0.1187     0.0244    4.86  6.8e-06
## birth        -0.1300     0.0156   -8.33  4.0e-12
## military     -0.0267     0.0142   -1.88    0.065
##
## n = 77, p = 6, Residual SE = 1.650, R-Squared = 0.92
```

Accounting for serial correlation changes the model substantially.

(c) Many variables in `divusa`, e.g. `birth`, change slowly over time. Therefore the previous year's birth rate is a strong indicator of the current birthrate, suggesting serial correlation. A plot of `birth` as a function of `year` demonstrates this.

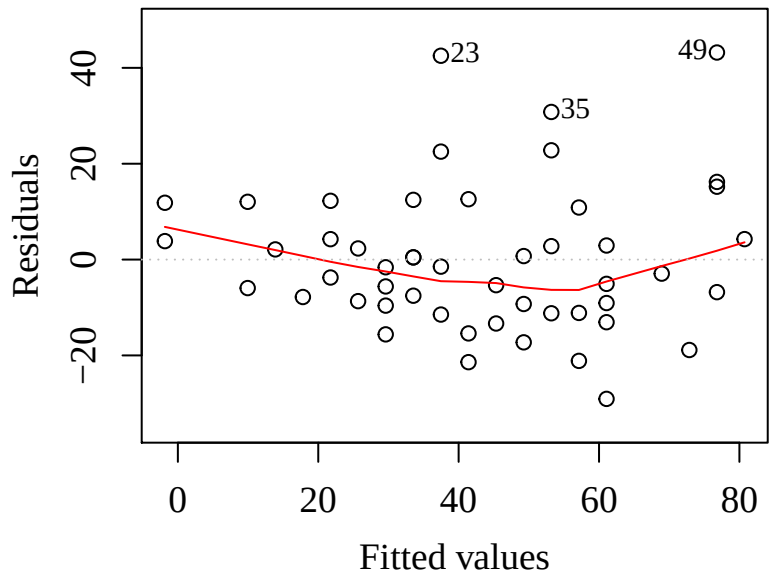
```
plot(birth ~ year, df)
```



Birth rates did not instantly plummet during the Great Depression, nor did they return to previous levels instantly after. Any model assuming i.i.d. errors without accounting for strongly nonlinear effects (and therefore risking overfitting) would observe serial correlations in the residuals.

Exercise 4: Detecting lack of fit. First, we employ a graphical method to detect lack of fit, namely a residuals vs. fitted plot.

```
lmod <- lm(dist ~ speed, cars)
plot(lmod, which = 1)
```



The trend line deviates slightly from flat. We investigate further by comparing the linear model to a saturated one with lowest possible RSE.

```
satmod <- lm(dist ~ factor(speed), cars)
anova(lmod, satmod)
```

```
## Analysis of Variance Table
##
```

```
## Model 1: dist ~ speed
## Model 2: dist ~ factor(speed)
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
## 1     48 11354
## 2     31  6765 17      4589 1.24   0.29
```

The large p-value suggests there is not lack of fit, i.e. the saturated model does not reduce the RSE in a statistically significant way.

Exercise 6: Differing ways to account for temporal effects.

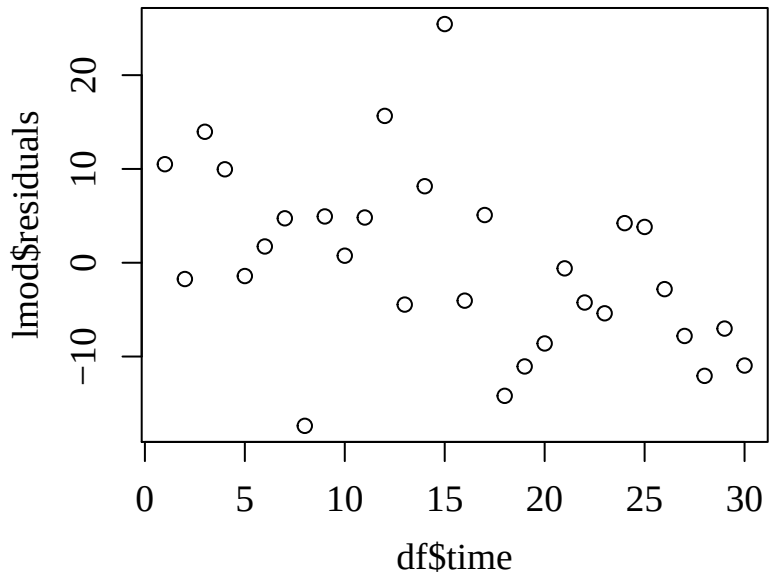
```
df <- cheddar
lmod <- lm(taste ~ ., df)
```

(a) Supposing the observations are equally spaced in time, one possible time variable is

```
df$time <- 1:nrow(df)
```

A plot of model residuals against `time` suggests little serial correlation.

```
plot(df$time, lmod$residuals)
```



(b) Fitting a `gls()` with `corAR1()` correlation structure suggests no evidence of serial correlation, as 0 is contained in the ϕ confidence interval.

```
glmod <- gls(taste ~ . - time, df, correlation = corAR1(form = ~ time))
intervals(glmod, which = 'var-cov')
```

```
## Approximate 95% confidence intervals
##
##  Correlation structure:
##        lower    est.   upper
## Phi -0.16903 0.26419 0.61186
## attr(,"label")
## [1] "Correlation structure:"
##
##  Residual standard error:
##   lower    est.   upper
```

```
##  7.6265 10.3328 13.9994
```

(c) Accounting for time as a predictor instead, we observe a significant effect at the $\alpha = 0.05$ level.

```
lmod_time <- lm(taste ~ ., df)
anova(lmod, lmod_time)
```

```
## Analysis of Variance Table
##
## Model 1: taste ~ Acetic + H2S + Lactic
## Model 2: taste ~ Acetic + H2S + Lactic + time
##   Res.Df  RSS Df Sum of Sq    F Pr(>F)
## 1     26 2668
## 2     25 2075  1       593 7.14  0.013 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

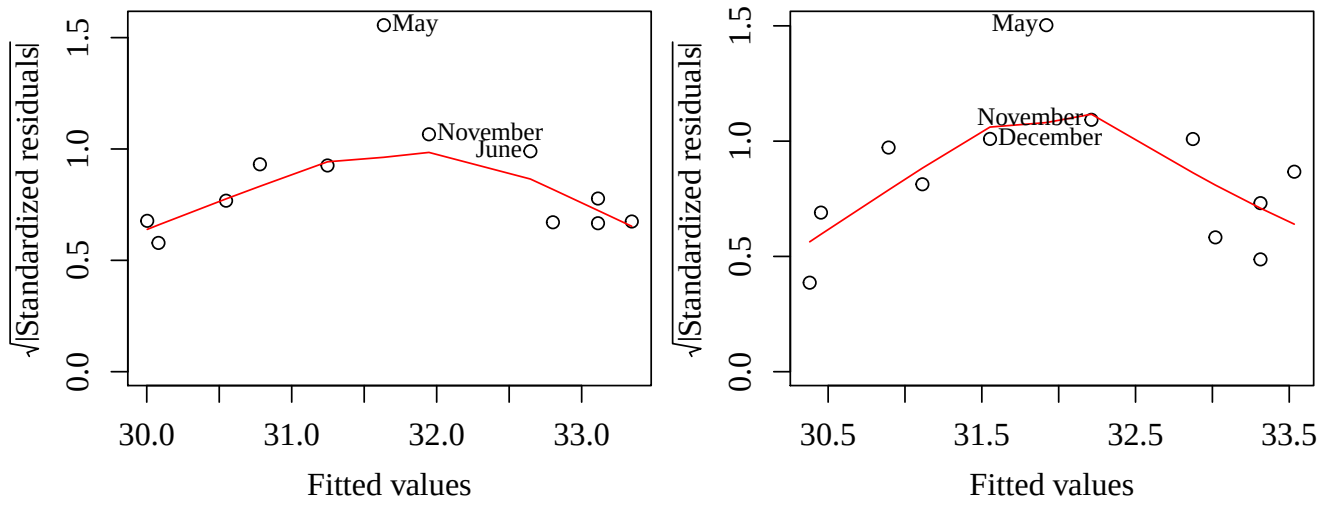
(d) The model in (b) corrects assumptions on the random error terms $\boldsymbol{\varepsilon}$, while the model in (c) corrects for assumptions on the fixed structural term, $\mathbb{E}\,\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}$. The model in (b) does not use time to directly predict the mean response, but accounts for correlations expected in residuals of data close in time. The model in (c) directly uses time to predict the mean response, but assumes the residuals are i.i.d.

(e) If the observations were not actually taken in time order, then the statistically significant result of (c) occurred by chance. If we try enough predictors, some will be significant by definition of p-values. The observations happened to be ordered so that, on average, `taste` decreased. This chance ordering was captured by our `time` variable.

Exercise 7: Choosing weights. Weights are chosen to adjust for potential unequal variance in the recorded responses. Even if all babies across all temperatures have the same variance σ^2 in crawling age, the observed responses `crawling` are averaged over n_{temp} observations, hence have variance σ^2/n_{temp}. Further, given that babies across temperatures may have differing variances σ^2_{temp}, variance within each temperature group is $\sigma^2_{\text{temp}}/n_{\text{temp}}$, so we should use weights $w_{\text{temp}} = n_{\text{temp}}/\sigma^2_{\text{temp}}$.

The weights have little effect here, evidenced by scale-location plots

```
df <- crawl
lmod <- lm(crawling ~ temperature, df)
plot(lmod, which = 3)
wlmod <- lm(crawling ~ temperature, df, weights = n/SD**2)
plot(wlmod, which = 3)
```



(though it is somewhat difficult to tell with so few data) and model summaries.

```
sumary(lmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  35.6781     1.3175    27.1  1.1e-10
## temperature  -0.0777     0.0251    -3.1    0.011
##
## n = 12, p = 2, Residual SE = 1.319, R-Squared = 0.49
```

```
sumary(wlmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  35.7326     1.2115   29.49  4.7e-11
## temperature  -0.0733     0.0233   -3.15     0.01
##
## n = 12, p = 2, Residual SE = 0.977, R-Squared = 0.5
```

Exercise 9: Robust regression.

(a) Fit using least squares and Huber's robust regression.

```
form <- brozek ~ (
  age + weight + height + neck + chest + abdom + hip + thigh + knee
  + ankle + biceps + forearm + wrist
)
lmod <- lm(form, fat)
rlmod <- rlm(form, fat)
```

Inspecting model summaries does not show substantial differences between models.

(b) The smallest weights in the robust regression are:

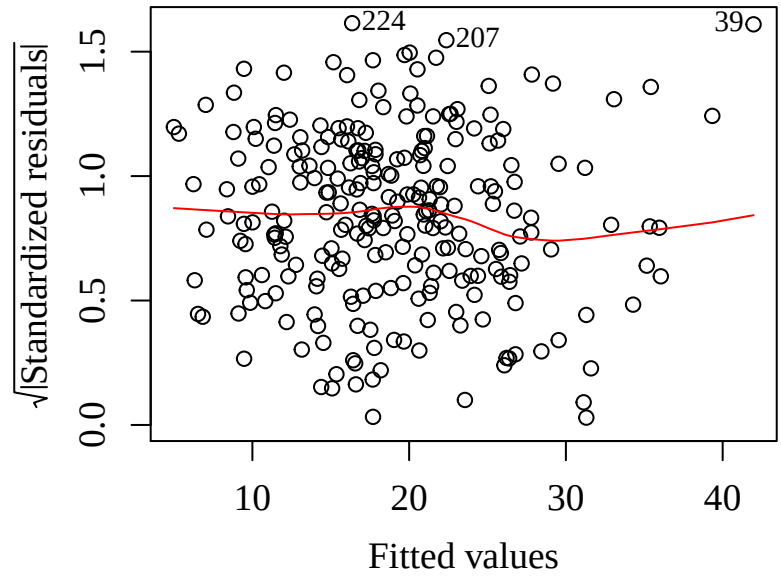
```
weights <- rlmod$w
names(weights) <- row.names(fat)
sort(weights)[1:5]
```

```
##     224     207      39     231     225
## 0.52697 0.58007 0.59878 0.62606 0.63040
```

These are the points that do not fit well, i.e. have larger residuals. Inspecting a scale-location plot

for the least squares models show the three largest standardized residuals correspond to the three smallest weights.

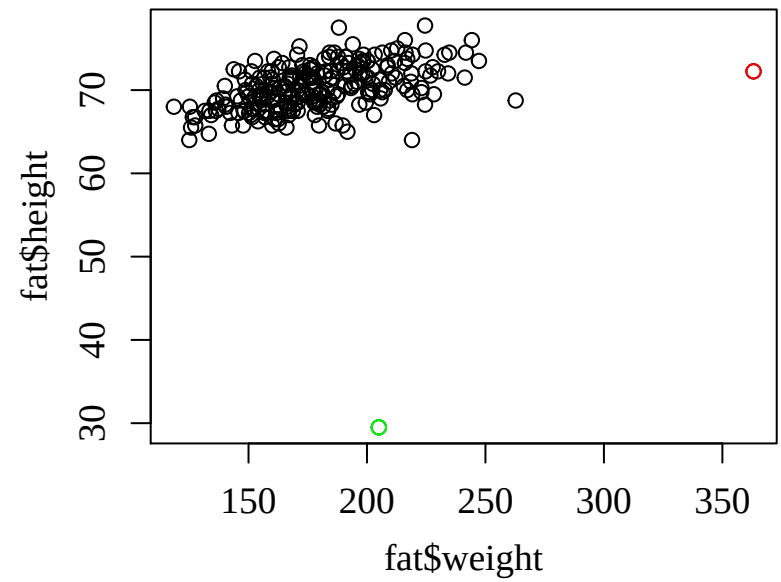
```
plot(lmod, which = 3)
```



(A scale-location plot using `rlmod` looks very similar.)

(c) A plot of height against weight show two unusual points, which do not correspond to the two smallest weights in the robust regression.

```
plot(fat$weight, fat$height)
points(fat$weight[39], fat$height[39], col = 'red')
points(fat$weight[42], fat$height[42], col = 'green')
```



We identified the large weight and small height as observations 39 and 42, respectively, in Exercise 4.5. These points have high leverage:

```
sort(hatvalues(lmod), decreasing = TRUE)[1:5] / mean(hatvalues(lmod))
```

```
##      42      39      86      31     175
## 13.3205  6.7522  6.4090  5.5622  4.8842
```

which, as mentioned in the text, are not addressed by M-estimation (and thus Huber's robust regression). Least-trimmed squares may be preferred.

Chapter 9

Transformation

9.1 Transforming the Response

Box–Cox details. Suppose the residuals of a linear model do not satisfy the Gauss–Markov assumptions of identically distributed with equal variance. Because OLS estimates are no longer BLUE, it can be useful to look for a transformation $y_i' = g(y_i)$ so that residuals of the transformed model are i.i.d. As there are an infinite number of possible transformations, it is difficult to find the best one. The Box–Cox method restricts the search space to the logarithm and all power transformations and finds the best-fitting transformation that results in normally distributed y_i' using likelihood methods, assuming $y_i \geq 0$ for all i. Specifically, it defines the family of transformations indexed by λ as

$$g_\lambda(y) = \begin{cases} (y^\lambda - 1)/\lambda & \lambda \neq 0 \\ \log y & \lambda = 0 \end{cases},$$

and chooses λ, i.e. the specific transformation, that maximizes the likelihood the observed responses y_i were observed assuming the transformed responses $y_i' = g_\lambda(y_i)$ are normally distributed. If the transformed responses are (not) normally distributed, the likelihood will be large (small) after adjusting for changes in scale, done by comparing likelihoods of the observed, not transformed, responses.

Derivation of the partial likelihood. Box–Cox assumes normality of transformed responses $y_i^{(\lambda)} = g_\lambda(y_i)$, $i = 1, 2, \ldots, n$, with means $\boldsymbol{\mu} = \boldsymbol{X}\boldsymbol{\beta}$ and constant variance σ^2. The likelihood of the transformed responses is

$$\begin{aligned} L^{(\lambda)}(\lambda, \boldsymbol{\beta}, \sigma^2) &= \prod_{i=1}^{n} \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{ -\frac{(y_i^{(\lambda)} - \mu_i)^2}{2\sigma^2} \right\} \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left\{ -\frac{(\boldsymbol{y}^{(\lambda)} - \boldsymbol{X}\boldsymbol{\beta})^T (\boldsymbol{y}^{(\lambda)} - \boldsymbol{X}\boldsymbol{\beta})}{2\sigma^2} \right\}. \end{aligned}$$

The likelihood can be viewed as a probability density function that is a function of the $y_i^{(\lambda)}$. The likelihood of the original responses $L(\lambda, \boldsymbol{\beta}, \sigma^2)$ is a density that is a function of the y_i obtained under the transformation from $y_i^{(\lambda)}$ to y_i. Using standard methods of transformations in probability theory (e.g., see Casella & Berger 2002, Section 4.6), the densities are related by a Jacobian determinant:

$$L(\lambda, \boldsymbol{\beta}, \sigma^2) = L^{(\lambda)}(\lambda, \boldsymbol{\beta}, \sigma^2)|J|,$$

where in this case $J_{ij} = \partial y_i^{(\lambda)}/\partial y_j = I(i=j)y_i^{\lambda-1}$, $I(\cdot)$ being the indicator function. It follows that $|J| = \prod_{i=1}^n y_i^{\lambda-1}$ when $y_i \geq 0$ for all i and the likelihood Box–Cox maximizes is

$$L(\lambda, \boldsymbol{\beta}, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left\{-\frac{(\boldsymbol{y}^{(\lambda)} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y}^{(\lambda)} - \boldsymbol{X}\boldsymbol{\beta})}{2\sigma^2}\right\} \prod_{i=1}^n y_i^{\lambda-1}.$$

As the logarithm is a monotonic transformation, it is sufficient and often simpler to maximize the log-likelihood

$$l(\lambda, \boldsymbol{\beta}, \sigma^2) \equiv \log L(\lambda, \boldsymbol{\beta}, \sigma^2) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\boldsymbol{y}^{(\lambda)} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y}^{(\lambda)} - \boldsymbol{X}\boldsymbol{\beta}) + (\lambda-1)\sum_{i=1}^n \log y_i.$$

As we want to maximize w.r.t. λ, we can make the computational task easier by analytically maximizing over the two nuisance parameters $\boldsymbol{\beta}$ and σ^2. Noting that the first two terms are just the log-likelihood of a normal random sample of transformed responses and the third term is independent of the nuisance parameters, the MLEs of $\boldsymbol{\beta}$ and σ^2 are just the MLEs in the simplified normal random sample problem:

$$\widehat{\boldsymbol{\beta}}_{\text{MLE}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}^{(\lambda)}, \qquad \widehat{\sigma^2}_{\text{MLE}} = (\boldsymbol{y}^{(\lambda)} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}_{\text{MLE}})^T(\boldsymbol{y}^{(\lambda)} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}_{\text{MLE}})/n.$$

Upon substitution and defining $\text{RSS}^{(\lambda)} \equiv (\boldsymbol{y}^{(\lambda)} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}_{\text{MLE}})^T(\boldsymbol{y}^{(\lambda)} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}_{\text{MLE}})$, we find the profile log-likelihood for λ is

$$l(\lambda) = c - \frac{n}{2}\log \text{RSS}^{(\lambda)} + (\lambda-1)\sum_{i=1}^n \log y_i, \qquad \text{where } c = -\frac{n}{2}\left[\log\left(\frac{2\pi}{n}\right) + 1\right]$$

is just a constant. Dropping the constant recovers the book's expression (noting the first term in the book can be simplified further by removing the constant piece in the log).

Derivation of the λ confidence interval. We can construct confidence intervals using the likelihood ratio hypothesis test and the duality between intervals and tests. First, recall the likelihood ratio statistic $t = L(\lambda_0)/L(\widehat{\lambda})$. As λ_0 is a subset of all possible λ, $0 \leq t \leq 1$. We reject H_0 if t is sufficiently small. We often use $\text{LR} = -2\log t = 2[L(\widehat{\lambda}) - L(\lambda_0)]$ instead and reject for large values because of its asymptotic properties by Wilks' theorem. For large n, $\text{LR} \approx \chi^2(\text{d.f.} = 1)$, hence we reject H_0 at the α significance level if $\text{LR} \geq \chi^2_{1-\alpha}(1)$. After some algebra, this leads to rejection criterion

$$L(\lambda_0) \leq L(\widehat{\lambda}) - \frac{1}{2}\chi^2_{1-\alpha}(1).$$

Since the corresponding $(1-\alpha)$ confidence interval consists of all λ_0 such that H_0 is not rejected, this is the set

$$\left\{\lambda : \quad L(\lambda) > L(\widehat{\lambda}) - \frac{1}{2}\chi^2_{1-\alpha}(1)\right\}.$$

9.2 Transforming the Predictors

The Box–Cox style approach on the response is useful because normality of the response is desirable (e.g., for inference). In contrast, we don't usually care about the distribution of the predictors, so we can use more flexible transformations like splines.

9.3 Broken Stick Regression

To show the basis function representation

$$y = \beta_0 + \beta_1 B_l(x) + \beta_2 B_r(x) + \epsilon$$

is equivalent to fitting two lines that meet at the knot c, consider the two cases separately. For $x < c$, $B_r(x) = 0$, so the above simplifies to the familiar simple linear regression

$$y_{x<c} = \beta_0 + \beta_1 B_l(x) + \epsilon = (\beta_0 + c\beta_1) - \beta_1 x + \epsilon.$$

Similarly for $x > c$,

$$y_{x>c} = \beta_0 + \beta_2 B_r(x) + \epsilon = (\beta_0 - c\beta_2) + \beta_2 x + \epsilon.$$

It is straightforward to check the lines meet at c: $y_{x<c}(c) = y_{x>c}(c)$. This additional constraint reduces the number of estimated parameters from four to three.

9.4 Polynomials

9.5 Splines

9.6 Additive Models

9.7 More Complex Models

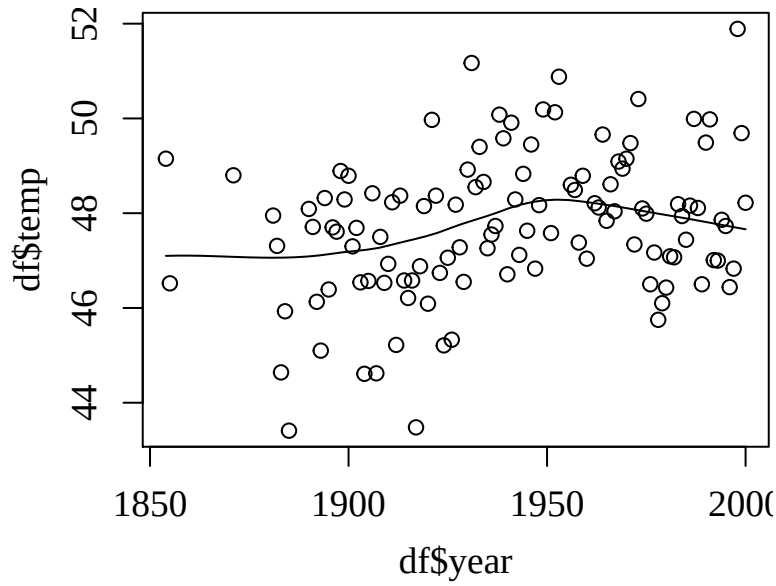
Exercises

```
library(faraway)  # For data and sumary().
library(splines)  # For splines, particularly splineDesign().
library(MASS)  # For boxcox().
```

Exercise 1: Comparing predictor transformation methods for annual temperature data.

(a) The temperature trend does not appear linear over time, evidenced by a smoothed loess trend line.

```
df <- aatemp
scatter.smooth(df$year, df$temp)
```



(b) Modeling serial correlations yields a statistically significant correlation.

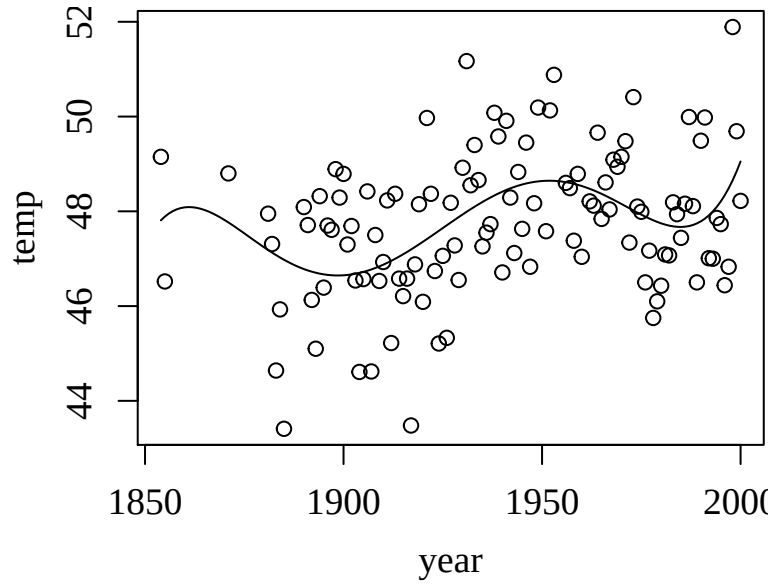
```
n <- nrow(df)
lmod <- lm(df$temp[2:n] ~ df$temp[1:n-1])
sumary(lmod)
```

```
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        35.5760     4.3489    8.18  4.9e-13
## df$temp[1:n - 1]    0.2546     0.0911    2.80   0.0061
##
## n = 114, p = 2, Residual SE = 1.483, R-Squared = 0.07
```

This does not change the conclusion reached in (a) because observations may be correlated while maintaining a nonlinear overall trend. For example, daily temperatures are correlated but there is a seasonal fluctuation.

(c) Using orthogonal polynomials, a degree 10 fit suggests a degree 5 fit is sufficient. This model and its fit is plotted next.

```
poly_mod <- lm(temp ~ poly(year, degree = 5), df)
x1 <- min(df$year)
x2 <- max(df$year)
x <- seq(x1, x2, 1)
y <- predict(poly_mod, newdata = data.frame(year = x))
plot(temp ~ year, df)
lines(x, y)
```



One limitation of polynomial fits is their behavior at the tails, where only a few data points can substantially alter a fit. As such, extrapolation can be especially dangerous evidenced in a 2020 prediction.

```
predict(poly_mod, newdata = data.frame(year = 2020))
```

```
##      1
## 60.078
```

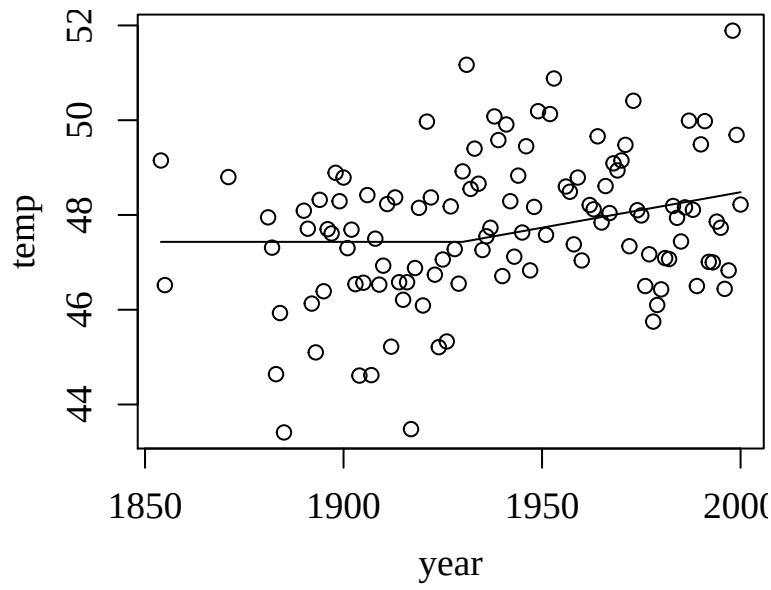
(d) The claimed model, enforcing continuity at 1930, can be modeled as $y = \beta_0 + \beta_1 f(x) + \varepsilon$,

where the basis function

$$f(x) = \begin{cases} x - 1930 & x \geq 1930 \\ 0 & x < 1930 \end{cases}.$$

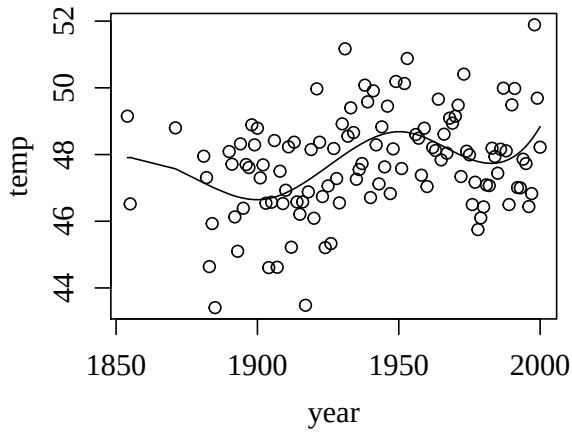
Using this to fit a linear model the usual way yields a poor fit and suggests the claim is false.

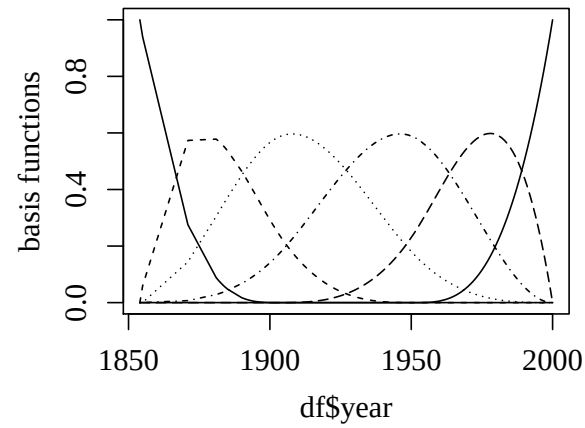
```
f <- function(x) ifelse(x >= 1930, x - 1930, 0)
piecewise_mod <- lm(temp ~ f(year), df)
y <- predict(piecewise_mod, newdata = data.frame(year = x))
plot(temp ~ year, df)
lines(x, y)
```



(e)

```
knots <- c(rep(x1, 3), seq(x1, x2, length.out = 4), rep(x2, 3))
X <- splineDesign(knots, df$year)
spline_mod <- lm(df$temp ~ X - 1)  # Intercept contained in X.
plot(temp ~ year, df)
lines(df$year, predict(spline_mod))  # B-spline fit.

matplot(df$year, X, type="l", col=1, ylab = 'basis functions')
```





The cubic B-spline fit is better behaved at the left tail but is otherwise similar to the polynomial fit. Considering the same number of parameters were estimated, it is an improvement, and greatly improves upon the piecewise model in (d). One difficulty with B-splines is their ability

to extrapolate because the basis functions are only defined on the support, i.e. between knots and hence data. Restricted cubic splines can circumvent this issue by constraining the tails to be linear.

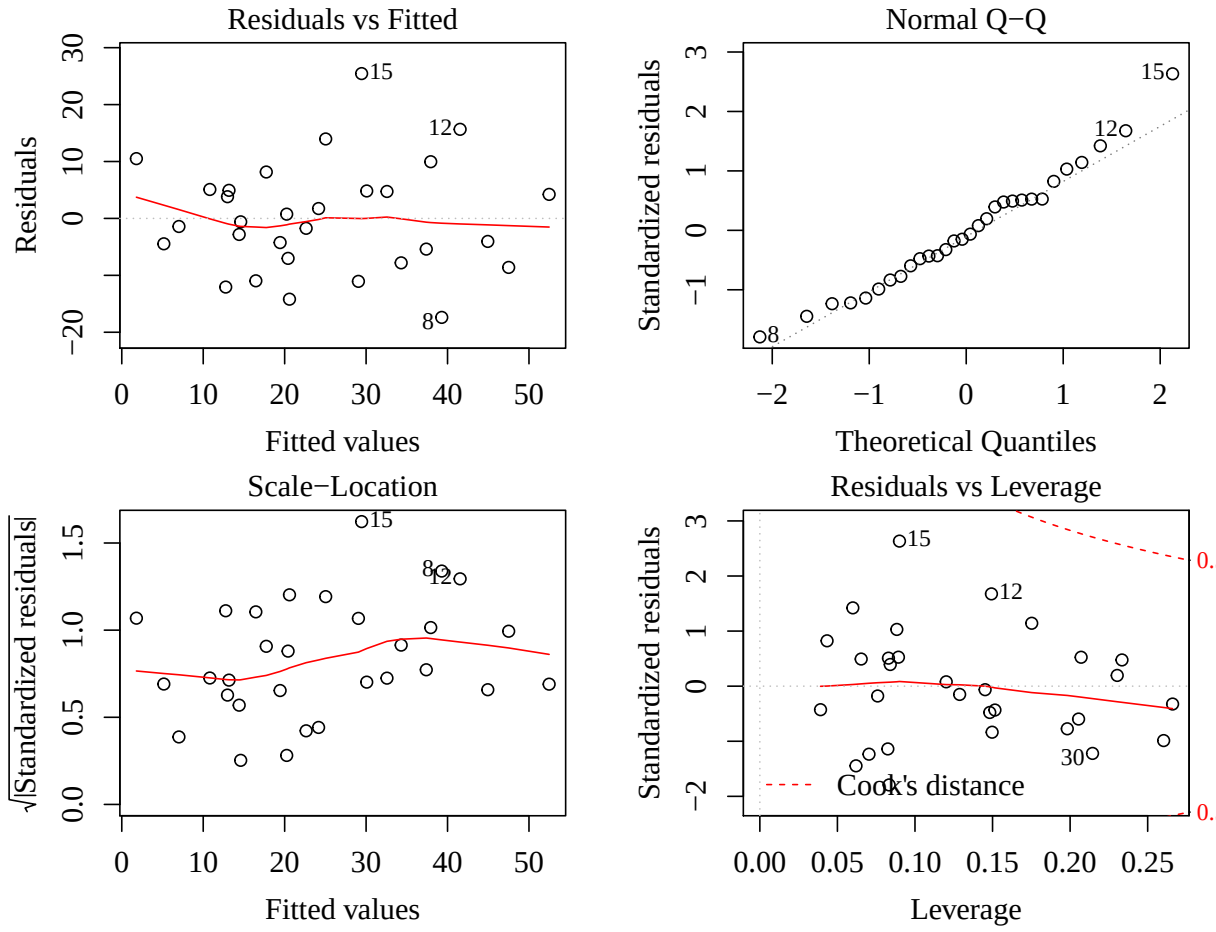
Exercise 7: Examining and re-examining transformations for predictors and response.

(a) We first try the simplest additive model with no transformations.

```
df <- cheddar
lmod <- lm(taste ~ Acetic + H2S + Lactic, df)
```

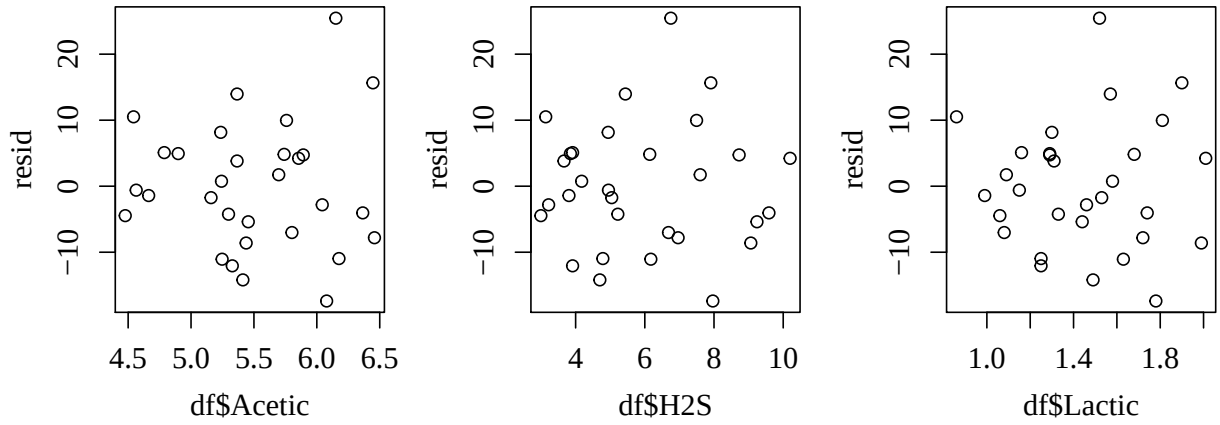
Inspecting the standard diagnostic plots shows no substantial issues, e.g. error assumptions, unusual observations, and model structure.

```
plot(lmod)
```



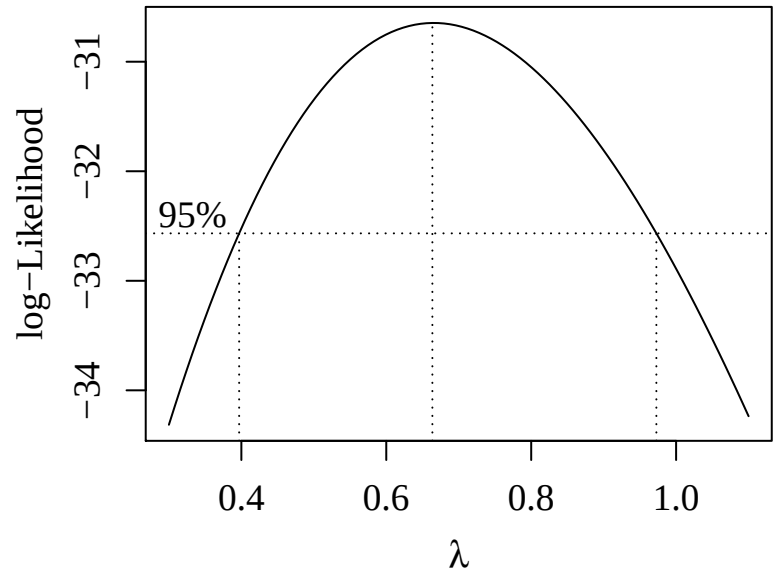
Further inspecting residual plots against individual predictors suggests no obvious transformations.

```
resid <- lmod$residuals
plot(df$Acetic, resid)
plot(df$H2S, resid)
plot(df$Lactic, resid)
```



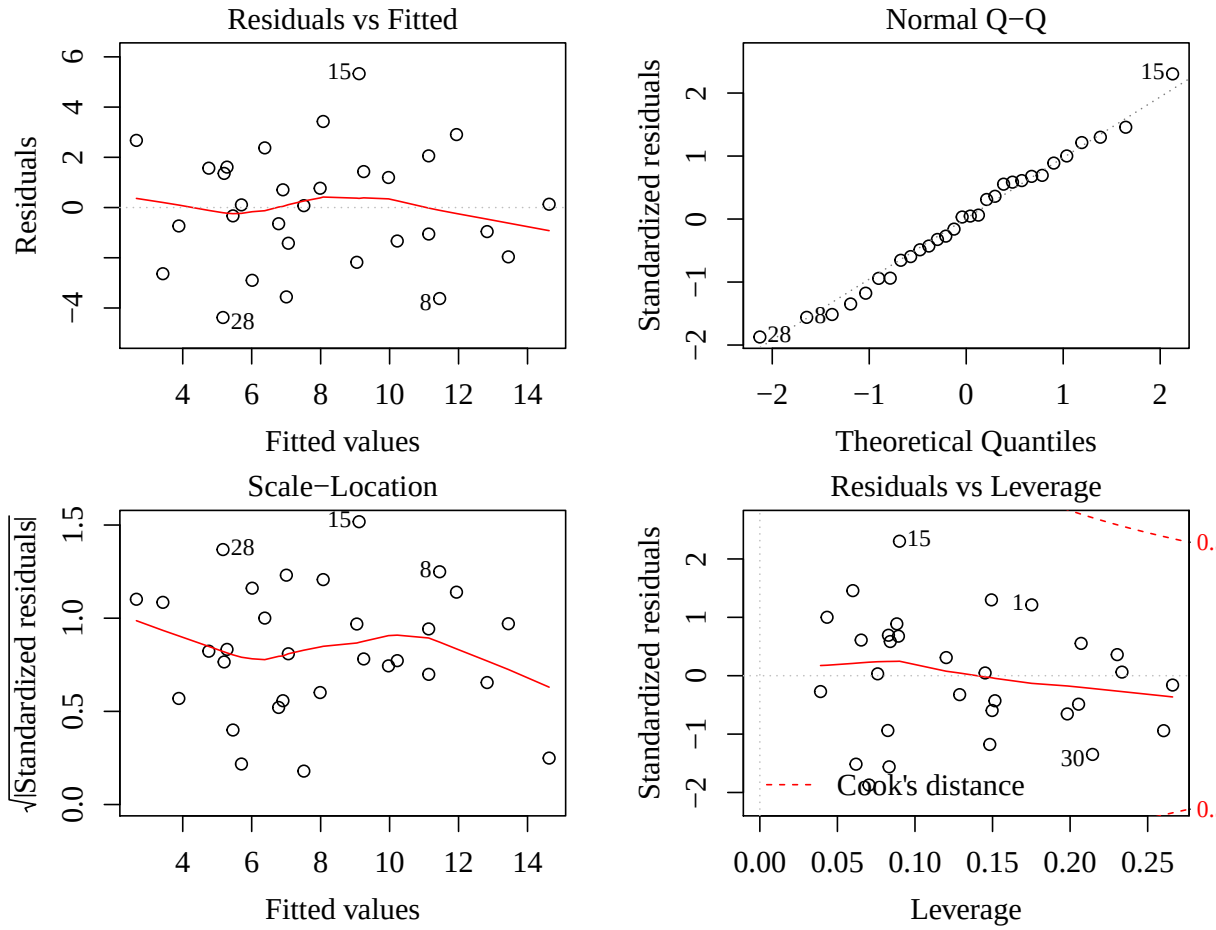
(b) While there are no obvious transformations of the response, we do notice a slight upward trend in the scale-location plot. We try the Box–Cox transformation, a likelihood based way of determining the best power or (unshifted) log transformation assuming the responses can be transformed to normality.

```
boxcox(lmod, data = df, lambda = seq(.3, 1.1, .01))
```



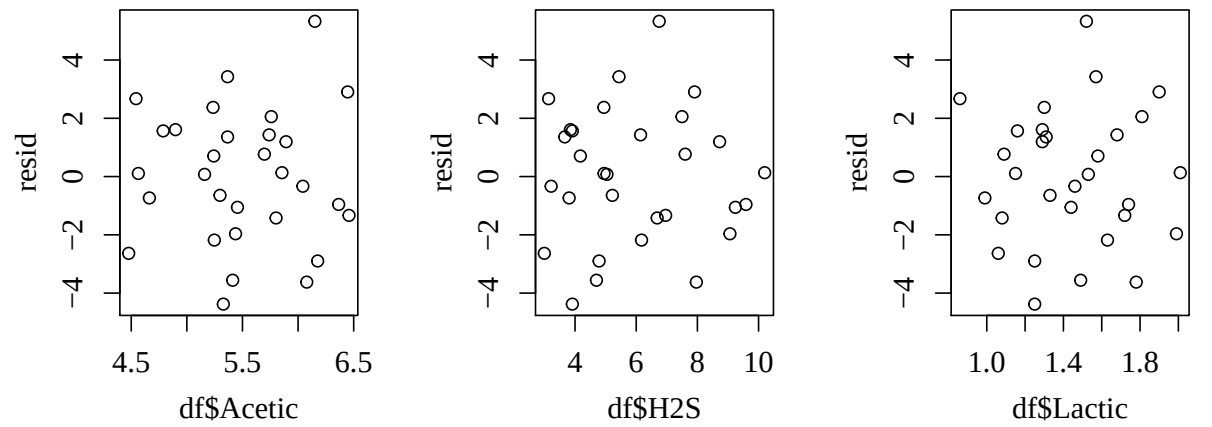
As no transformation, $\lambda = 1$, is outside the 95% confidence interval, the plot suggests we should try a transformation. For interpretability, we may default to the square root $\lambda = \frac{1}{2}$. For the purposes of this exercise, we will use the optimal value $\lambda \approx \frac{2}{3}$.

(c)
```
g <- function(y) y^(2/3)
lmod_bc <- lm(g(taste) ~ Acetic + H2S + Lactic, df)
plot(lmod_bc)
```



The qqplot improves slightly, but there is otherwise little change. There is still no obvious suggested transformation of the predictors, evidenced in partial plots.

```
resid <- lmod_bc$residuals
plot(df$Acetic, resid)
plot(df$H2S, resid)
plot(df$Lactic, resid)
```



Chapter 10

Model Selection

10.1 Hierarchical Models

10.2 Testing-Based Procedures

10.3 Criterion-Based Procedures

More on adjusted R^2. Like the AIC, R_a^2 has two competing factors: model fit measured by RSS and a penalty term based on the number of predictors. Unlike the AIC, the factors are a ratio rather than additive contributions. Specifically, from the book

$$R_a^2 = 1 - \frac{\widehat{\sigma}^2_{model}}{\widehat{\sigma}^2_{null}}$$

only has one model dependent term: $\widehat{\sigma}^2_{model} = \text{RSS}/(n-p)$, a ratio of the two factors.

More on Mallow's C_p. To reason about possible values of C_p, let $k \geq p$ be the number of parameters in the full model. Then Faraway's $\widehat{\sigma}^2 = \text{RSS}_k/(n-k)$, so we can express

$$C_p = \frac{\text{RSS}_p}{\text{RSS}_k}(n-k) + 2p - n\,.$$

Let the RSS ratio $r \equiv \text{RSS}_p/\text{RSS}_k$. Since the RSS never increases as more predictors are added, $r \geq 1$. Therefore the smallest C_p can be is $C_p^* = 2p - k$. For $p = k$, C_p will always intersect the line $y = p$. Decreasing p will usually lead to an optimal minimum [possibly above the line $y = p$, see Exercise 1(d)], but decreasing too far will yield a poor fit, hence large r and C_p.

10.4 Summary

Criterion based vs. hypothesis testing methods. For concreteness, we will use AIC as a representation of criterion based methods and the general F-test (or likelihood based test) discussed in Sections 3.1 and 3.2, implemented in R using `anova()`, as a representation of hypothesis testing methods.

Faraway generally recommends AIC over `anova()` for model selection. One major advantages is AIC's ability to compare non-nested models. One limitation is that it is hard to tell how much better two models are with different AICs. In contrast, for nested models `anova()` can be

used as a statistical test for model comparison. However for variable selection, e.g. backward elimination, the p-values associated with the test lose meaning as so many models are examined. In general, I would recommend AIC for variable selection. However, if only a few (ideally 2) models are compared, `anova()` may be better; an example is a test of linearity comparing two models, e.g. $H_0 : \beta_2 = \beta_3 = 0$. More discussion can be found in these stackexchange threads:

- Partial F-test vs Model Selection
- AIC or ANOVA to compare models?

Exercises

```
library(faraway)  # For data and sumary().
library(leaps)  # For regsubsets().
```

Exercise 1: Comparing multiple model selection methods.

```
df <- prostate
```

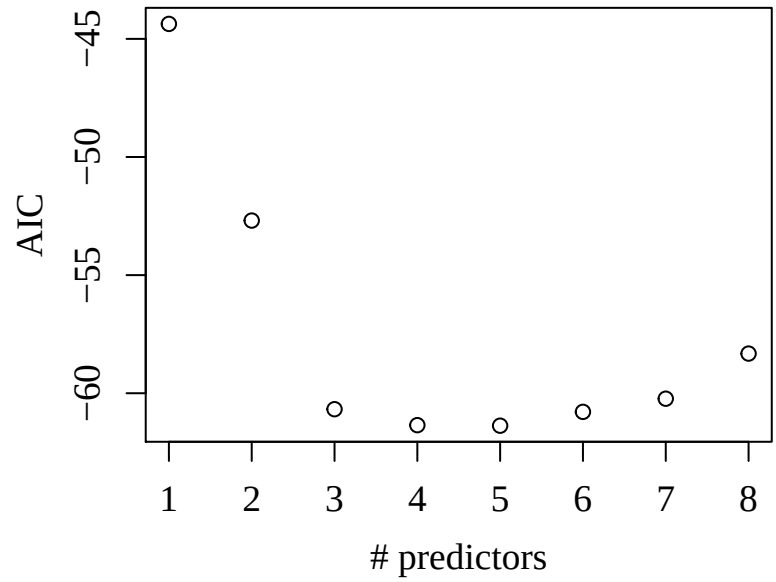
(a) Backward elimination with $\alpha = 0.1$ cutoff results in a 3-variable model of `lcavol`, `lweight`, and `svi`. Intermediate summaries omitted.

```
lmod <- lm(lpsa ~ ., df)
lmod <- update(lmod, . ~ . - gleason)
lmod <- update(lmod, . ~ . - lcp)
lmod <- update(lmod, . ~ . - pgg45)
lmod <- update(lmod, . ~ . - age)
lmod <- update(lmod, . ~ . - lbph)
sumary(lmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.2681     0.5435   -0.49    0.623
## lcavol        0.5516     0.0747    7.39  6.3e-11
## lweight       0.5085     0.1502    3.39    0.001
## svi           0.6662     0.2098    3.18    0.002
##
## n = 97, p = 4, Residual SE = 0.717, R-Squared = 0.63
```

(b) Minimizing AIC results in a 5-variable model, adding `age` and `lbph` to the backward elimination's model (a).

```
n <- nrow(df)
p_max <- ncol(df)
results <- summary(regsubsets(lpsa ~ ., data = df))
rss <- results$rss
aic <- n*log(rss/n) + 2*(2:p_max)
plot(1:(p_max-1), aic, xlab = '# predictors', ylab = 'AIC')
```




```
results$which[which(aic == min(aic)), ]
```

```
## (Intercept)      lcavol     lweight         age        lbph
##        TRUE        TRUE        TRUE        TRUE        TRUE
##         svi         lcp     gleason       pgg45
##        TRUE       FALSE       FALSE       FALSE
```

Because the 4-predictor model is so close in AIC, we may favor it for simplicity.

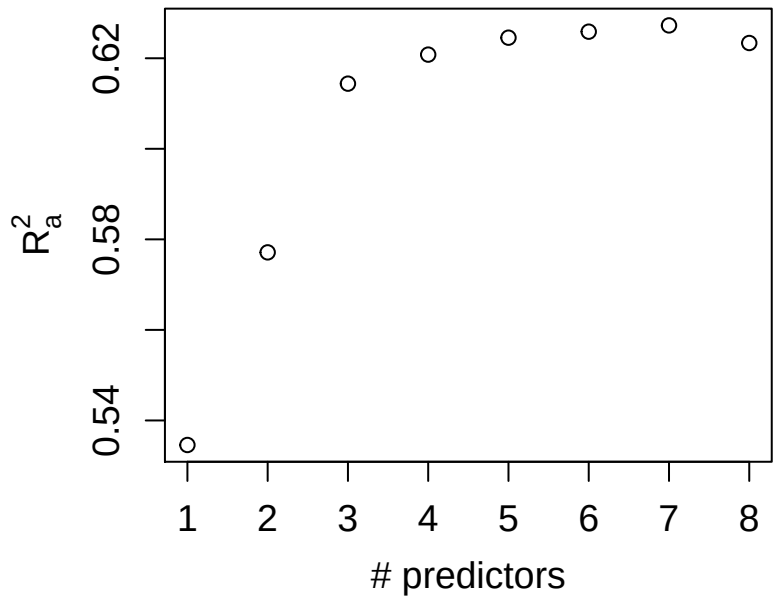
```
results$which[4, ]
```

```
## (Intercept)      lcavol     lweight         age        lbph
##        TRUE        TRUE        TRUE       FALSE        TRUE
##         svi         lcp     gleason       pgg45
##        TRUE       FALSE       FALSE       FALSE
```

The 3-variable model results in the same variables as backward elimination (a). (This will be true for all models using `regsubsets()` since the RSS based selection criteria, e.g. AIC, R_a^2, and C_p, only differ in how the number of predictors penalizes the model.)

(c) Maximizing R_a^2 results in a 7-variable model–all variables except `gleason` (the first to be backward eliminated).

```
adjr2 <- results$adjr2
plot(1:(p_max-1), adjr2, xlab = '# predictors', ylab = expression('R'[a]^2))
```

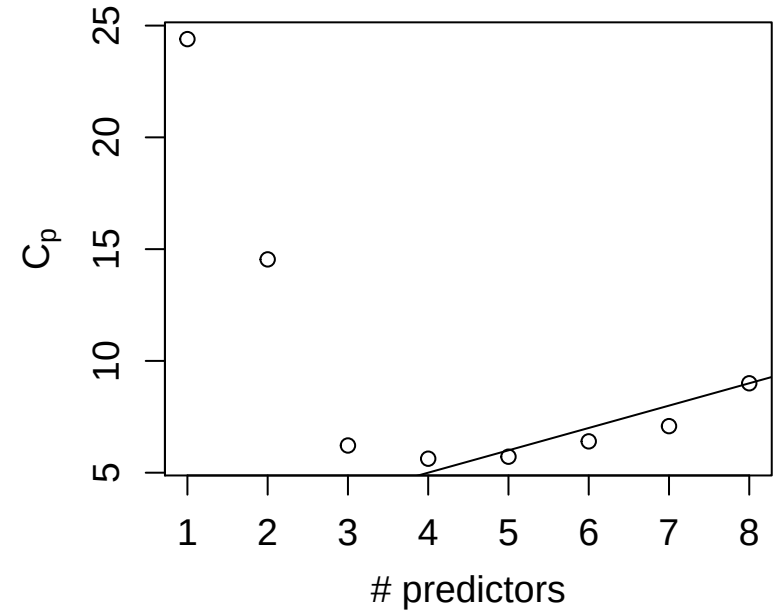



```
results$which[which(adjr2 == max(adjr2)), ]
```

```
## (Intercept)      lcavol     lweight         age        lbph         svi
##        TRUE        TRUE        TRUE        TRUE        TRUE        TRUE
##         lcp     gleason       pgg45
##        TRUE       FALSE        TRUE
```

(d) Minimizing C_p results in a 4-variable model. The 5-variable model lies below the line $y = p$, but is slightly larger.

```
cp <- results$cp
plot(1:(p_max-1), cp, xlab = '# predictors', ylab = expression('C'[p]))
abline(1, 1)
```



```
results$which[which(cp == min(cp)), ]
```

```
## (Intercept)      lcavol     lweight         age         lbph          svi
##        TRUE        TRUE        TRUE       FALSE        TRUE        TRUE
##          lcp     gleason       pgg45
##       FALSE       FALSE       FALSE
```

AIC and C_p result in similar models. R_a^2 results in a larger model, while backward elimination results in a smaller model.

Exercise 4: Attempting model simplification. First, fit the full second order model.

```
df <- trees
lmod <- lm(
  log(Volume) ~ Girth + Height + I(Girth**2) + I(Height**2) + Girth:Height, df
)
```

A test of nonlinearity is often a practical simplification, which we can do using the general F-test formalism via `anova()` with a reduced model.

```
lmod_red <- lm(log(Volume) ~ Girth + Height, df)
anova(lmod_red, lmod)
```

```
## Analysis of Variance Table
##
## Model 1: log(Volume) ~ Girth + Height
## Model 2: log(Volume) ~ Girth + Height + I(Girth^2) + I(Height^2) + Girth:Height
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
## 1     28 0.262
## 2     25 0.179  3    0.0828 3.85  0.022 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see the nonlinear terms are statistically significant at the $\alpha = 0.05$ level. This may be slightly surprising, considering there is only a minor improvement in R^2:

```
summary(lmod_red)$r.squared
```

```
## [1] 0.96845
```

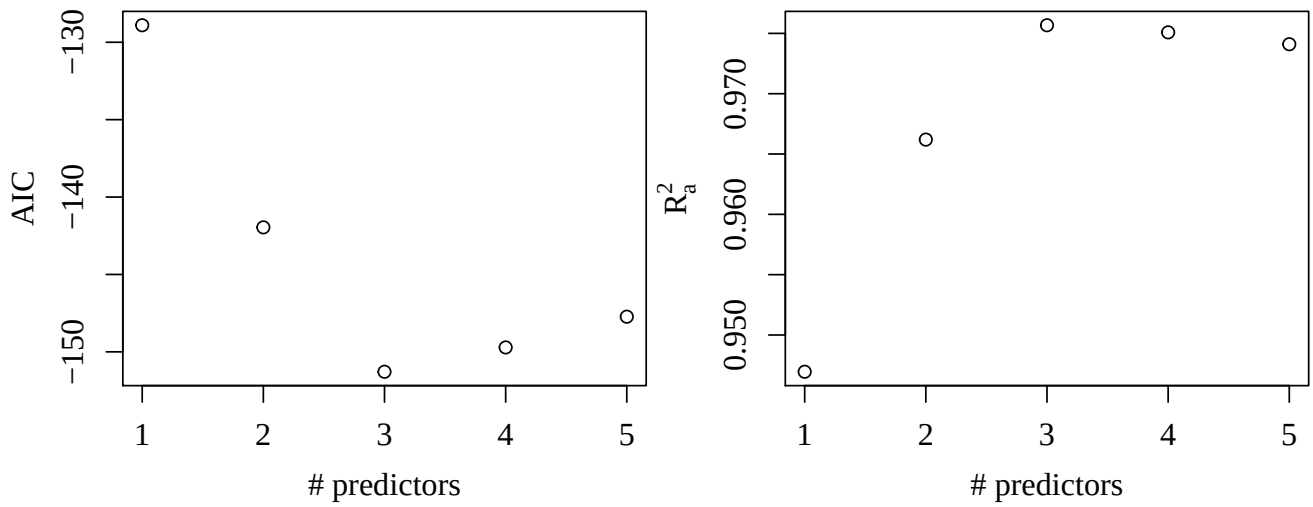
```
summary(lmod)$r.squared
```

```
## [1] 0.97842
```

Since the number of parameters is relatively small, we can supplement this test by running an exhaustive search with selection criteria. I will use AIC and R_a^2.

```
n <- nrow(df)
p_max <- ncol(df) + 3
results <- summary(
  regsubsets(
    log(Volume) ~ Girth + Height + I(Girth**2) + I(Height**2) + Girth:Height, df
  )
)
```

```
rss <- results$rss
aic <- n*log(rss/n) + 2*(2:p_max)
plot(1:(p_max-1), aic, xlab = '# predictors', ylab = 'AIC')

adjr2 <- results$adjr2
plot(1:(p_max-1), adjr2, xlab = '# predictors', ylab = expression('R'[a]^2))
```



Both criteria suggest the following 3-variable model. (4- and 5- variable models are still an improvement over a purely linear model.)

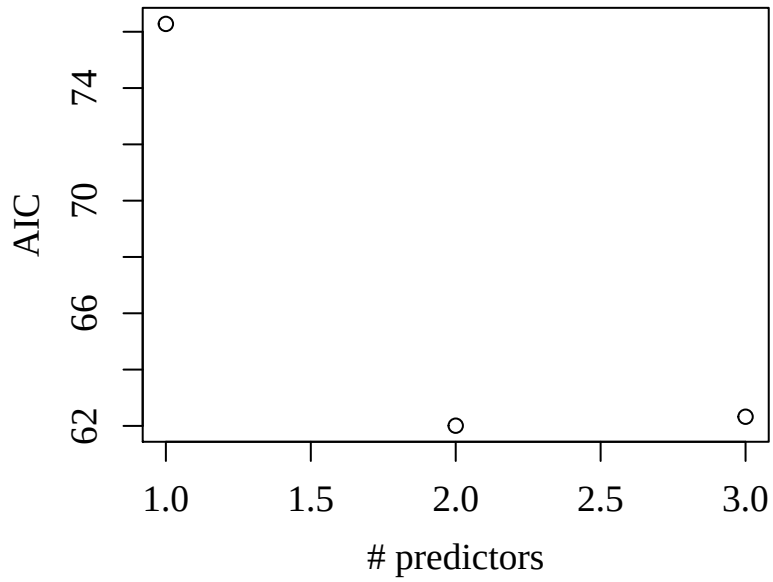
```
results$which[which(aic == min(aic)), ]
```

```
##  (Intercept)        Girth       Height   I(Girth^2)  I(Height^2)
##         TRUE         TRUE         TRUE         TRUE        FALSE
## Girth:Height
##        FALSE
```

It seems it is most important to make `Girth` nonlinear. For simplicity, it seems reasonable to use this reduced model which assumes additivity and linearity of `Height`. The evidence is more compelling if we take a step back from statistics and think about a tree physically, noting from `?trees` that `Girth` is a measurement of tree diameter. As height increases, the tree grows in one dimension, hence volume increases roughly linearly. As diameter increases, the tree grows in two dimensions, hence volume increases roughly quadratically. Ideally we would have made these observation to inform our first model, but they do increase our confidence in the 3-variable model chosen via statistical considerations only. In hindsight, our `anova()` test of nonlinearity was practical from a statistical view, but not a physical one.

Exercise 5: Model selection with potential outliers. I run an exhaustive search to minimize AIC.

```
df <- stackloss
p_max <- ncol(df)
results <- summary(regsubsets(stack.loss ~ ., df))

rss <- results$rss
aic <- n*log(rss/n) + 2*(2:p_max)
plot(1:(p_max-1), aic, xlab = '# predictors', ylab = 'AIC')
```

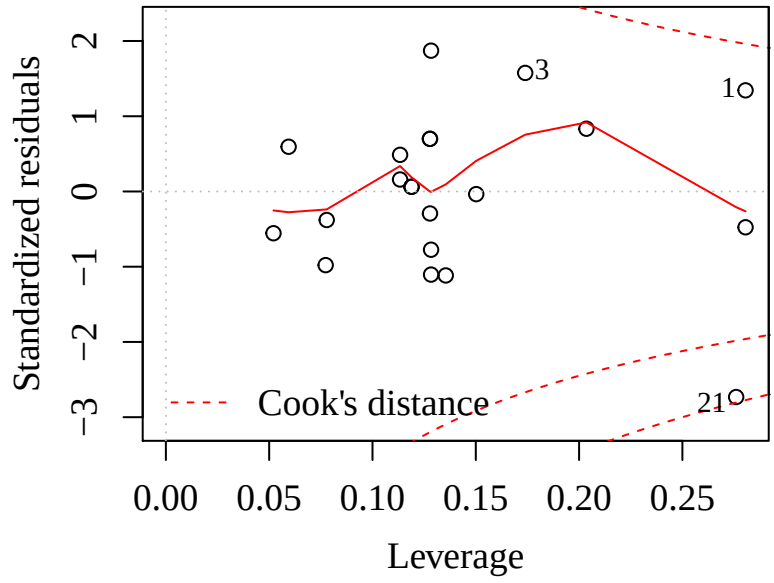



```
results$which[which(aic == min(aic)), ]
```

```
## (Intercept)    Air.Flow  Water.Temp  Acid.Conc.
##        TRUE        TRUE        TRUE       FALSE
```

This results in a 2-variable model of `Air.Flow` and `Water.Temp`. Fitting this reduced model and examining a residuals vs. leverage plot with Cook's distance contours,

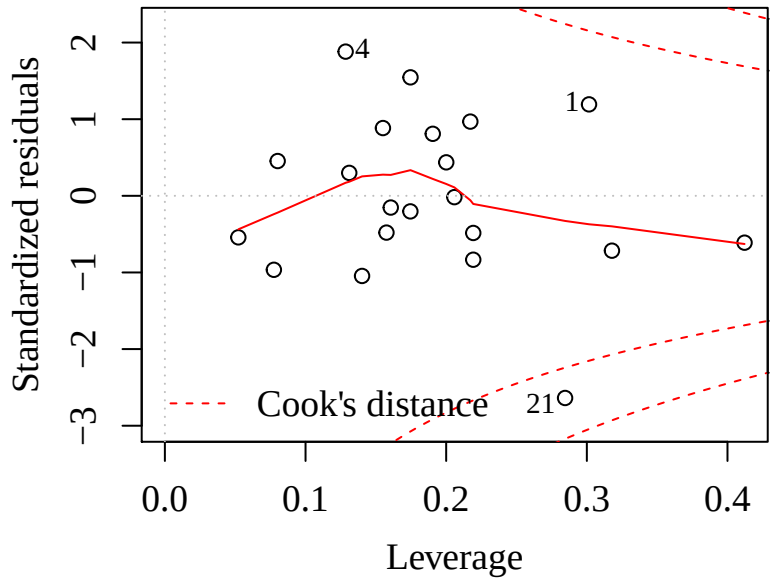
```
lmod_red <- lm(stack.loss ~ Air.Flow + Water.Temp, df)
plot(lmod_red, which = 5)
```



we observe that the 21st observation is very influential.

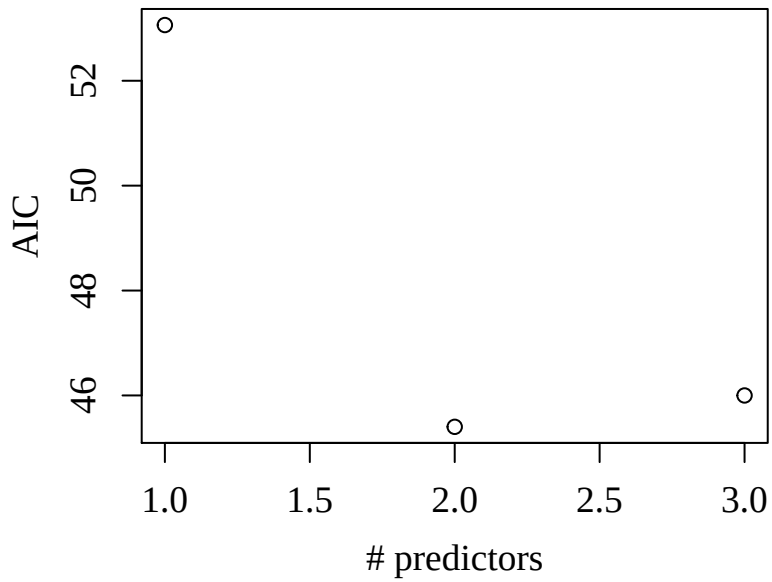
Suppose we examined the residuals vs. leverage plot before performing model selection.

```
lmod <- lm(stack.loss ~ ., df)
plot(lmod, which = 5)
```



We again observe the 21st observation is influential, but somewhat less. Removing it and repeating the model selection process:

```
results <- summary(regsubsets(stack.loss ~ ., df, subset = -c(21)))

rss <- results$rss
aic <- n*log(rss/n) + 2*(2:p_max)
plot(1:(p_max-1), aic, xlab = '# predictors', ylab = 'AIC')
```



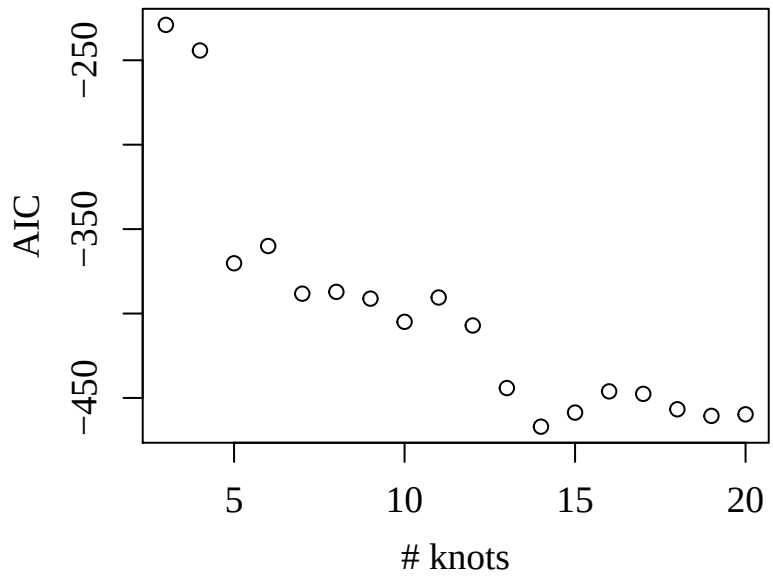

```
results$which[which(aic == min(aic)), ]
```

```
## (Intercept)    Air.Flow  Water.Temp  Acid.Conc.
##        TRUE        TRUE        TRUE       FALSE
```

Only the degree of influence changed in this case, and not the final model. However, it is not unreasonable to imagine a scenario where the same observation is highly influential in one model but not another. Quoting Glen_B on if "outlier detection should be done before or after model selection?": "You can't separate the two. . . some kind of iterative process might work." If outliers are suspected to be an issue, one could use a form of robust regression in the model selection process that is less sensitive to outliers.

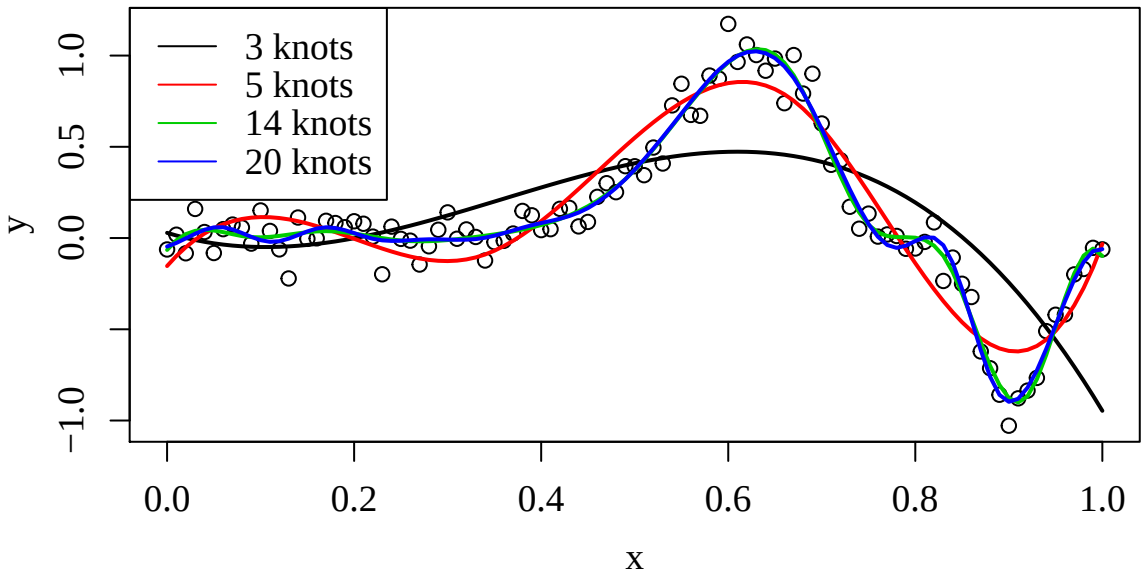
Exercise 7: Selecting the optimal number of cubic B-spline knots. I plot AIC as a function of knot number.

```
library(splines)  # For bs().
set.seed(1)
funky <- function(x) sin(2*pi*x^3)^3
x <- seq(0,1,by=0.01)
y <- funky(x) + 0.1*rnorm(101)
n <- length(x)

nknots = 3:20
aic = vector('numeric', length(nknots))
for (i in 1:length(nknots)) {
  nknot <- nknots[i]
  p <- nknot + 1

  lmod <- lm(y ~ bs(x, nknot))
  rss <- deviance(lmod)
  aic[i] <- n*log(rss/n) + 2*p
}

plot(nknots, aic, xlab = '# knots', ylab = 'AIC')
```



Four selected spline models are overlaid, including the minimum AIC at 14 knots.

```
plot(x, y)
selected_knots <- c(3, 5, 14, 20)
i <- 0
for (nknot in selected_knots) {
  i <- i + 1
  lmod <- lm(y ~ bs(x, nknot))
  lines(x, predict(lmod), col = i, lwd = 2)
}
legend('topleft', legend = paste(c(3, 5, 14, 20), 'knots'), lty = 1, col = 1:4)
```



The 3-knot model model is a clear underfit, hence has substantially larger AIC. Adding two knots yields a large improvement, but the fit is not flexible enough. The 14-knot, optimal AIC model improves the fit dramatically and begins to model the curvature near $x = 0.8$. The 20-knot model only improves the optimal AIC fit (reduces RSS) slightly at the cost of 6 additional parameters, increasing the AIC.

Chapter 11

Shrinkage Methods

11.1 Principal Components

Terminology. In Figure 11.3, Faraway plots each element of $\boldsymbol{u}_j$, $j = 1, 2, 3$, against the predictor it represents. In the text he calls the linear combinations $\boldsymbol{u}_j$ *loadings*, while in the figure he calls them *eigenvectors*. The loadings are just the normalized eigenvectors of the covariance matrix $\boldsymbol{\Sigma}$ scaled by their corresponding eigenvalue. Since any nonzero scalar multiple of an eigenvector is an eigenvector, loadings can also be considered eigenvectors, but not the normalized ones in the rotation matrix returned by `prcomp()` such that $\boldsymbol{u}_j \cdot \boldsymbol{u}_j = 1$. For instance,

```
library(faraway)
meatpca <- prcomp(meatspec[1:172, ])
meatpca$rotation[, 1] %*% meatpca$rotation[, 1]
```

```
##      [,1]
## [1,]    1
```

In Figure 11.5, Faraway plots the standard deviations of each principal component. In the text, he mentions square roots of the eigenvalues. These two are equivalent; another way to say it is that each eigenvalue represents the proportion of variance explained in their corresponding eigenvector $\boldsymbol{u}_j$.

PCA (original) coefficient plots. Plots like Figure 11.4(right) show (shrinked) regression coefficients of the original variables, using the reduced PCR model. To see the connection between the coefficients of the original variables and the PCR model, write the PCR model as

$$\mathbb{E}\, y = \beta_1 \mathtt{PC}_1 + \beta_2 \mathtt{PC}_2 + \beta_3 \mathtt{PC}_3 + \beta_4 \mathtt{PC}_4.$$

[There is no intercept because the data is centered (and usually scaled as well).] $\mathtt{PC}_j$, $j = 1, 2, 3, 4$, is a linear combination of the original 100 x variables, hence this regression equation can be expressed as

$$\mathbb{E}\, y = \gamma_1 x_1 + \gamma_2 x_2 + \cdots + \gamma_{100} x_{100}$$

where each γ_i is a linear combination of PC coefficients β_j. The coefficients γ_i are plotted in Figure 11.4(right).

11.2 Partial Least Squares

11.3 Ridge Regression

Ridge regression and centering and scaling. Faraway emphasizes the importance of centering because ordinarily we would not want to shrink the model intercept to zero; indeed, our results should not depend on a location shift. If we center the data, we don't need to include an intercept in the model. Alternatively, we could apply a ridge penalty on all coefficients except the intercept; this is done in `lm.ridge()` used in the text, hence why `trainmeat` is not centered.

When obtaining predictions, the text also mentions rescaling the predictors and returning the intercept. As `lm.ridge` has no `predict()` method, we have to extract the coefficients from the fit object `rgmod` ourselves.

```
library(MASS)
rgmod <- lm.ridge(fat ~ ., meatspec[1:172, ], lambda = seq(0, 5e-8, len=21))
```

However, internally the coefficients are scaled and centered. The latter can be checked from

```
dim(rgmod$coef)
```

```
## [1] 100  21
```

We expect 1 (intercept) + 100 (predictor) coefficients for each of the 21 values of λ, but further examination of `rgmod$coef` shows no intercept is present. Instead, we must use the `coef()` function which rescales the internal `lm.ridge` coefficients and returns the intercept.

```
dim(coef(rgmod))
```

```
## [1]  21 101
```

We can check the scaling factor by examining `rgmod$scale`. Using it, the two methods for obtaining predictor coefficients agree.

```
sum(coef(rgmod)[1, 2:101] == rgmod$coef[, 1]/rgmod$scales)
```

```
## [1] 100
```

Generalized cross-validation (GCV). GCV is a generalization of leave-one-out cross-validation (LOOCV), which is a special case of k-fold cross validation where the number of folds is equal to the sample size: $k = n$. In particular, LOOCV does not depend on how the k splits in the data are made, and therefore in some cases like OLS or ridge regression, a simple analytic formula exists. (Otherwise, it can be computationally expensive.) Consequently, LOOCV requires a model to be fit a single time. GCV operates similarly, which is why Faraway says GCV is "easier to compute". GCV improves upon LOOCV by being invariant to rotations of the original data, see this article for details.

```
length(coef(rgmod)[1, ])
```

```
## [1] 101
```

```
length(rgmod$coef[, 1])
```

```
## [1] 100
```

11.4 Lasso

Exercises

```
library(faraway)  # For data and sumary().
```

Exercise 4: Comparing variable selection and shrinkage methods for model prediction optimization. Split the data into a training and test set, as described.

```
df <- fat[, !(names(fat) %in% c('brozek', 'density'))]
n <- nrow(df)
test_ind <- seq(1, n, by = 10)
test <- df[test_ind, ]
train <- df[-test_ind, ]
```

We will use the RMSE to quantify predictions using

```
get_rmse <- function(ypred, yobs) {
  sqrt(mean((ypred - yobs)**2))
}
```

and store RMSEs for each method in

```
rmse <- vector('numeric', 6)
methods <- c('OLS', 'AIC', 'PCR', 'PLS', 'ridge', 'LASSO')
names(rmse) <- methods
```

where I have included LASSO as a bonus in part (f). Finally, note that the `fat` dataset contains outliers, examined in Exercises 4.5 and 8.9, but we will ignore them here.

(a) OLS.

```
lmod <- lm(siri ~ ., train)
ypred <- predict(lmod, newdata = test)
(rmse['OLS'] <- get_rmse(ypred, test$siri))
```

```
## [1] 1.946
```

(b) Exhaustive AIC search yields a model with the following 9 predictors.

```
library(leaps)
ntrain <- nrow(train)
pmax <- ncol(train)
results <- summary(
  regsubsets(siri ~ ., data = train, nvmax = pmax, method = 'exhaustive')
)
rss <- results$rss
aic <- ntrain*log(rss/ntrain) + 2*(2:pmax)
results$which[which(aic == min(aic)), ]
```

```
## (Intercept)         age      weight      height      adipos
##        TRUE       FALSE        TRUE       FALSE        TRUE
```

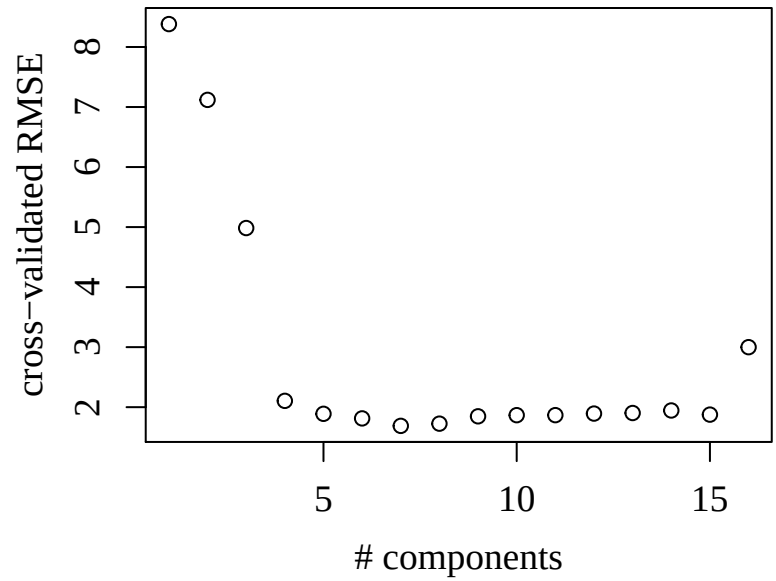
```
##        free        neck       chest       abdom         hip
##        TRUE       FALSE        TRUE        TRUE       FALSE
##       thigh        knee       ankle      biceps     forearm
##        TRUE       FALSE        TRUE        TRUE        TRUE
##       wrist
##       FALSE
```

```
aicmod <- lm(
  siri ~ weight + adipos + free + chest + abdom + thigh + ankle + biceps
    + forearm,
  data = train
)
ypred <- predict(aicmod, newdata = test)
(rmse['AIC'] <- get_rmse(ypred, test$siri))
```

```
## [1] 1.9891
```

(c) 10-fold cross-validated PCR yields a 7-component model.

```
library(pls)
set.seed(1)  # For reproducibility.
pcrmod <- pcr(
  siri ~ ., data = train, validation = 'CV', ncomp = pmax-1, segments = 10
)
cv_rmse <- RMSEP(pcrmod, estimate = 'CV')$val
plot(cv_rmse, xlab = '# components', ylab = 'cross-validated RMSE')
```




```
(best_comp <- which.min(cv_rmse))
```

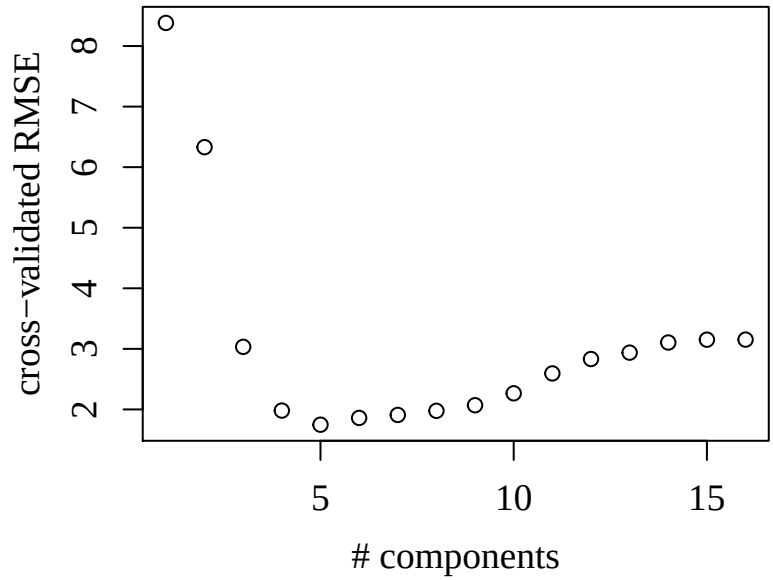
```
## [1] 7
```

```
ypred <- predict(pcrmod, newdata = test, ncomp = best_comp)
(rmse['PCR'] <- get_rmse(ypred, test$siri))
```

```
## [1] 2.0438
```

(d) 10-fold cross-validated PLS yields a 5-component model.

```
plsmod <- plsr(
  siri ~ ., data = train, validation = 'CV', ncomp = pmax-1, segments = 10
)
cv_rmse <- RMSEP(plsmod, estimate = 'CV')$val
plot(cv_rmse, xlab = '# components', ylab = 'cross-validated RMSE')
```




```
(best_comp <- which.min(cv_rmse))
```

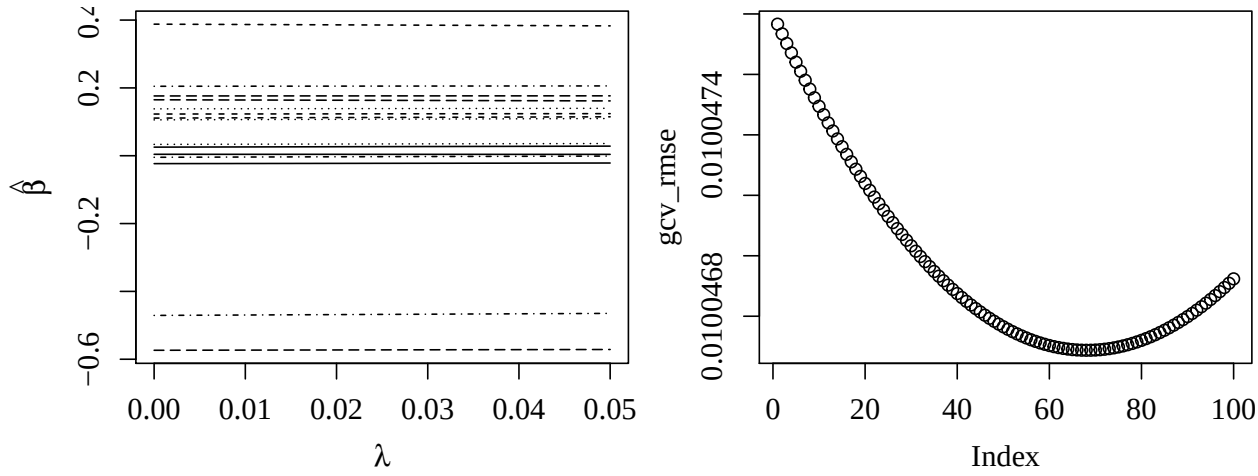
```
## [1] 5
```

```
ypred <- predict(plsmod, newdata = test, ncomp = best_comp)
(rmse['PLS'] <- get_rmse(ypred, test$siri))
```

```
## [1] 2.0284
```

(e) A ridge trace plot, ignoring the intercept, shows estimates are fairly stable even with small λ. Combining with generalized cross-validation suggests optimal $\lambda \approx 0.034$.

```
library(MASS)
ridgemod <- lm.ridge(siri ~ ., train, lambda = seq(0, 5e-2, len = 100))
matplot(
  ridgemod$lambda, coef(ridgemod)[, -1],
  type = 'l', xlab=expression(lambda), ylab=expression(hat(beta)), col=1
)
gcv_rmse <- ridgemod$GCV
plot(gcv_rmse)
```




```
(best_lambda <- which.min(gcv_rmse))
```

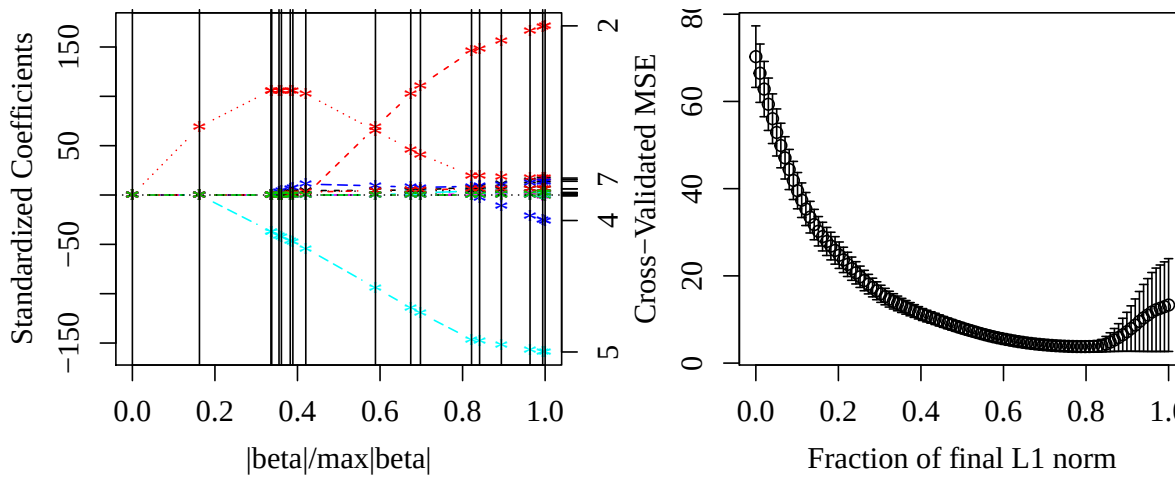
```
## 0.03383838
##         68
```

```
ypred <- (
  cbind(1, as.matrix(test[, names(test) != 'siri']))
    %*% coef(ridgemod)[best_lambda, ]
)
(rmse['ridge'] <- get_rmse(ypred, test$siri))
```

```
## [1] 1.9372
```

(f) 10-fold cross-validated LASSO yields a 10-predictor model.

```
library(lars)
lassomod <- lars(as.matrix(train[, names(train) != 'siri']), train$siri)
plot(lassomod)
cvmod <- cv.lars(as.matrix(train[, names(train) != 'siri']), train$siri)
```




```
cv_mse <- cvmod$cv
(best_s <- cvmod$index[which.min(cv_mse)])
```

```
## [1] 0.78788
```

```
predict(lassomod, s = best_s, type = 'coef', mode = 'fraction')$coef
```

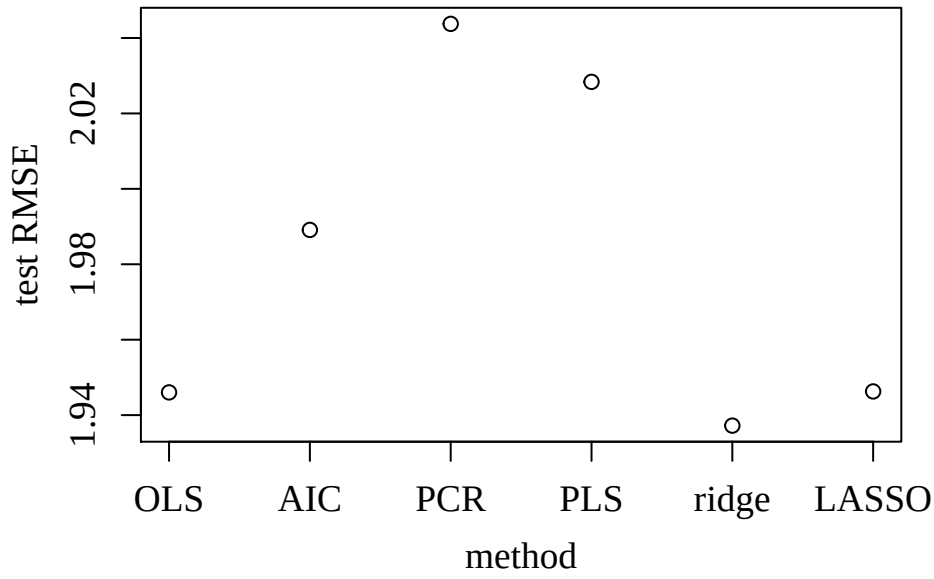
```
##       age    weight    height    adipos      free      neck     chest
##  0.000000  0.308643  0.064777  0.000000 -0.501488  0.000000  0.056243
##     abdom       hip     thigh      knee     ankle    biceps   forearm
##  0.157302  0.000000  0.107655  0.094822  0.037292  0.100538  0.193507
##     wrist
##  0.000000
```

```
ypred <- predict(
  lassomod, newx = test[names(test) != 'siri'], s = best_s, mode = 'frac'
)$fit
(rmse['LASSO'] <- get_rmse(ypred, test$siri))
```

```
## [1] 1.9463
```

Overall comments. Test RMSEs for each method are summarized in the following plot.

```
plot(rmse, xlab = 'method', xaxt = 'none', ylab = 'test RMSE')
axis(1, at = 1:length(methods), labels = methods)
```



In this instance, ridge regression has the best predictive accuracy. OLS with all variables is second, closely followed by LASSO. PCR and PLS perform poorly in comparison. The variable selection by AIC model, despite an exhaustive search, predicts worse than LASSO; this may be because the LASSO model has one more predictor, but is more likely a result of LASSO's additional shrinkage, reducing overfitting. The variables selected are not identical, but the majority are the same.

Chapter 12

Insurance Redlining – A Complete Example

```
library(faraway)
library(ggplot2)
```

12.1 Ecological Correlation

12.2 Initial Data Analysis

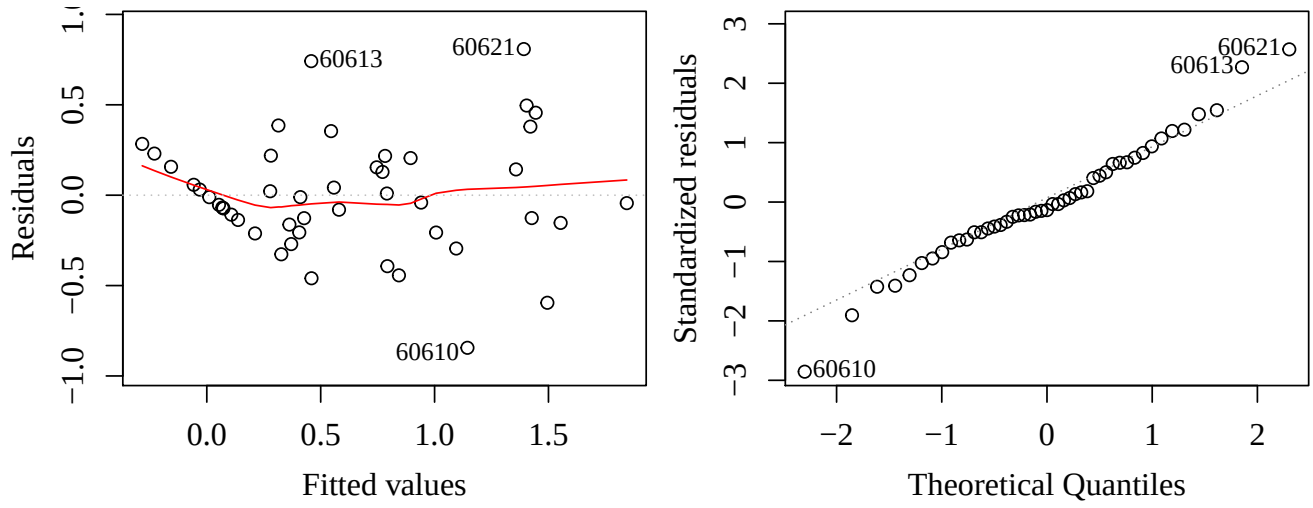
Does the log(income) transformation matter? Not really. None of the coefficients (except income, of course) change substantially; they are well within 1 s.e. of the log(income) model.

```
lmod <- lm(involact ~ race + fire + theft + age + income, chredlin)
sumary(lmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.60898    0.49526   -1.23  0.22585
## race         0.00913    0.00232    3.94  0.00031
## fire         0.03882    0.00844    4.60    4e-05
## theft       -0.01030    0.00285   -3.61  0.00083
## age          0.00827    0.00278    2.97  0.00491
## income       0.02450    0.03170    0.77  0.44398
##
## n = 47, p = 6, Residual SE = 0.335, R-Squared = 0.75
```

The model diagnostic plots are also quite similar.

```
plot(lmod, which = 1:2)
```



12.3 Full Model and Diagnostics

12.4 Sensitivity Analysis

More on model uncertainty. As the text states, typical regression output does not account for variation in the data analysis procedure to select the 'optimal' model. Were we to repeat the analysis on a different sample drawn from the same population, in addition to sampling variance, we may choose another model entirely. The result: overoptimistic estimates and intervals that are too narrow. Data splitting and the bootstrap are two methods to remedy this, described next.

Data splitting. The conceptually simplest approach begins by splitting the data into two equal parts. We build our model using one part, then use this model on the other part to obtain our estimates.[1] This is similar to the training and test set approach in Chapter 11. Because the test set doesn't inform the model, our data analysis procedure will not find the 'optimal' model w.r.t. to the test set, so our estimates shouldn't be overoptimistic. This approach is somewhat unsatisfying as it is an inefficient use of data; the model is trained on less data (training set) and estimates from that model are obtained from less data (test set). A better approach uses the bootstrap, discussed in Faraway's 1992 paper "On the Cost of Data Analysis".

Bootstrap. The bootstrap is an effective way to obtain the distribution of a complex statistic or set of statistics. Here, the statistics of interest are regression output like predictions and predictor coefficients. If we ignore variability from the data analysis procedure, we can often appeal to distributional or asymptotic assumptions to obtain prediction and confidence intervals. If those assumptions do not hold, we can use the bootstrap discussed in Section 3.6. In either case, we account for sampling variation.
Accounting for variation in the data analysis procedure is a much harder problem analytically — how do you account for variation created from a procedure that tests nonlinearity of all predictors? On some datasets, this procedure will conclude a linear fit is sufficient; on other datasets (from the same population), it will conclude a nonlinear fit is needed. The bootstrap provides a straightforward recipe, provided we have a fixed set of rules in our data analysis procedure.

• Define a data analysis procedure to determine the 'optimal' model. An example is provided in Faraway's 1992 paper. For instance, start with an initial model containing linear terms for all

[1] The split doesn't have to be equal, in general. If the model training part is larger, we can be more confident in the form of our model at the cost of more uncertain estimates obtained from the smaller part.

predictors, remove outliers based on some criteria, check for heteroscedasticity and add weights if necessary, test for transformations and add significant terms, then perform variable selection.

• Resample from the original data. Apply data analysis procedure in step (1) and record statistics of interest (e.g. predictions at point $\boldsymbol{x}_0$, coefficients and associated p-values).

• Repeat step (2) n_{boot} times. All recorded values of statistic T are the estimated distribution of T. The larger n_{boot} is, the better our estimate. The standard deviation of the estimated distribution is the standard error, and confidence intervals can be obtained from its quantiles.

The main drawback of the bootstrap method is the automated data analysis procedure. The automation is necessary from both a practical and statistical perspective; analyzing 100 datasets is time consuming and a human analyst's subjective decisions, e.g. examination of a residual plot, may be inconsistent. Unfortunately, this means the automated procedure is less flexible than a typical data analysis, particularly in its lack of use of graphical methods recommended in Chapter 6.
Another difficulty arises from models that contain different variables and/or transformations. Regression coefficients can have completely different interpretations in the presence of nonlinearities. Instead of recording β_i directly, we can record the effect of a change in x_i on the response y. Recall in the simple linear regression model

$$\mathbb{E}\, y = \beta_0 + \beta_1 x_1.$$

One interpretation of β_1 is as follows: increasing x_1 by 1 increases the mean response by β_1. This can be shown formally by computing $\mathbb{E}\, y(x_1 + 1) - \mathbb{E}\, y(x_1)$. In a similar fashion for a quadratic model

$$\mathbb{E}\, y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2,$$

$\mathbb{E}\, y(x_1 + 1) - \mathbb{E}\, y(x_1) = \beta_1 + \beta_2[(x_1 + 1)^2 - x_1^2]$. The change in response now also depends on the value of x_1. Consequently when nonlinearities are present, we must choose which point(s) we wish to record the effect of a change in x_i, which doesn't have to be 1. Other methods are discussed in Section 3.3 of Faraway's 1992 paper.

12.5 Discussion

Exercises

```
library(faraway)  # For data, sumary(), and vif().
```

I prefer graphical summaries of a data frame over numerical summaries provided by `summary()`. I will use the following function that takes a data frame and plots each predictor: barplots for categorical data and histograms for continuous data.

```
summary_plot <- function(df) {
  for (var in names(df)) {
    data <- df[[var]]
    if (is.factor(data)) {
      plot(data, xlab = var)
    } else {
```

```
      hist(data, main = '', xlab = var)
    }
  }
}
```

I'll also use the `corrplot()` function from the `corrplot` library to visualize correlations.

```
library(corrplot)
```

Finally, I'll use the following function to quickly compute condition numbers.

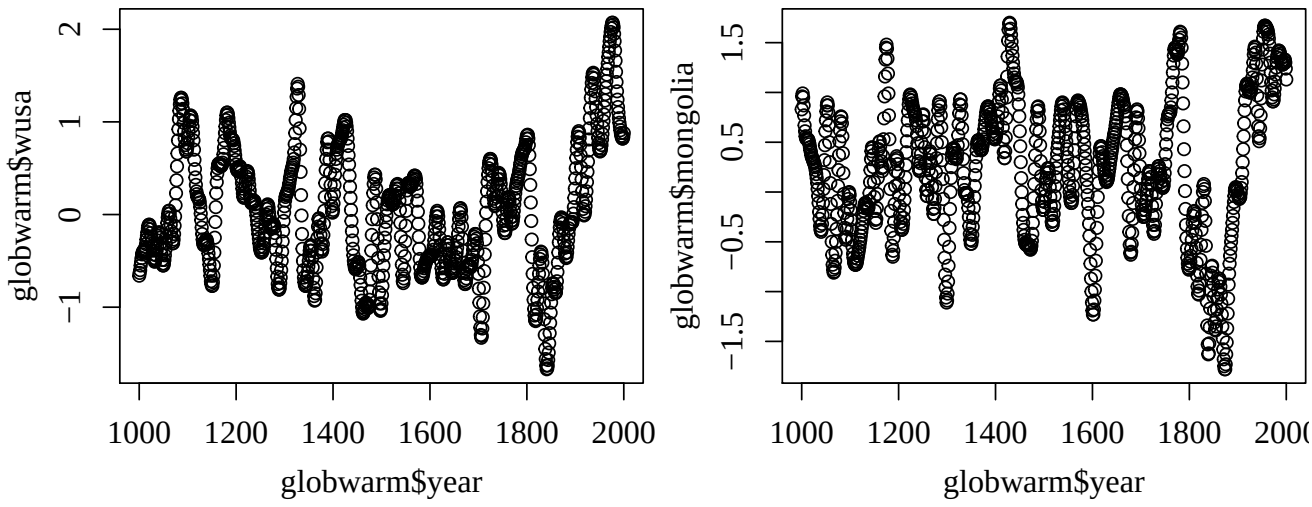
```
cond_num <- function(lmod) {
  X <- model.matrix(lmod)[, -1]
  e <- eigen(t(X) %*% X)
  return(sqrt(e$values[1] / e$values))
}
```

Exercise 1: Predicting past temperatures via proxy measurements.
Data examination. The first step is to learn about the data using `?globwarm`. There, we learn `nhtemp` is the northern hemisphere average temperature (in Celsius) but is only available for years 1856–2000. We wish to build a model that can predict `nhtemp` from the eight climate proxies because proxy data is available for years 1000-2000. Since the model requires available responses for training, for most of my analysis I'll use a subset of data where `nhtemp` is available.

```
df <- globwarm[!is.na(globwarm$nhtemp), ]
```

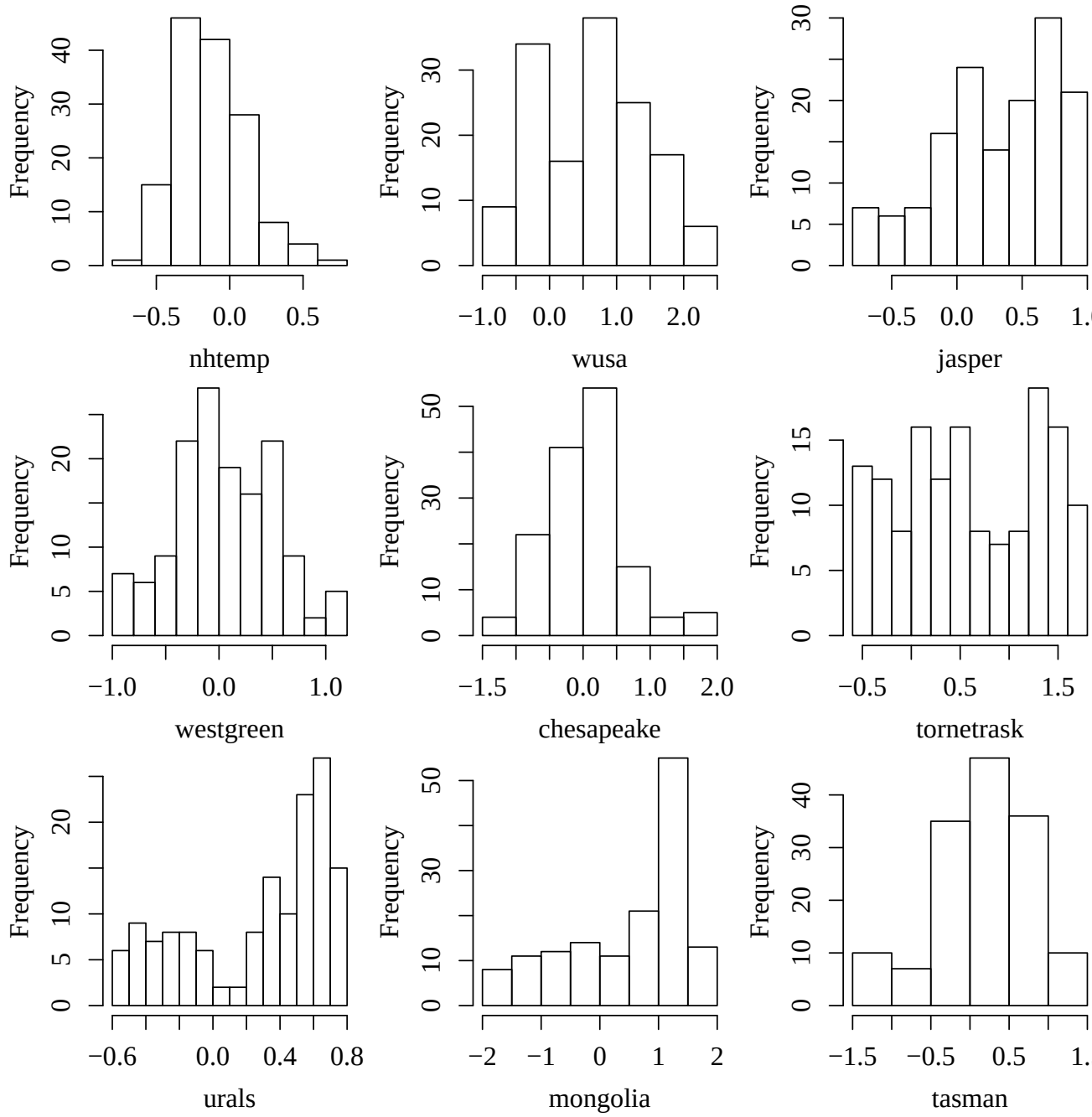
I'll return to the full dataset for predictions once I've chosen the model. Finally, I won't include `year` as a predictor in the regression model as we are interested in past prediction dating back to 1000. This would be a serious extrapolation in terms of time while the proxies are somewhat periodic, for example:

```
plot(globwarm$year, globwarm$wusa)
plot(globwarm$year, globwarm$mongolia)
```



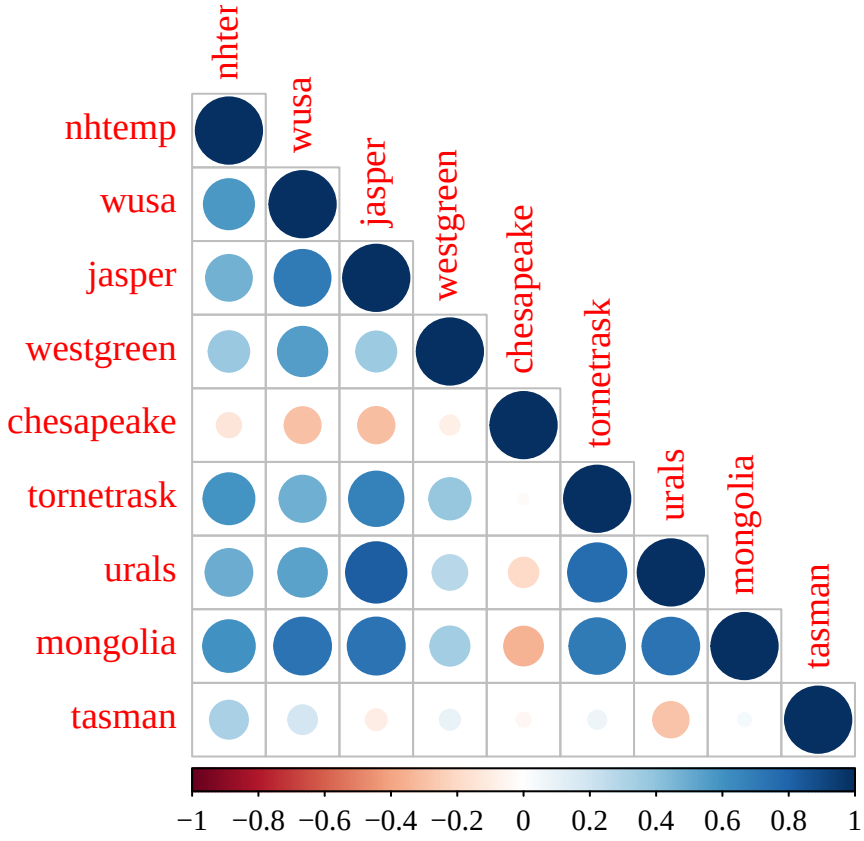
Note that this does not exclude the use of `year` in informing the model, e.g. model diagnostics. A graphical summary of the response `nhtemp` and predictors is shown next.

```
summary_plot(df[, !(names(df) %in% 'year')])
```



Tree ring proxies `jasper`, `urals`, and `mongolia` are strongly skewed left. `tornetrask`, another tree ring proxy, appears nearly uniform. The other variables, including the response, are approximately symmetric and unimodal. It is also worth considering relationships between variables. I do this using Spearman's rank correlation, which measures the strength of monotonic (but not necessarily linear) relationships.

```
correlations <- cor(df[, !(names(df) %in% 'year')], method = 'spearman')
corrplot(correlations, type = 'lower')
```



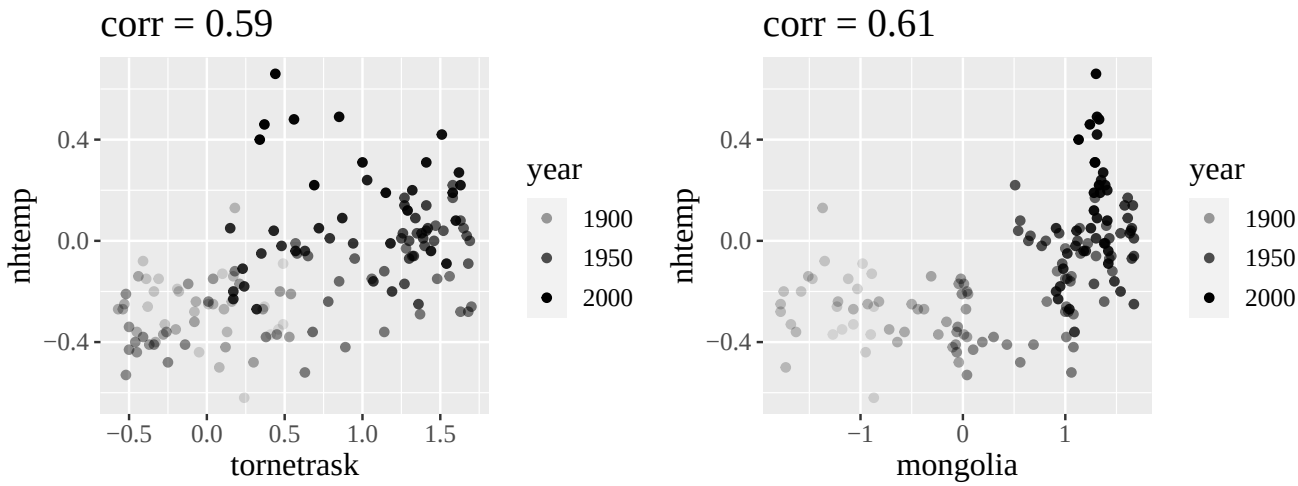
A few variables have multiple strong correlations, e.g. `mongolia` and `urals`, so I may need to address collinearity later although it is less problematic for prediction. There is little noise in the proxy data from year to year, so scatterplots of proxies exhibit interesting relationships over time even when there is little correlation.

```
library(gridExtra)  # Plot multiple ggplots in a grid.
plot1 <- ggplot(df) + geom_point(aes(jasper, urals, alpha = year)) +
  ggtitle(paste('corr =', round(correlations['jasper', 'urals'], 2)))
plot2 <- ggplot(df) + geom_point(aes(mongolia, tasman, alpha = year)) +
  ggtitle(paste('corr =', round(correlations['mongolia', 'tasman'], 2)))
grid.arrange(plot1, plot2, ncol = 2)
```

Scatterplots of response against correlated proxies yield more familiar plots; indeed the response is

more variable which is why we're modeling it with a random error.

```
plot1 <- ggplot(df) + geom_point(aes(tornetrask, nhtemp, alpha = year)) +
  ggtitle(paste('corr =', round(correlations['tornetrask', 'nhtemp'], 2)))
plot2 <- ggplot(df) + geom_point(aes(mongolia, nhtemp, alpha = year)) +
  ggtitle(paste('corr =', round(correlations['mongolia', 'nhtemp'], 2)))
grid.arrange(plot1, plot2, ncol = 2)
```



The left (right) plot exhibits a (non)linear trend. Because prediction is of primary interest, I will consider nonlinear transformations later.

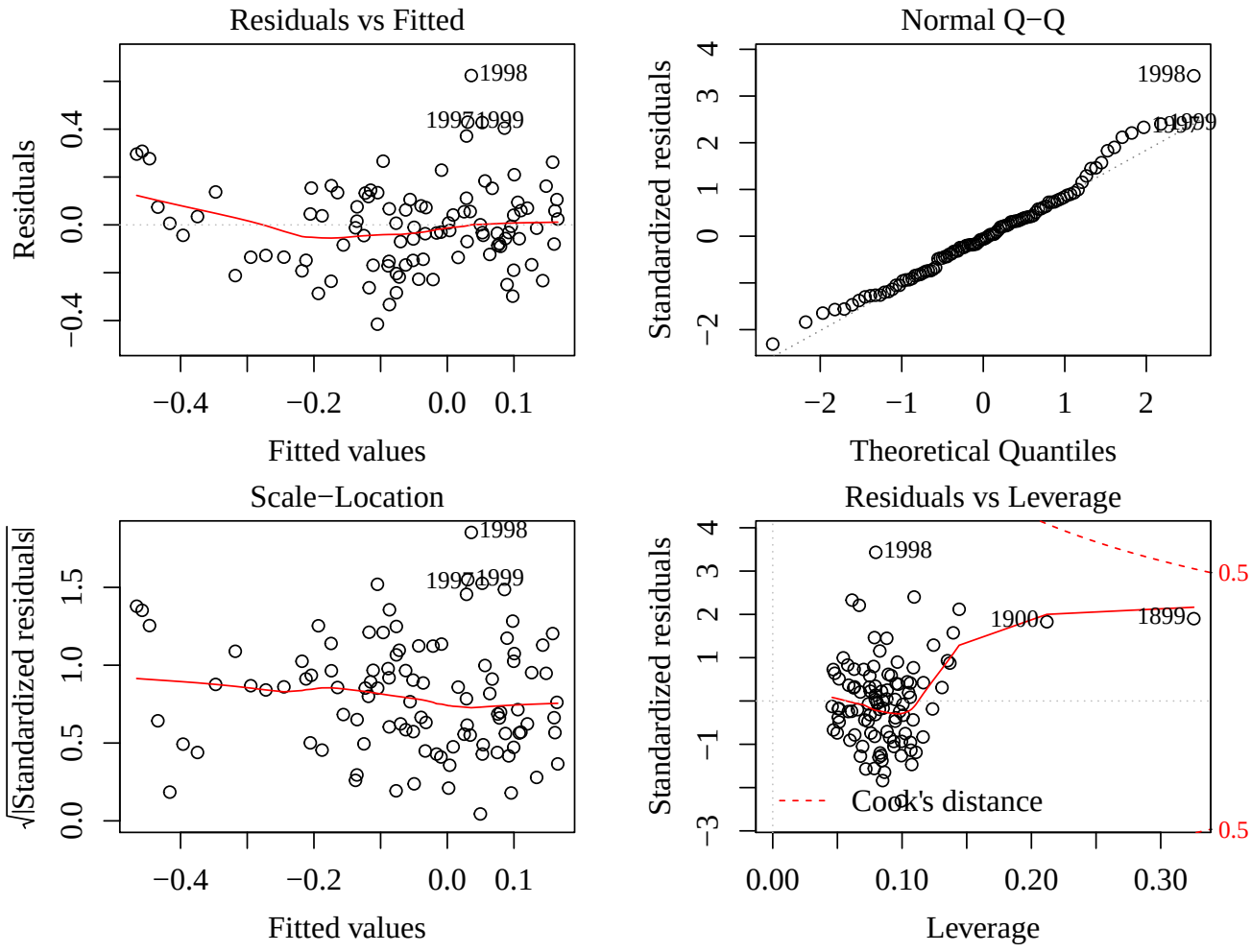
Model diagnostics and (re)selection. For now, nothing is particularly alarming, so I'll start with a simple model linear in all predictors. Since we're interested in past prediction, I'll reserve 30% of the oldest data as a test set to select the best model, measured by RMSE, and build the model on the rest of the data.

```
n <- nrow(df)
test <- df[1:floor(n*.3), ]
train <- df[(floor(n*.3)+1):n, ]
lmod <- lm(nhtemp ~ . - year, train)
sumary(lmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.4211     0.1398   -3.01   0.0033
## wusa          0.1490     0.0789    1.89   0.0620
## jasper       -0.3024     0.1083   -2.79   0.0064
## westgreen    -0.0346     0.0763   -0.45   0.6511
## chesapeake   -0.0243     0.0547   -0.44   0.6582
## tornetrask    0.0887     0.0531    1.67   0.0984
## urals         0.4939     0.2193    2.25   0.0267
## mongolia      0.0324     0.0707    0.46   0.6481
## tasman        0.1048     0.0544    1.93   0.0570
##
## n = 102, p = 9, Residual SE = 0.189, R-Squared = 0.42
```

The residual standard error is moderate: $2 \times$ RSE is more than the interquartile range. I'll examine R's default diagnostic plots next.

```
plot(lmod)
```



• **Residuals vs Fitted.** There is no obvious trend suggestive of poor model structure. There are a few large residuals, particularly the later years implying the model underestimates the latest data.

• **Normal Q-Q.** The right tail violates the normality assumption, which seem to be the latest data.

• **Scale-Location.** The residuals do not deviate much from the constant variance assumption.

• **Residuals vs Leverage.** The oldest points in the dataset, 1899 and 1900, have the highest leverage and influence, but not high enough for me to consider their removal.

Aside from normality of errors, which is the least important assumption, the simple model looks OK. One could argue the large residuals are problematic for past predictions; perhaps they are indicative of global warming in recent years, a small range of time with distinct behavior that my simple linear model cannot capture. I'll test this formally using leave-one-out Studentized residuals, which are approximately distributed as $\mathcal{T}(n-p-1)$, where p is the number of parameters. Using a Bonferroni correction at the $\alpha = 0.05$ level,

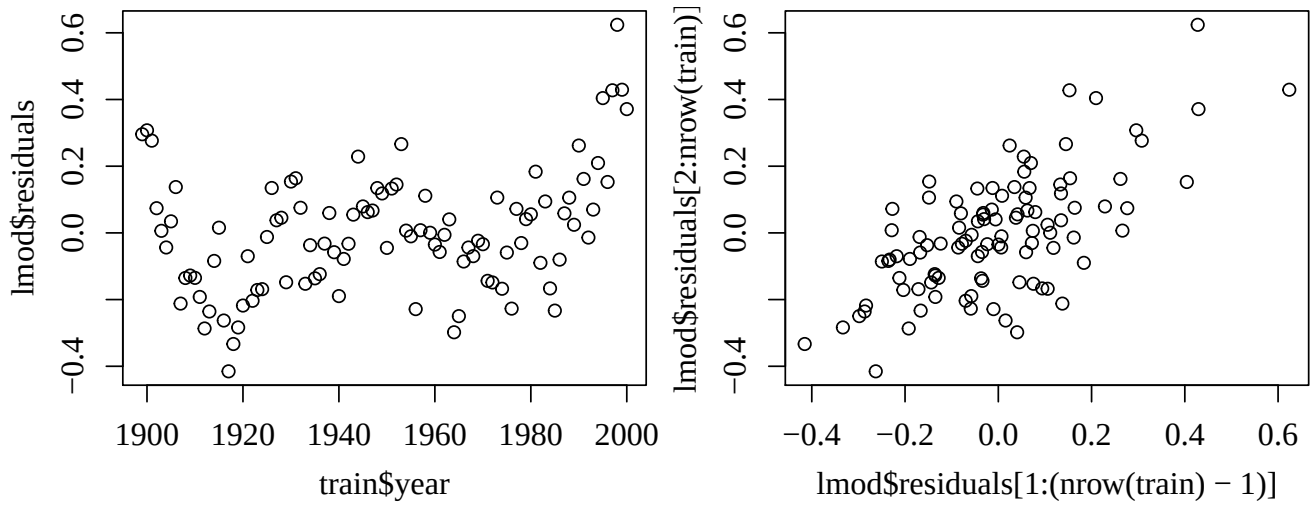
```
tresid <- rstudent(lmod)
which(abs(tresid) > abs(qt(.05/n, lmod$df.residual - 1)))
```

```
## 1998
```

```
##  100
```

only the largest residual is a statistical outlier. Before I consider removing it, I'll investigate a larger issue that is difficult to observe from the above four diagnostics: the independent errors assumption. This assumption can be difficult to check in general, and often requires some domain knowledge. Time series data is often correlated because variables don't change instantaneously. For instance, today's temperature is correlated with tomorrow's; dramatic swings in daily temperature are rare. Unless the model perfectly predicts these seasonal correlations, I'd expect residuals to be correlated from year to year. This is hard to see from the diagnostic plots because `year` has not yet informed our model, but can be seen in plots of residuals against `year` and $\widehat{\varepsilon}_{i+1}$ against $\widehat{\varepsilon}_i$.

```
plot(train$year, lmod$residuals)
plot(lmod$residuals[1:(nrow(train)-1)], lmod$residuals[2:nrow(train)])
```



To address the correlated errors, I'll use generalized least squares (GLS) with an AR1 correlation structure. Specifically, I'll model the the errors as a linear 1-step lag:

$$\varepsilon_{i+1} = \phi\varepsilon_i + \delta_i$$

where $\delta_i \sim \mathcal{N}(0, \tau^2)$. The resultant covariance matrix of the errors has the form

$$\boldsymbol{\Sigma} = \sigma^2 \begin{bmatrix} 1 & \phi & \phi^2 & \cdots & \phi^n \\ & 1 & \phi & \cdots & \phi^{n-1} \\ & & 1 & \cdots & \phi^{n-2} \\ & & & \ddots & \vdots \\ & & & & 1 \end{bmatrix}.$$

We can then transform the response and design matrix to $\boldsymbol{y}' = \boldsymbol{S}^{-1}\boldsymbol{y}$ and $\boldsymbol{X}' = \boldsymbol{S}^{-1}\boldsymbol{X}$, where $\boldsymbol{S}$ is obtained from the Choleski decomposition $\boldsymbol{\Sigma} = \boldsymbol{S}\boldsymbol{S}^T$, and regress $\boldsymbol{y}'$ on $\boldsymbol{X}'$:

$$\boldsymbol{y}' = \boldsymbol{X}'\boldsymbol{\beta} + \boldsymbol{\varepsilon}'.$$

If $\boldsymbol{\Sigma}$ is correctly specified, the transformed errors $\boldsymbol{\varepsilon}' = \boldsymbol{S}^{-1}\boldsymbol{\varepsilon}$ are i.i.d.[2] See Chapter 8 for more details.

I'll implement GLS using the `nlme` package.

[2]There appear to be two estimated parameters in the AR1 covariance matrix: σ and ϕ. However, we don't actually need to specify σ for GLS to work. We only need to specify a covariance matrix proportional to $\boldsymbol{\Sigma}$; the constant σ^2 just amounts to a change in scale which is irrelevant in the transformed regression model.

```
library(nlme)
glmod <- gls(nhtemp ~ . - year, correlation = corAR1(form = ~ year), train)
summary(glmod$modelStruct)
```

```
## Correlation Structure: AR(1)
##  Formula: ~year
##  Parameter estimate(s):
##     Phi
## 0.85698
```

We observe strong correlation between residuals. Let's see how the coefficients change:

```
summary(glmod)$tTable
```

```
##                  Value Std.Error   t-value p-value
## (Intercept) -0.1873802   0.31072 -0.603050 0.54794
## wusa         0.0076926   0.18299  0.042039 0.96656
## jasper      -0.1792867   0.32053 -0.559352 0.57727
## westgreen    0.0625113   0.16329  0.382829 0.70272
## chesapeake   0.0589333   0.12426  0.474266 0.63642
## tornetrask   0.0163659   0.13112  0.124819 0.90094
## urals        0.3641912   0.55790  0.652786 0.51550
## mongolia     0.0298747   0.20151  0.148252 0.88246
## tasman       0.1027733   0.15453  0.665076 0.50765
```

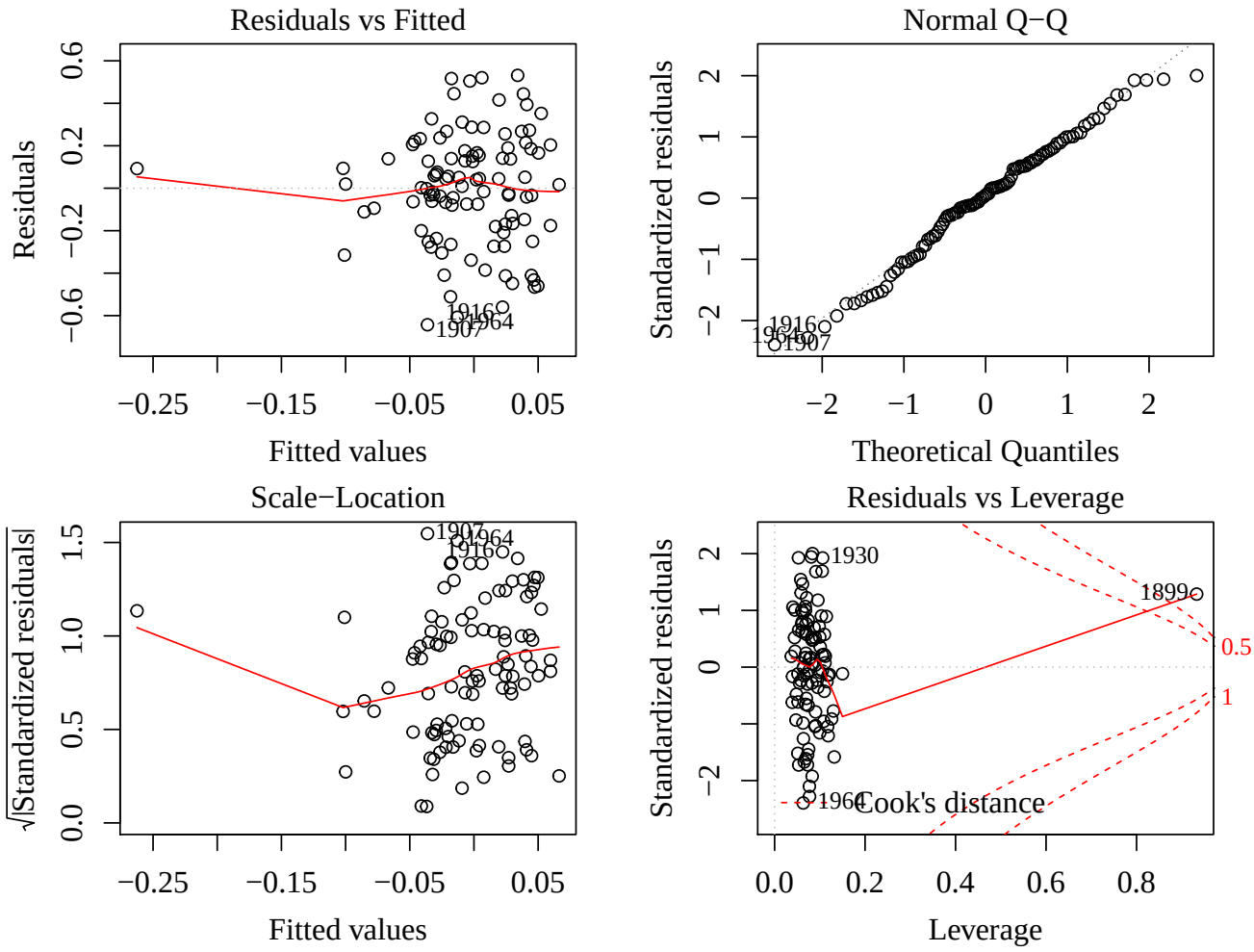
Point estimates change and there is substantially more uncertainty. Remember, though, from a statistical standpoint this model should be better, e.g. its residuals should no longer be correlated. `gls` objects do not have the four graphical diagnostics we are accustomed to. I'll implement GLS using `lm()` instead, with the help of other functions from `nlme` to specify the covariance structure $\mathbf{\Sigma}$ and transform the data accordingly, using the estimated ϕ above.

```
Sigma <- corMatrix(glmod$modelStruct$corStruct)
S <- t(chol(Sigma))
y <- solve(S) %*% train$nhtemp  # y'
X <- solve(S) %*% model.matrix(nhtemp ~ . - year, train)  # X'
rownames(y) <- rownames(X) <- rownames(train)
glmod <- lm(y ~ X + 0)
sumary(glmod)
```

```
##              Estimate Std. Error t value Pr(>|t|)
## X(Intercept) -0.18738    0.31072   -0.60     0.55
## Xwusa         0.00769    0.18299    0.04     0.97
## Xjasper      -0.17929    0.32053   -0.56     0.58
## Xwestgreen    0.06251    0.16329    0.38     0.70
## Xchesapeake   0.05893    0.12426    0.47     0.64
## Xtornetrask   0.01637    0.13112    0.12     0.90
## Xurals        0.36419    0.55790    0.65     0.52
## Xmongolia     0.02987    0.20151    0.15     0.88
## Xtasman       0.10277    0.15453    0.67     0.51
##
## n = 102, p = 9, Residual SE = 0.277, R-Squared = 0.03
```

We've recovered the same GLS model using `lm()`. On to diagnostics, which appear substantially different:

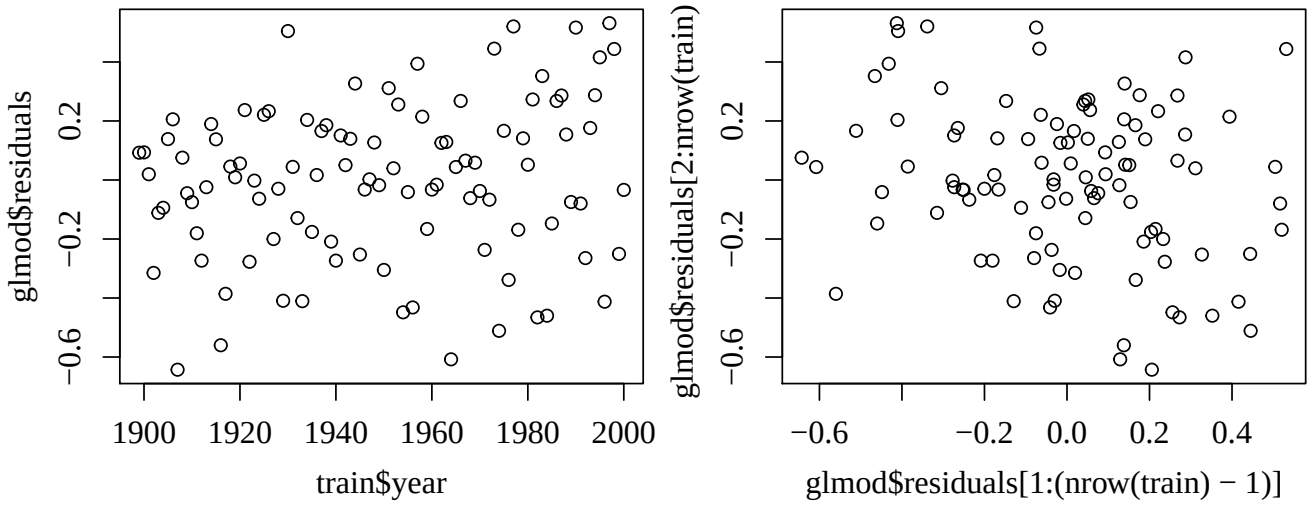
```
plot(glmod)
```



1988 is no longer an outlier—in fact, no Studentized residual is statistically significant after Bonferroni correction at $\alpha = 0.05$. 1899 is now a high leverage point point with large influence. However, this is an artifact of the transformed design matrix $\boldsymbol{X}'$, whose first row is identical to $\boldsymbol{X}$. For this reason, I won't remove this point.[3] The qqplot is improved, though still has a problematic right tail. The GLS model introduces heteroscedasticity, which I'll address later. Importantly, residual plots no longer show correlation.

```
plot(train$year, glmod$residuals)
plot(glmod$residuals[1:(nrow(train)-1)], glmod$residuals[2:nrow(train)])
```

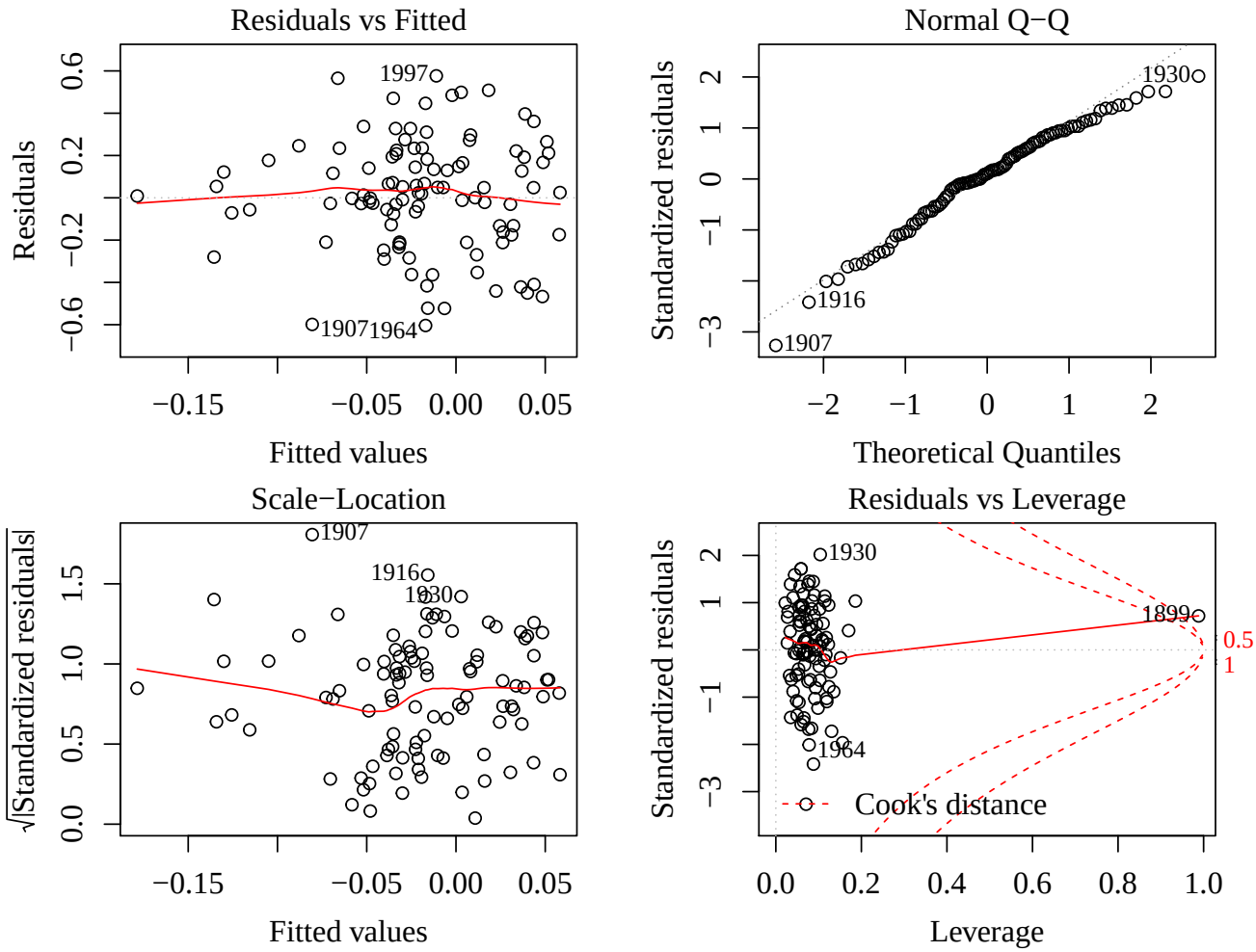
[3]Removal leads to a model with substantial changes to estimates and errors. Prediction on the test set is substantially worse.



To check the degree of collinearity, I examined the condition numbers. The largest was 11.75677, so I won't take any special precautions.

We can't address heteroscedasticity using a Box–Cox transformation of the response because some $y < 0$. Residuals plots against each predictor don't suggest any predictor transformations. Instead, I'll add another candidate model using weights. Since `year` and `nhtemp` are correlated and higher predicted `nhtemp` values have more variance, I'll use some form of temporal weighting.

```
wts <- 1/(as.integer(rownames(X)) - min(as.integer(rownames(X))) + 1)**.5
wglmod <- lm(y ~ X + 0, weights = wts)
plot(wglmod)
```



The exact form of the weights was found by trial and error. The resultant weights span a large range; the oldest point is weighed more than 10 times the most recent point, which results in a larger Cook's distance. I'll see whether the weights help in prediction next, skipping variable selection since n/p is not too small and we're focused on prediction.

Optimizing predictive accuracy. I'll grade models on the mean RMSE of their predictions for the test data.

```
get_rmse <- function(ypred, yobs) {
  sqrt(mean((ypred - yobs)**2))
}
```

Since the GLS models were trained on transformed data, I'll compute predictions using extracted model coefficients rather than the `predict()` method.

```
X_test <- model.matrix(nhtemp ~ . - year, test)
get_rmse(X_test %*% coef(lmod), test$nhtemp)
```

```
## [1] 0.30448
```

```
get_rmse(X_test %*% coef(glmod), test$nhtemp)
```

```
## [1] 0.16485
```

```
get_rmse(X_test %*% coef(wglmod), test$nhtemp)
```

```
## [1] 0.22604
```

The AR1 model without the additional weighting performs best. There is the possibility of improvement if we use shrinkage methods. I'll employ ridge regression using the `glmnet` package, which allows us to flexibly shrink predictor coefficients; in particular, we can specify no penalty on the (transformed) intercept term in the GLS models.

```
library(glmnet)
set.seed(1)  # For reproducibility (from cross-validation).
ridge_lmod <- cv.glmnet(
  x = as.matrix(train[, !(names(train) %in% c('nhtemp', 'year'))]),
  y = train$nhtemp, alpha = 0, intercept = TRUE
)
get_rmse(X_test %*% coef(ridge_lmod), test$nhtemp)
```

```
## [1] 0.14675
```

```
ridge_glmod <- cv.glmnet(
  X, y, alpha = 0, intercept = FALSE, penalty.factor = c(0, rep(1, ncol(X)-1))
)
get_rmse(X_test %*% coef(ridge_glmod)[2:(ncol(X)+1)], test$nhtemp)
```

```
## [1] 0.2748
```

```
ridge_wglmod <- cv.glmnet(
  X, y, alpha = 0, intercept = FALSE, penalty.factor = c(0, rep(1, ncol(X)-1)),
  weights = wts
)
get_rmse(X_test %*% coef(ridge_wglmod)[2:(ncol(X)+1)], test$nhtemp)
```

```
## [1] 0.18499
```

Surprisingly, the linear model with ridge regression obtains the best accuracy among all models considered. Ridge regression doesn't benefit the GLS models—inspection of the penalized coefficients show all terms but the intercept are shrunk to zero. I'll continue with `glmod` and `ridge_lmod` to compare their predictions. If there are stark differences I'll need to think harder on picking one model.

Making predictions. I'll train the models on all data with available `nhtemp`.

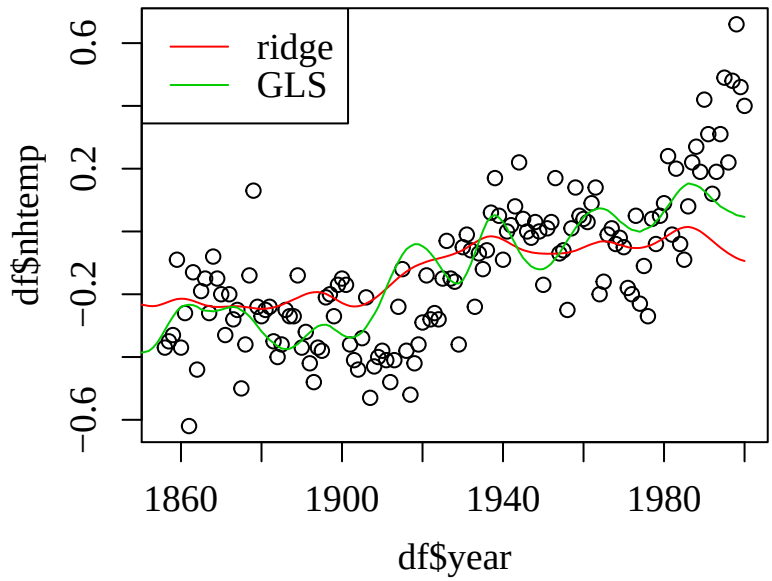
```
ridge_lmod <- cv.glmnet(
  x = as.matrix(df[, !(names(df) %in% c('nhtemp', 'year'))]),
  y = df$nhtemp, alpha = 0, intercept = TRUE
)

glmod <- gls(nhtemp ~ . - year, correlation = corAR1(form = ~ year), df)
```

Prediction on all data are computed next.

```
ridge_lmod_pred <- predict(
  ridge_lmod,
  newx = as.matrix(globwarm[, !(names(globwarm) %in% c('nhtemp', 'year'))])
)
glmod_pred <- predict(glmod, globwarm[, !(names(globwarm) %in% c('nhtemp'))])
```

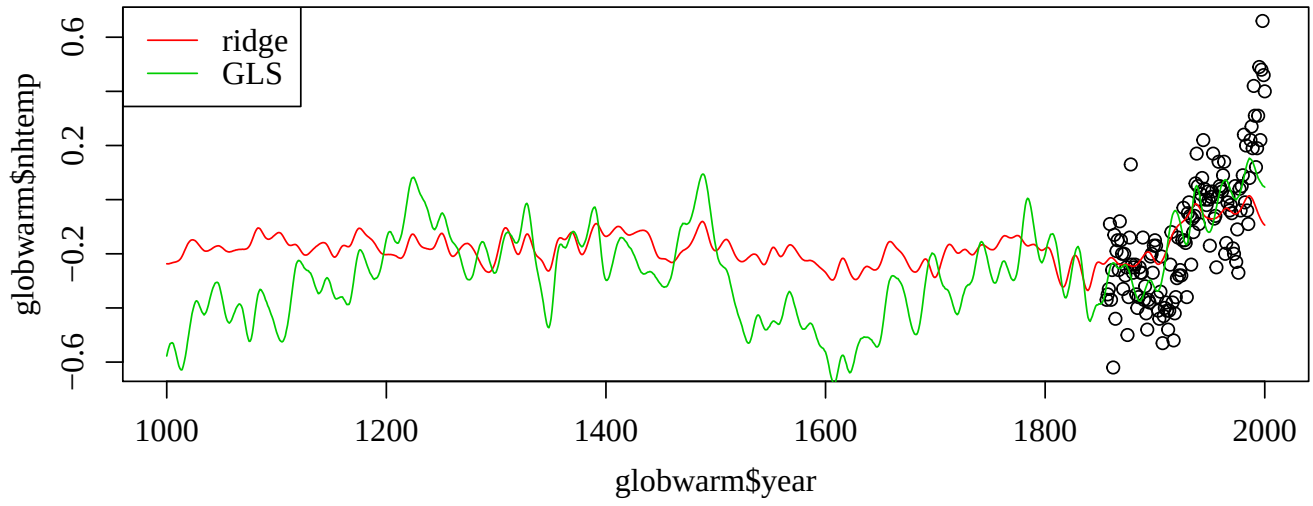
I think it's instructive to first plot the fits on the subset of fully available data.

```
plot(df$year, df$nhtemp)
lines(globwarm$year, ridge_lmod_pred, col = 2)
lines(globwarm$year, glmod_pred, col = 3, lty = 1)
legend('topleft', legend = c('ridge', 'GLS'), lty = 1, col = 2:3)
```



The GLS model better captures temporal variation, while the ridge model hovers around the mean temperature -0.12034. This feature is even more pronounced when we predict back to year 1000, leading to drastically different predictions.

```
plot(globwarm$year, globwarm$nhtemp)
lines(globwarm$year, ridge_lmod_pred, col = 2)
lines(globwarm$year, glmod_pred, col = 3)
legend('topleft', legend = c('ridge', 'GLS'), lty = 1, col = 2:3)
```



Since the ridge model makes no correction for the strongly correlated residuals, other than shrinking estimates, I favor the GLS model. Still, it is instructive to build confidence intervals for both models. In both cases, there is no straightforward formula to produce intervals. The usual OLS formula requires computation of $\boldsymbol{x}_0^T(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{x}_0$ where $\boldsymbol{x}_0$ represents the new data and $\boldsymbol{X}$ is the training data. The problem with GLS is that the model is trained on transformed data $\boldsymbol{X}' = \boldsymbol{S}^{-1}\boldsymbol{X}$, which is on a different scale from $\boldsymbol{x}_0$, so the usual formula doesn't apply. In ridge regression, the

estimate itself is biased (by design) by an unknown amount, which is hard to quantify. I'll rely on the bootstrap to compute prediction intervals, though they do not circumvent the issue of bias in ridge regression. I'll resample rows from the data, adding small amounts of noise when fitting the GLS model so repeated rows don't have the same `year`, throwing off the AR1 fit. To account for variability in the random component, I'll resample residuals from the bootstrap fit.[4]

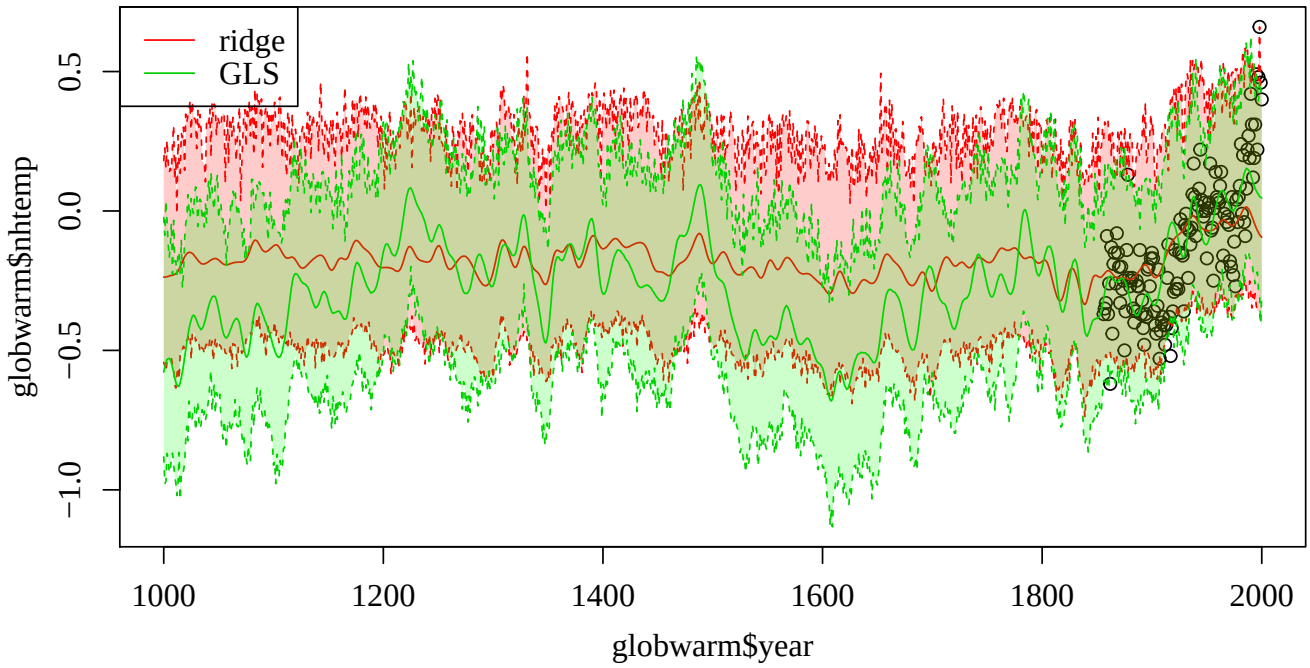
```
nboot <- 200
results <- array(
  dim = c(nboot, nrow(globwarm), 2),
  dimnames = list(1:nboot, globwarm$year, c('ridge', 'GLS'))
)
N <- nrow(globwarm)
for (i in 1:nboot) {
  boot_sample <- sample(1:n, replace = TRUE)
  boot_df <- df[boot_sample, ]
  Xboot <- as.matrix(boot_df[, !(names(df) %in% c('nhtemp', 'year'))])
  boot_ridge_lmod <- cv.glmnet(
    x = Xboot, y = boot_df$nhtemp, alpha = 0, intercept = TRUE
  )
  resid <- boot_df$nhtemp - predict(boot_ridge_lmod, Xboot)
  results[i, , 'ridge'] <- predict(
    boot_ridge_lmod,
    newx = as.matrix(globwarm[, !(names(globwarm) %in% c('nhtemp', 'year'))])
  ) + sample(resid, size = N, replace = TRUE)

  noise <- numeric(n)
  for (j in 1:n) {
    yr <- boot_df$year[j]
    if (sum(boot_df$year == yr) > 1) {
      noise[j] <- rnorm(1, 0, .01)
    } else {
      noise[j] <- 0
    }
  }
  boot_df$year <- boot_df$year + noise
  boot_glmod <- gls(
    nhtemp ~ . - year, correlation = corAR1(form = ~ year), boot_df
  )
  results[i, , 'GLS'] <- predict(
    boot_glmod, globwarm[, !(names(globwarm) %in% c('nhtemp'))]
  ) + sample(residuals(boot_glmod), size = N, replace = TRUE)
}
```

95% confidence intervals are computed and plotted next.

```
ci_ridge <- apply(
  results[, , 'ridge'], 2, function(x){quantile(x, probs = c(.025, .975))}
)
ci_gls <- apply(
```

[4]This residuals sampling does not account for correlated residuals in the GLS model. (The residuals are transformed back to the original scale, hence their correlation.)

```
  results[, , 'GLS'], 2, function(x){quantile(x, probs = c(.025, .975))}
)
plot(globwarm$year, globwarm$nhtemp, ylim = c(min(ci_gls), max(df$nhtemp)))
lines(globwarm$year, ridge_lmod_pred, col = 2)
lines(globwarm$year, ci_ridge[1, ], col = 2, lty = 2)
lines(globwarm$year, ci_ridge[2, ], col = 2, lty = 2)
polygon(
  x = c(globwarm$year, rev(globwarm$year)),
  y = c(ci_ridge[1, ], rev(ci_ridge[2, ])),
  col = rgb(1, 0, 0, .2), border = NA
)
lines(globwarm$year, glmod_pred, col = 3, lty = 1)
lines(globwarm$year, ci_gls[1, ], col = 3, lty = 2)
lines(globwarm$year, ci_gls[2, ], col = 3, lty = 2)
polygon(
  x = c(globwarm$year, rev(globwarm$year)),
  y = c(ci_gls[1, ], rev(ci_gls[2, ])),
  col = rgb(0, 1, 0, .2), border = NA
)
legend('topleft', legend = c('ridge', 'GLS'), lty = 1, col = 2:3)
```



The resultant intervals are wide for both methods, suggesting the random error is the dominant source of uncertainty rather than sampling error, particularly for the ridge model. This can be verified by examining confidence intervals: the ridge CIs are narrower and asymmetric, while the PIs tend to be wider and symmetric. The profile of the PIs over time reflect the point predictions (solid lines) for the means.

Conclusion. Looking only at the GLS predictions, the periodic peaks and valleys suggest the recent warm temperatures *may* be consistent with temporal fluctuations, i.e. no global warming. However, closer examination of recent data show several more points warmer than the upper limit of the PI. (We'd expect approximately 95% of points to fall within the PIs.) These are data points the model

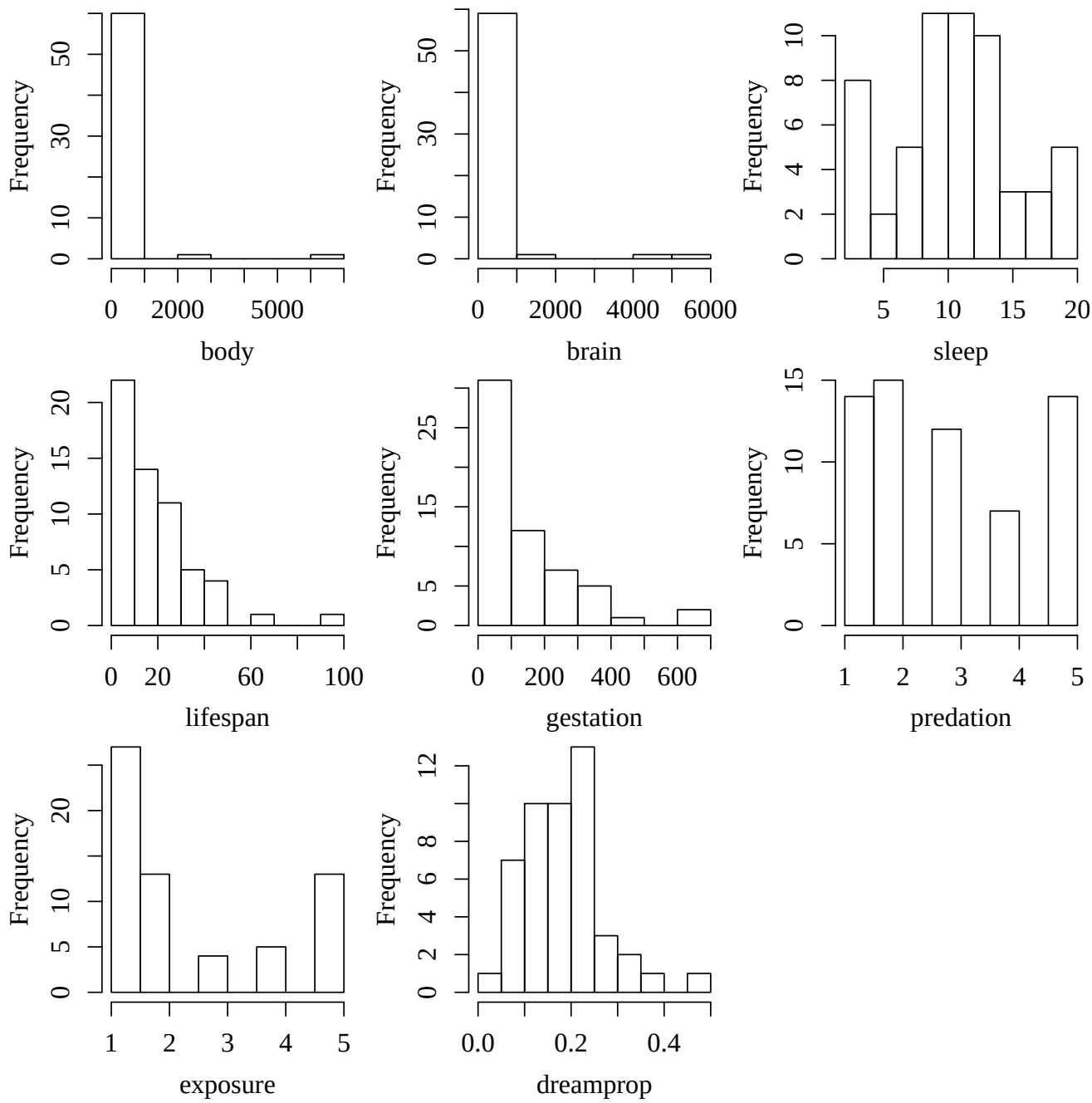
was trained on, so we'd expect fewer inconsistencies if anything. Further investigation on claims of global warming are warranted.

Exercise 3: Determining the effect of predation on mammal sleep.
Data examination. I'll start with `?mammalsleep` to learn about the data. Our response variable `sleep` is the number of hours per day a mammal sleeps. We are primarily interested in `predation`, an index taking discrete integers quantifying how likely a mammal is preyed upon: 1 being least likely and 5 being most likely. Since we want to extract the effect of `predation` in the presence of other predictors, I won't use `danger` because it's a function of `predation`. I'll also drop `nondream` and `dream` since mathematically they sum to `sleep`—we'd be able to build a model that predicts perfectly, but is completely useless for both future prediction and inference. To account for possible dream effects (for inference), I'll create a `dreamprop` variable measuring the proportion of dreaming sleep a mammal experiences.

```
df <- mammalsleep[, !(names(mammalsleep) %in% c('nondream', 'dream', 'danger'))]
df$dreamprop <- mammalsleep$dream / mammalsleep$sleep
```

A numerical summary via `summary(df)` shows a fair number of `NA`s, particularly `dreamrate`. The response `sleep` also contains a few missing values. I'll treat the missing data simply: I'll drop mammals with `NA sleep` and perform single imputation on missing predictors. A graphical summary of complete cases of each variable is shown next.

```
summary_plot(df)
```

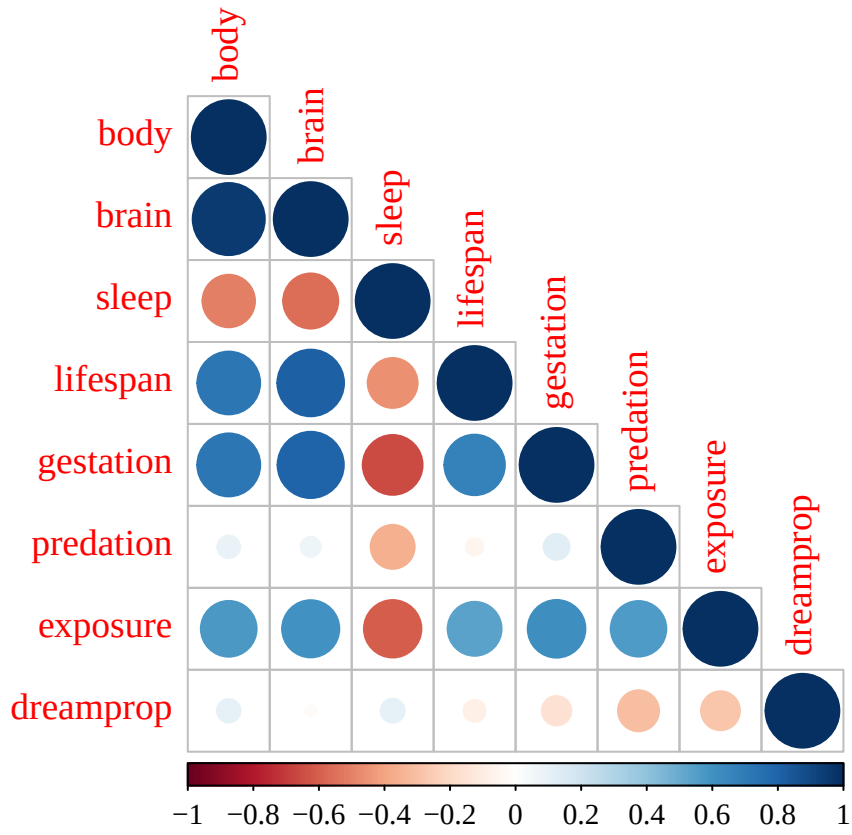


Two weight related and two (cumulative) time variables are highly skewed. While we don't usually need to make distributional assumptions on predictors, I'll (natural) log transform them so imputed estimates are more accurate (where predictors are temporarily the response variable). Furthermore, the final model's coefficients are then interpreted on a multiplicative, rather than additive, scale. This is arguably a more natural interpretation. For instance, $\beta_{\texttt{body}}$ no longer represents the change in y when `body` is increased by 1. Instead, it represents the change in y when `body` is increased by a factor of $e = 2.718\ldots$

```
vars_to_transf <- c('body', 'brain', 'lifespan', 'gestation')
for (var in vars_to_transf) {
  df[, var] <- log(df[, var])
}
```

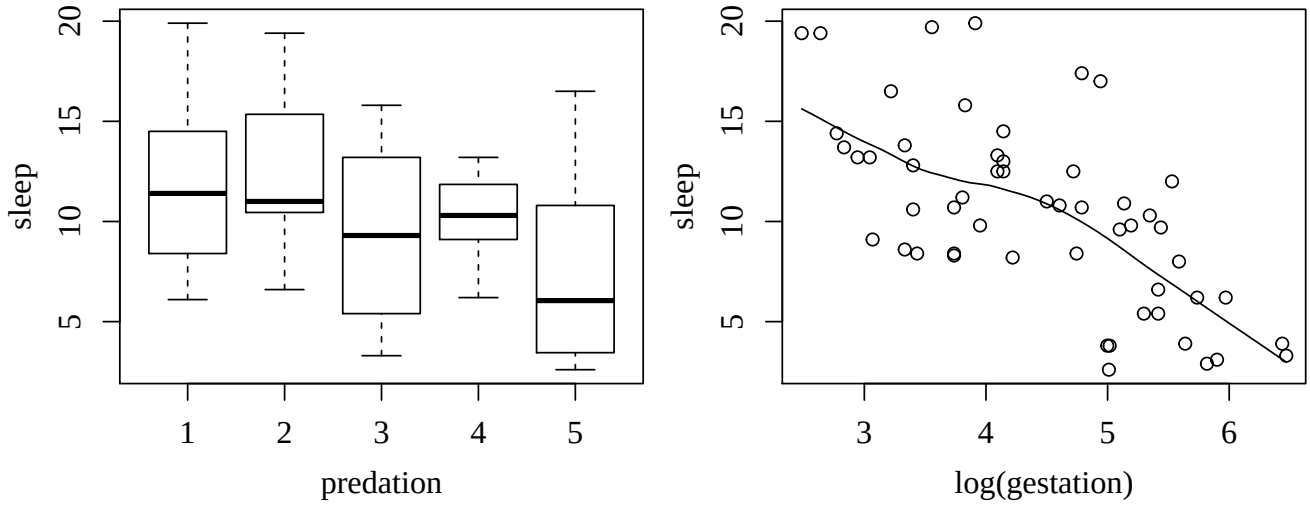
I'll quantify relations between variables using Spearman's rank correlation, which is invariant under log transformations.

```
correlations <- cor(df, use = 'pairwise.complete.obs', method = 'spearman')
corrplot(correlations, type = 'lower')
```



All the transformed variables are strongly correlated as we might expect since they're all some measure of size. These variables are also correlated with `exposure`; smaller mammals tend to sleep in protected dens. Due to these correlations among five variables, we may want to correct for collinearity. `sleep` has moderate (negative) correlations with size variables, `exposure`, and `predation`. `dreamprop` has the weakest correlations among variables. Plots between variables provide a more detailed picture; `plot(df)` generates all pairwise scatterplots. I'll show two representative plots including the response `sleep`: a boxplot against `predation`, the predictor of interest, and a scatterplot against `gestation`, the strongest correlation with `sleep`.

```
boxplot(sleep ~ predation, df)
scatter.smooth(df$gestation, df$sleep, xlab = 'log(gestation)', ylab = 'sleep')
```



The decreasing trend is not as clear in the boxplot, particularly for `predation`= 4, while the scatterplot appears roughly linear.

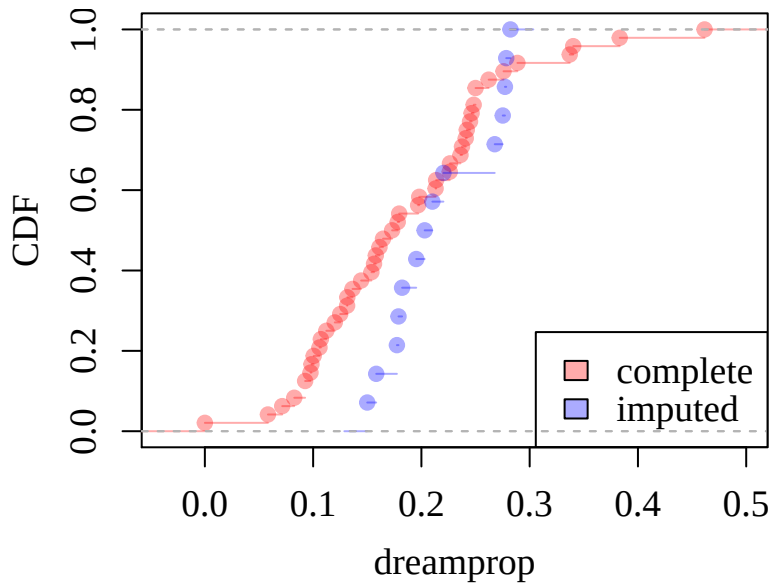
Data imputation. I'll impute missing predictors by fitting a simple regression against all other predictors, as we saw strong correlations in the data. Since several predictors have `NA` values, I'll iteratively impute starting with the predictor with the most missing data, `dreamprop` in this case. If more than one predictor is simultaneously `NA`, I'll fit a reduced model with all available predictors. These steps are implemented using my `impute()` function.

```
impute <- function(x, y, df, exclude = NULL) {
  "Regress predictor x (string) against all other variables in df (data frame)
  except the response y (string). Optionally specify exclude (vector of strings)
  to remove a set of variables from the imputation regression.

  Returns data frame with imputations made."
  form_str <- paste(x, '~ . -', y)
  if (length(exclude) > 0) {
    for (var in exclude) {
      form_str <- paste(form_str, '-', var)
    }
  }
  form <- as.formula(form_str)
  impute_mod <- lm(formula = form, data = df)
  df[is.na(df[, x]), x] <- predict(impute_mod, df[is.na(df[, x]), ])
  return(df)
}

df <- impute('dreamprop', 'sleep', df)
df <- impute('gestation', 'sleep', df)
df <- impute('lifespan', 'sleep', df)
df <- impute('lifespan', 'sleep', df, exclude = 'dreamprop')
df <- impute('lifespan', 'sleep', df, exclude = 'gestation')
df <- impute('gestation', 'sleep', df)
df <- impute('dreamprop', 'sleep', df)
```

I examined `df` after each step to determine the next most missing variable to impute. A fair amount of dream data was imputed: 22.6%. Because we don't want our results to be too sensitive

to the imputations, it's a good idea to examine the distribution of imputed values compared to complete cases. Since histogram binning is somewhat arbitrary, (empirical) cumulative distribution functions are better for comparison.

```
dreamprop_complete <- df$dreamprop[!is.na(mammalsleep$nondream)]
dreamprop_imputed <- df$dreamprop[is.na(mammalsleep$nondream)]
plot(
  ecdf(dreamprop_complete),
  col = rgb(1, 0, 0, 1/3), main = '', xlab ='dreamprop', ylab = 'CDF'
)
plot(ecdf(dreamprop_imputed), col = rgb(0, 0, 1, 1/3), add = T)
legend(
  'bottomright', legend = c('complete', 'imputed'),
  fill = c(rgb(1, 0, 0, 1/3), rgb(0, 0, 1, 1/3))
)
```



We see that imputed values seem to be regressed to the mean. This may be an apparent effect due to the small absolute number (14) of imputed values or it may be real for a number of reasons, e.g. those missing `dreamprop` are inherently different from complete cases. Still, the imputations are not drastically off and are an improvement over mean substitution (and throwing away data). Since imputation is not the focus of this chapter, I'll not investigate further. Finally, I won't impute any missing responses because the importance of the predictors could be inflated, resulting in overoptimistic estimates. For simplicity, I'll drop these 4 mammals in the final analysis. (These mammals were used to impute predictors.)

```
df <- df[!is.na(df$sleep), ]
summary(df)
```

```
##       body             brain            sleep          lifespan
##  Min.   :-5.30   Min.   :-1.97   Min.   : 2.60   Min.   :0.693
##  1st Qu.:-0.70   1st Qu.: 1.37   1st Qu.: 8.05   1st Qu.:1.755
##  Median : 1.01   Median : 2.63   Median :10.45   Median :2.553
##  Mean   : 1.14   Mean   : 3.02   Mean   :10.53   Mean   :2.509
##  3rd Qu.: 3.52   3rd Qu.: 4.99   3rd Qu.:13.20   3rd Qu.:3.296
##  Max.   : 8.80   Max.   : 8.65   Max.   :19.90   Max.   :4.605
##    gestation      predation       exposure      dreamprop
```

```
##  Min.   :2.48   Min.   :1.00   Min.   :1.00   Min.   :0.000
##  1st Qu.:3.47   1st Qu.:2.00   1st Qu.:1.00   1st Qu.:0.131
##  Median :4.28   Median :2.50   Median :2.00   Median :0.181
##  Mean   :4.38   Mean   :2.79   Mean   :2.31   Mean   :0.192
##  3rd Qu.:5.18   3rd Qu.:4.00   3rd Qu.:3.75   3rd Qu.:0.246
##  Max.   :6.47   Max.   :5.00   Max.   :5.00   Max.   :0.462
```

Model diagnostics and (re)selection. I'll start with a model linear in all predictors.

```
lmod <- lm(sleep ~ . , df)
sumary(lmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  23.5398     4.5959    5.12  4.9e-06
## body          0.2583     0.5972    0.43    0.667
## brain        -0.6347     0.8494   -0.75    0.458
## lifespan      0.0461     0.9316    0.05    0.961
## gestation    -1.7677     0.8275   -2.14    0.038
## predation    -0.9361     0.5434   -1.72    0.091
## exposure     -0.3571     0.6563   -0.54    0.589
## dreamprop    -1.6564     6.9198   -0.24    0.812
##
## n = 58, p = 8, Residual SE = 3.361, R-Squared = 0.53
```

`gestation` is the only statistically significant predictor at the $\alpha = 0.05$ level; it also has the largest coefficient. Before looking at graphical diagnostics, I'll inspect the condition numbers.

```
cond_num(lmod)
```

```
## [1]  1.0000  2.5129  5.2204 11.0156 14.3469 19.6596 85.0633
```

The variance inflation factors (VIFs)

```
vif(lmod)
```

```
##      body     brain  lifespan gestation predation  exposure dreamprop
##   17.2439   21.8911    4.0256    3.6097    3.2290    5.1261    1.7405
```

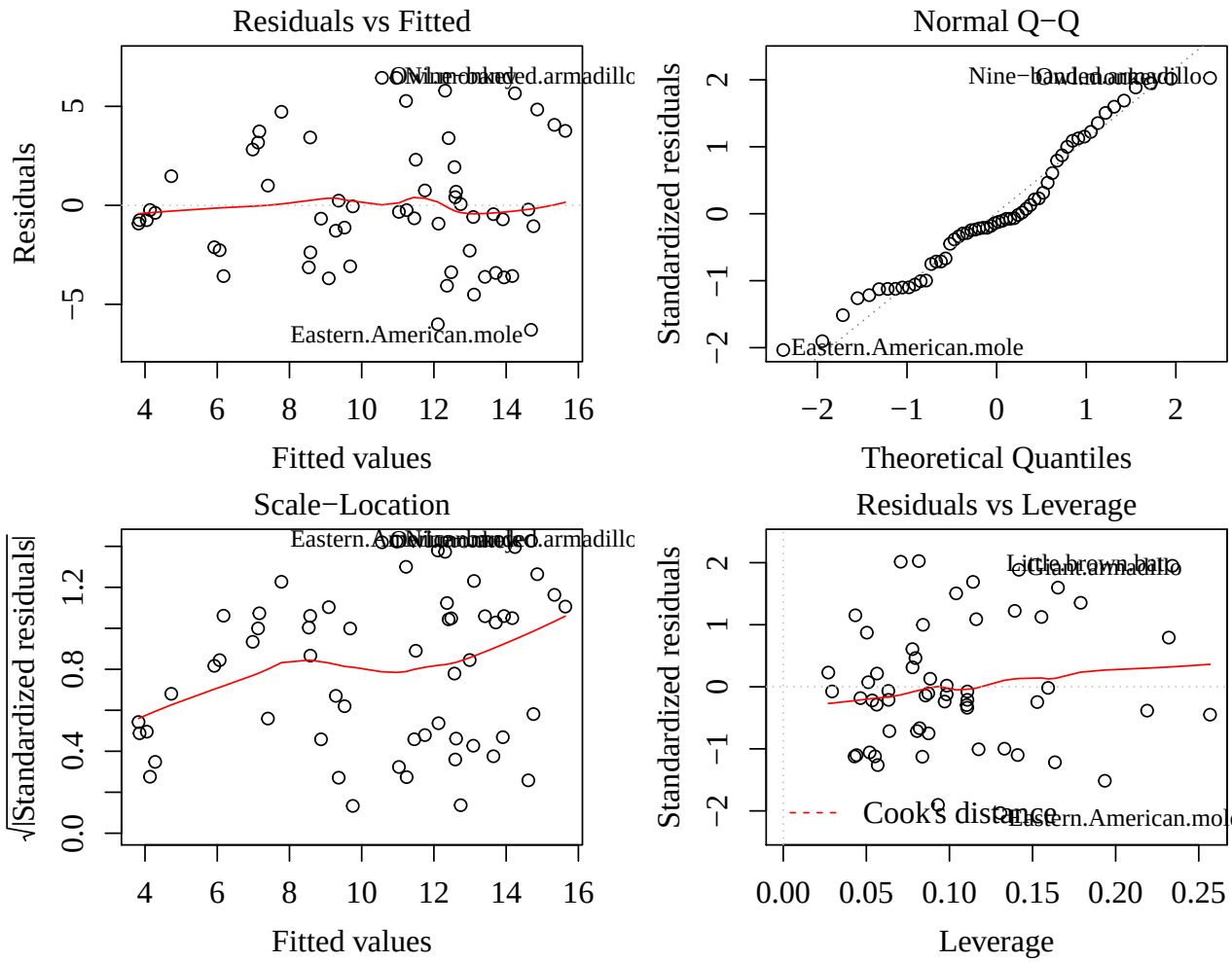
suggest `brain` is the most problematic. It turns out that removal of `brain` (and/or other size related variables) only mildly improves the condition numbers. The problematic variable there is `dreamprop`, which indeed has a larger standard error. Because `dreamprop` doesn't seem to have a substantial impact, I'll drop it as well, resulting in the following model.

```
lmod <- lm(sleep ~ body + lifespan + gestation + predation + exposure, df)
sumary(lmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   23.080      3.176    7.27  1.9e-09
## body          -0.157      0.242   -0.65   0.5194
## lifespan      -0.219      0.769   -0.28   0.7773
## gestation     -1.956      0.724   -2.70   0.0092
## predation     -0.933      0.503   -1.86   0.0692
## exposure      -0.279      0.635   -0.44   0.6618
##
## n = 58, p = 6, Residual SE = 3.315, R-Squared = 0.53
```

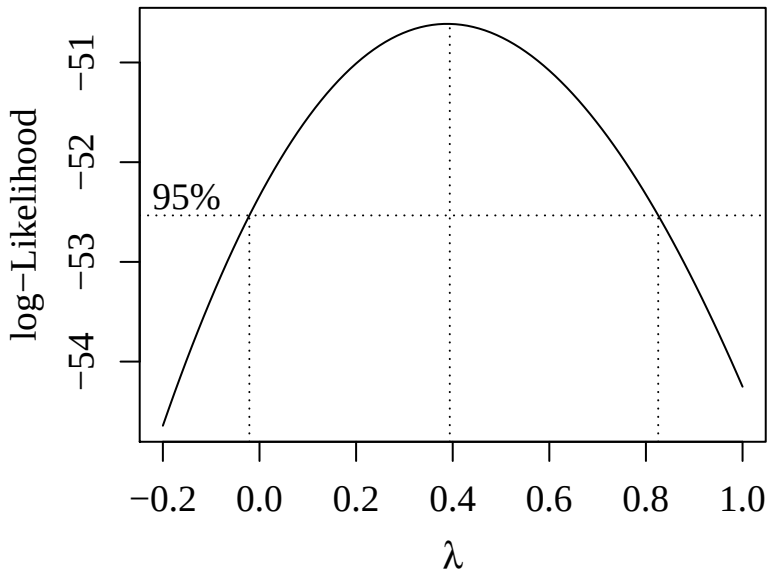
Despite removal of two variables, the R^2 is practically unchanged. The estimates are also less uncertain, though `gestation` is still the only significant predictor. The VIFs and condition numbers are also reasonable. Next up: visual diagnostics.

```
plot(lmod)
```



The qqplot and scale-location plot look problematic. We can try to address this with a Box–Cox transformation of the response.

```
library(MASS)  # For boxcox().
box <- boxcox(lmod, plotit = TRUE, lambda = seq(-.2, 1, length = 50), data = df)
```




```
(lambda <- box$x[which.max(box$y)])
```

```
## [1] 0.39394
```

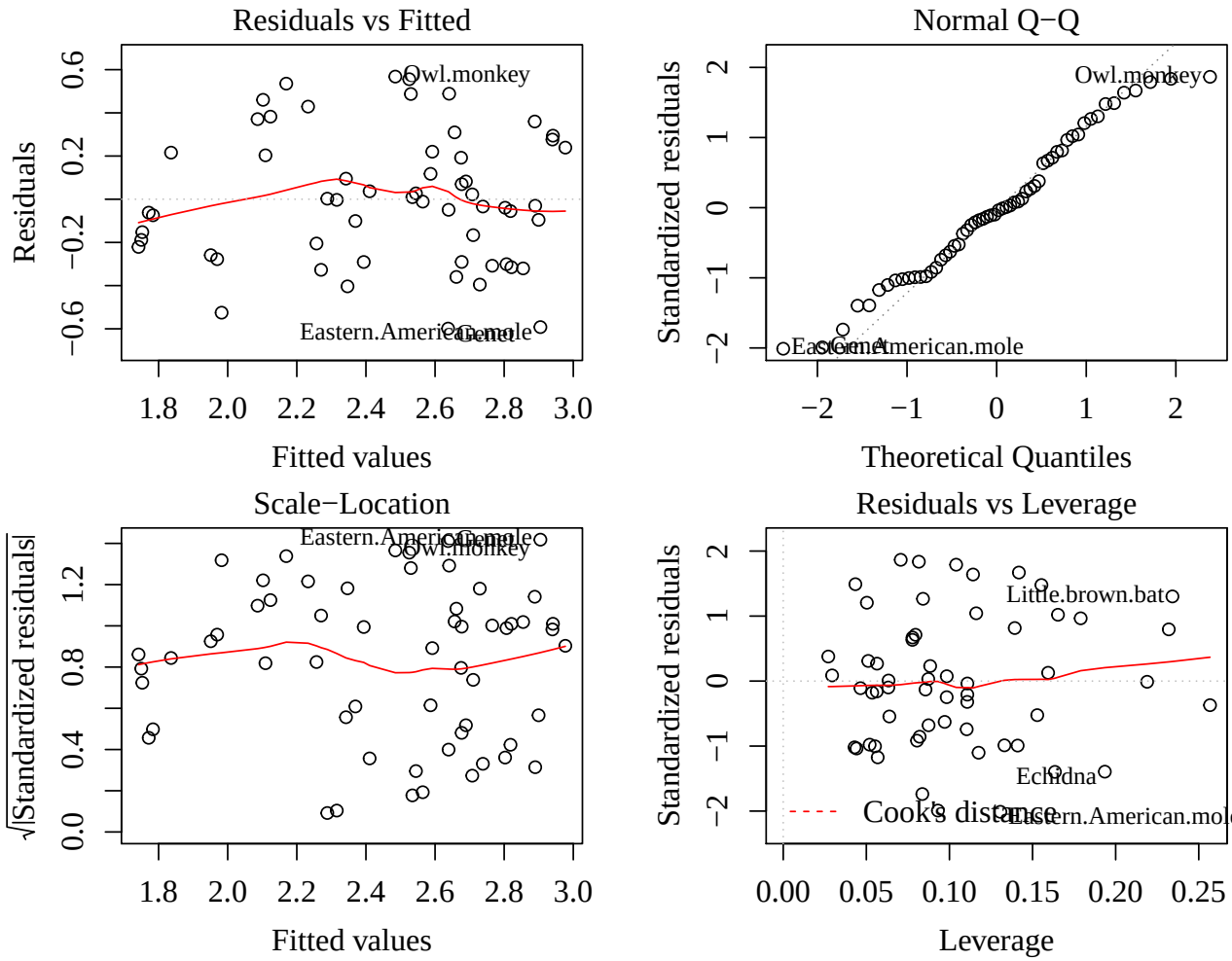
The likelihood suggests a transformation, as $\lambda = 1$ is outside the confidence interval. The optimal value is near $\lambda = 0.4$, which I'll use.

```
f <- function(x) {x**(lambda)}
boxmod <- lm(f(sleep) ~ body + lifespan + gestation + predation + exposure, df)
sumary(boxmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.7072     0.3026   12.25   <2e-16
## body         -0.0211     0.0230   -0.92   0.3631
## lifespan     -0.0136     0.0733   -0.19   0.8537
## gestation    -0.1880     0.0689   -2.73   0.0087
## predation    -0.0925     0.0479   -1.93   0.0590
## exposure     -0.0450     0.0605   -0.74   0.4605
##
## n = 58, p = 6, Residual SE = 0.316, R-Squared = 0.59
```

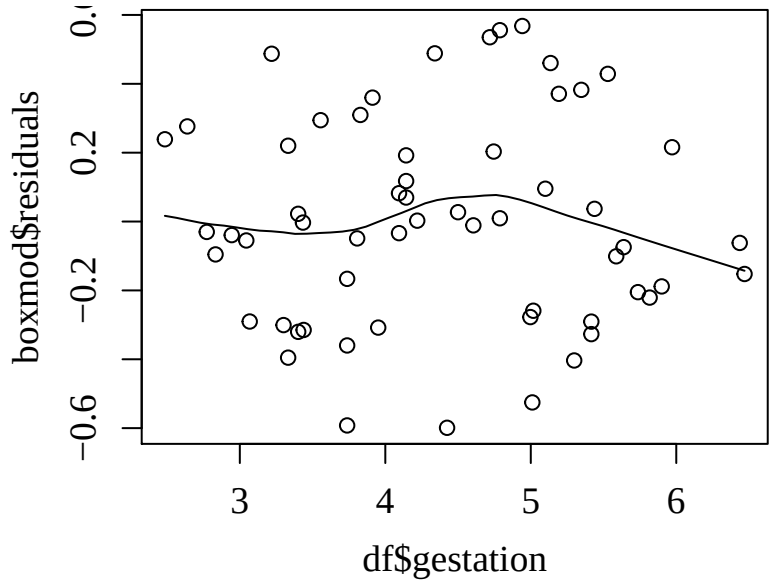
The p-values remain similar despite the change of scale. The qqplot and scale-location plot improve slightly, though now there is more variation in the residuals vs. fitted trendline.

```
plot(boxmod)
```



Examination of residual plots against predictors reveals that `gestation`, the most important predictor, displays a trend suggesting a linear `gestation` term is not sufficient.

```
scatter.smooth(df$gestation, boxmod$residuals)
```



I'll try to improve the fit using restricted cubic splines with four knots, requiring estimation of three `gestation` parameters.

```
library(splines)  # For ns().
spline_boxmod <- lm(
  f(sleep) ~ body + lifespan + ns(gestation, df = 3) + predation + exposure, df
)
sumary(spline_boxmod)
```

```
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)               3.3336     0.2550   13.07   <2e-16
## body                     -0.0175     0.0242   -0.72   0.4737
## lifespan                 -0.0158     0.0731   -0.22   0.8300
## ns(gestation, df = 3)1   -0.2441     0.1928   -1.27   0.2112
## ns(gestation, df = 3)2   -1.2396     0.4686   -2.65   0.0109
## ns(gestation, df = 3)3   -0.8417     0.2900   -2.90   0.0055
## predation                -0.0975     0.0485   -2.01   0.0501
## exposure                 -0.0381     0.0605   -0.63   0.5322
##
## n = 58, p = 8, Residual SE = 0.315, R-Squared = 0.6
```

The residuals vs. fitted plot and qqplot are improved. Residuals from the spline model also show little trend against `gestation`. Qualitatively, the model is similar to `boxmod`. We can test the overall significance of `gestation` using the general ANOVA framework, comparing to a nested model with no spline terms.

```
anova(lm(f(sleep) ~ body + lifespan + predation + exposure, df), spline_boxmod)
```

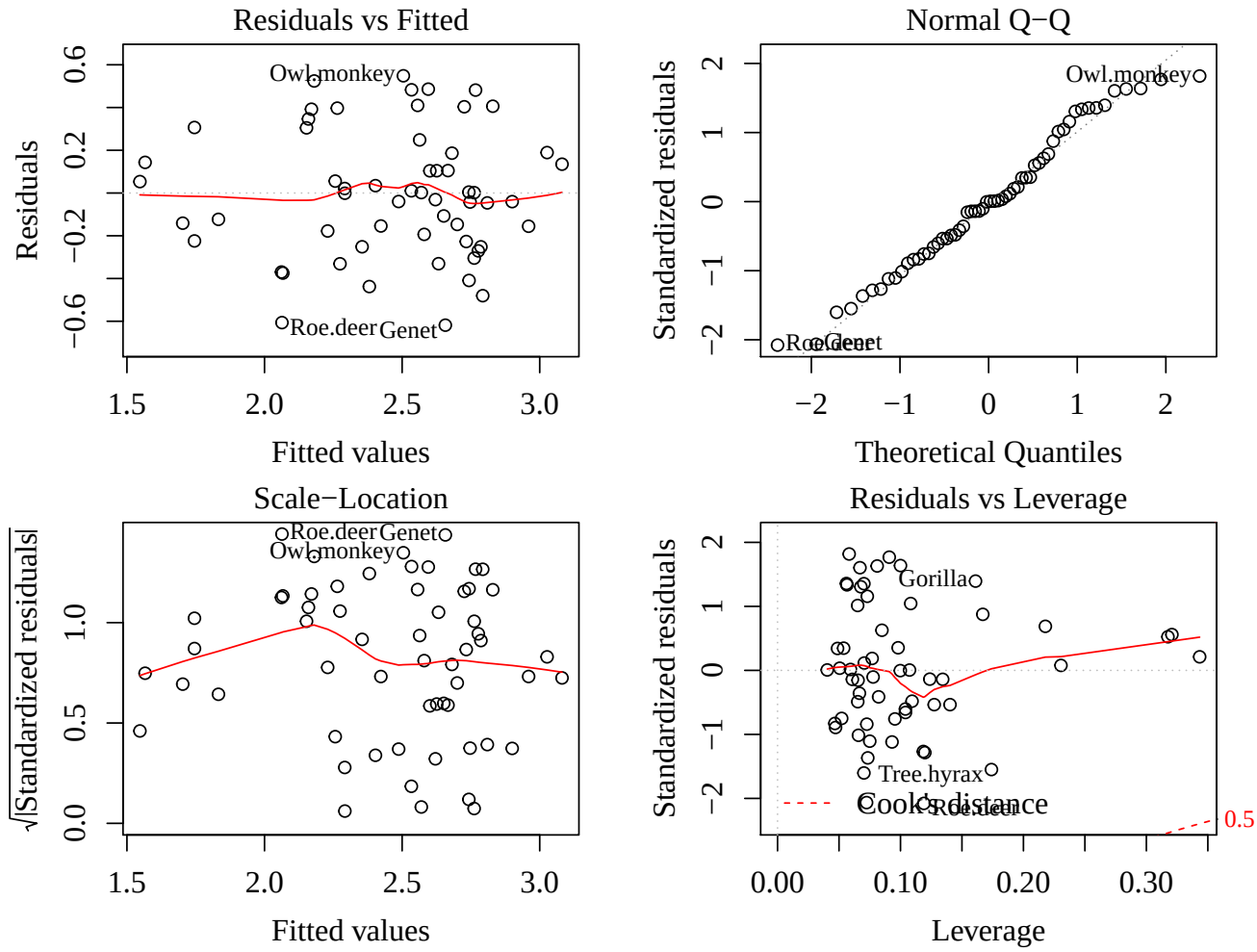
```
## Analysis of Variance Table
##
## Model 1: f(sleep) ~ body + lifespan + predation + exposure
## Model 2: f(sleep) ~ body + lifespan + ns(gestation, df = 3) + predation +
##     exposure
##   Res.Df  RSS Df Sum of Sq    F Pr(>F)
## 1     53 5.93
## 2     50 4.96  3      0.97 3.26  0.029 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The resultant p-value is still significant, but larger than before. Since I might be overfitting, I'll check if I should remove `body` or `lifespan` from the model, measuring performance using AIC. `predation` and `exposure` are fundamental in explaining `sleep`, so I'll keep those in all models. The resultant model with the lowest AIC is

```
reduced_spline_boxmod <- lm(
  f(sleep) ~ ns(gestation, df = 3) + predation + exposure, df
)
```

which improves the AIC by 2.53%. Diagnostic plots show that the model looks good.

```
plot(reduced_spline_boxmod)
```



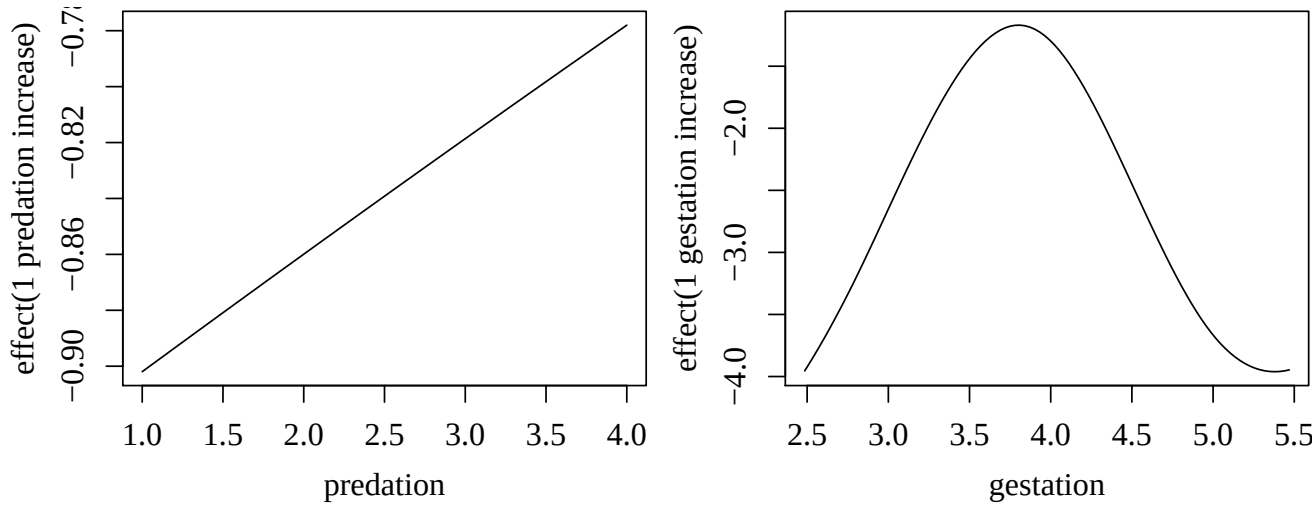
The model summary also shows no qualitative change. Indeed, `predation`'s p-value is still above the $\alpha = 0.05$ threshold.

```
sumary(reduced_spline_boxmod)
```

```
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                3.3031     0.1993   16.58  < 2e-16
## ns(gestation, df = 3)1    -0.2831     0.1833   -1.54  0.12857
## ns(gestation, df = 3)2    -1.3126     0.4528   -2.90  0.00548
## ns(gestation, df = 3)3    -0.9602     0.2494   -3.85  0.00032
## predation                 -0.0803     0.0404   -1.99  0.05231
## exposure                  -0.0614     0.0506   -1.21  0.23069
##
## n = 58, p = 6, Residual SE = 0.311, R-Squared = 0.6
```

Effect of predation on sleep. Equipped with our final model, we can begin to draw conclusions. While `predation` is not statistically significant, it would be erroneous to conclude it has no effect. We can understand its effect by plotting the change in `sleep` as `predation` changes for various values of `predation`, at some fixed values of other predictors (I'll use medians). Because of the Box–Cox transformation of the response, this effect is not a constant proportional to $\beta_{\texttt{predation}}$. Such an effect plot is implemented below, which has the added advantage of understanding the effect of nonlinear transformations, e.g. splines. Effect plots for `predation` and `gestation`, the most important variable, are shown.

```
effect_plot <- function(
  pred, lmod, df, delta = 1, n = 100, g = function(x){x}, add = FALSE
  ) {
  # Plot effect of predictor 'pred' on the response for a 'delta' change in x
  # for a range of x spaced by (max(x) - min(x) / 'n'), given other predictors
  # are at their median values.
  # User must supply lm object 'lmod' and data 'df'. If the response has been
  # transformed, specify the inverse transformation g so effects are on the
  # original scale. Overlay over existing plot by specifying 'add = TRUE'.

  # Compute effects for an otherwise median observation.
  median_obs <- as.data.frame(t(apply(df, 2, median)))
  xmin <- min(df[, pred])
  xmax <- max(df[, pred]) - delta
  xvec <- seq(xmin, xmax, length = n)
  effect <- numeric(length(xvec))
  for (i in 1:length(xvec)) {
    x <- xvec[i]
    obs1 <- obs2 <- median_obs
    obs1[, pred] <- x
    obs2[, pred] <- x + delta
    y1 <- g(predict(lmod, newdata = obs1))
    y2 <- g(predict(lmod, newdata = obs2))
    effect[i] <- y2 - y1
  }

  # Plot effects.
  if (add) {
    lines(xvec, effect, col = 2, lty = 2)
  } else {
    plot(
      xvec, effect, type = 'l',
      xlab = pred, ylab = paste0('effect(', delta, ' ', pred, ' increase)')
    )
  }
}

g <- function(x) {x**(1/lambda)}
effect_plot('predation', reduced_spline_boxmod, df, g = g)
effect_plot('gestation', reduced_spline_boxmod, df, g = g)
```



In general, as `predation` is increased, `sleep` is decreased. As mentioned previously, the effect is not uniform for all values of predation. For instance, the decrease in sleep as `predation` is changed from 1 to 2 is greater than the decrease in sleep as `predation` is changed from 4 to 5. This result is intuitive; presumably mammals with large predation are already taking the necessary precautions and/or sleeping less, so the effects of increasing predation are diminished. The effect of `gestation` is more complex, though the direction of effect is always negative.

Sensitivity analysis. To determine if the imputations significantly changed the results, I'll repeat the analysis on complete cases only. For simplicity, I'll use the structural form of the final model found in the imputed analysis. (A more rigorous sensitivity analysis would repeat all steps of my main analysis.)

```
is_cc <- (
  !is.na(mammalsleep$sleep)
  & !is.na(mammalsleep$gestation)
  & !is.na(mammalsleep$exposure)
)
df_cc <- mammalsleep[is_cc, ]
reduced_spline_boxmod_cc <- lm(
  f(sleep) ~ ns(gestation, df = 3) + predation + exposure, df_cc
)
sumary(reduced_spline_boxmod_cc)
```

```
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)               3.1335     0.1426   21.98   <2e-16
## ns(gestation, df = 3)1   -0.6471     0.2255   -2.87   0.0061
## ns(gestation, df = 3)2   -1.0316     0.3491   -2.96   0.0048
## ns(gestation, df = 3)3   -0.8665     0.2823   -3.07   0.0035
## predation                -0.0810     0.0416   -1.94   0.0577
## exposure                 -0.0582     0.0513   -1.13   0.2624
##
## n = 54, p = 6, Residual SE = 0.306, R-Squared = 0.62
```

The results are qualitatively similar to the main analysis which imputed 4 cases. The imputations did not lead to significant biases.

Chapter 13

Missing Data

13.1 Types of Missing Data

Examples of types of missingness. I'll illustrate MCAR, MAR, and MNAR in a simulated example with one continuous variable and one binary group variable. First, a model summary on the complete dataset.

```
library(faraway)  # For sumary().
set.seed(1)
n <- 100
x <- rnorm(n)
group <- ifelse(runif(n) > .5, 1, 0)
y <- x + group + rnorm(n)
df <- data.frame(y, x, group = as.factor(group))
sumary(lm(y ~ x + group, df))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.0679     0.1343    0.51     0.61
## x             0.8860     0.1179    7.52  2.8e-11
## group1        0.9139     0.2171    4.21  5.7e-05
##
## n = 100, p = 3, Residual SE = 1.054, R-Squared = 0.44
```

The estimates are good: they are unbiased with small relatively errors. Next, I'll simulate various incomplete datasets where, for simplicity, only x can be missing.
MCAR is relatively harmless and truly MCAR data can be thought of as just a reduction in sample size. The following randomly removes approximately 20% of data points.

```
df_mcar <- df[runif(n) < .8, ]
sumary(lm(y ~ x + group, df_mcar))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.0153     0.1413   -0.11  0.91394
## x             0.7539     0.1271    5.93  8.4e-08
## group1        0.9776     0.2416    4.05  0.00012
##
## n = 79, p = 3, Residual SE = 1.017, R-Squared = 0.41
```

The estimates are still good but have larger errors, as expected from the reduced sample size.
MAR is also salvageable because though certain groups may have more missing data, the available data within each group is representative and thus group differences can still be estimated. As an example, I'll randomly drop 40% of those in group 1.

```
df_mar <- df[ifelse(df$group == 1, runif(n) < .6, TRUE), ]
summary(lm(y ~ x + group, df_mar))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.0694     0.1318    0.53      0.6
## x             0.8712     0.1217    7.16  3.1e-10
## group1        1.0589     0.2528    4.19  7.0e-05
##
## n = 85, p = 3, Residual SE = 1.033, R-Squared = 0.44
```

Estimates are still good but more uncertain, particularly for $\beta_{\texttt{group}}$ because the groups are more unbalanced.
MNAR is problematic because data is missing in a systematic, but unknown way. The resulting data may no longer be representative and thus inferences can be biased and overoptimistic.

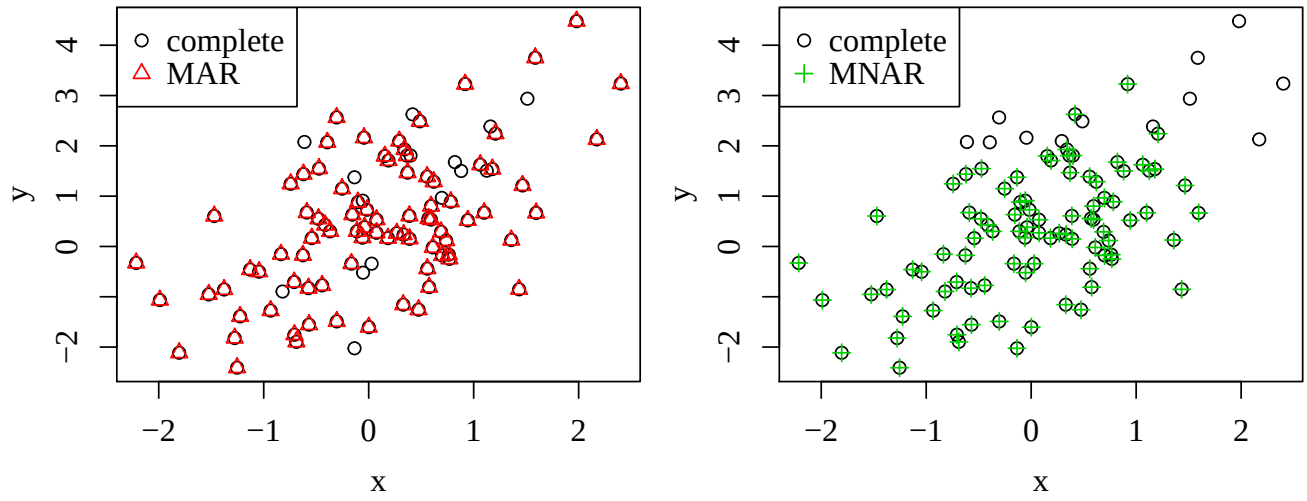
```
df_mnar <- df[ifelse(df$y > 2, runif(n) < .1, TRUE), ]
summary(lm(y ~ x + group, df_mnar))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.0429     0.1249   -0.34  0.73230
## x             0.7458     0.1233    6.05  3.8e-08
## group1        0.7361     0.2140    3.44  0.00091
##
## n = 88, p = 3, Residual SE = 0.951, R-Squared = 0.35
```

The estimates are biased *and* the errors are smaller. In theory, we could weight the existing large y cases, but in practice this requires outside knowledge to select the weights.

This simple example allows us to visualize the missingness by plotting y against x.

```
plot(y ~ x, df)
points(y ~ x, df_mar, col = 2, pch = 2)
legend('topleft', legend = c('complete', 'MAR'), col = 1:2, pch = 1:2)
plot(y ~ x, df)
points(y ~ x, df_mnar, col = 3, pch = 3)
legend('topleft', legend = c('complete', 'MNAR'), col = c(1, 3), pch = c(1, 3))
```



The data dropped in the MAR dataset are spread across x and y and are representative of the group 1 population. On the other hand, the data dropped in the MNAR dataset are clearly not representative and substantially change the fit.

13.2 Deletion

13.3 Single Imputation

13.4 Multiple Imputation

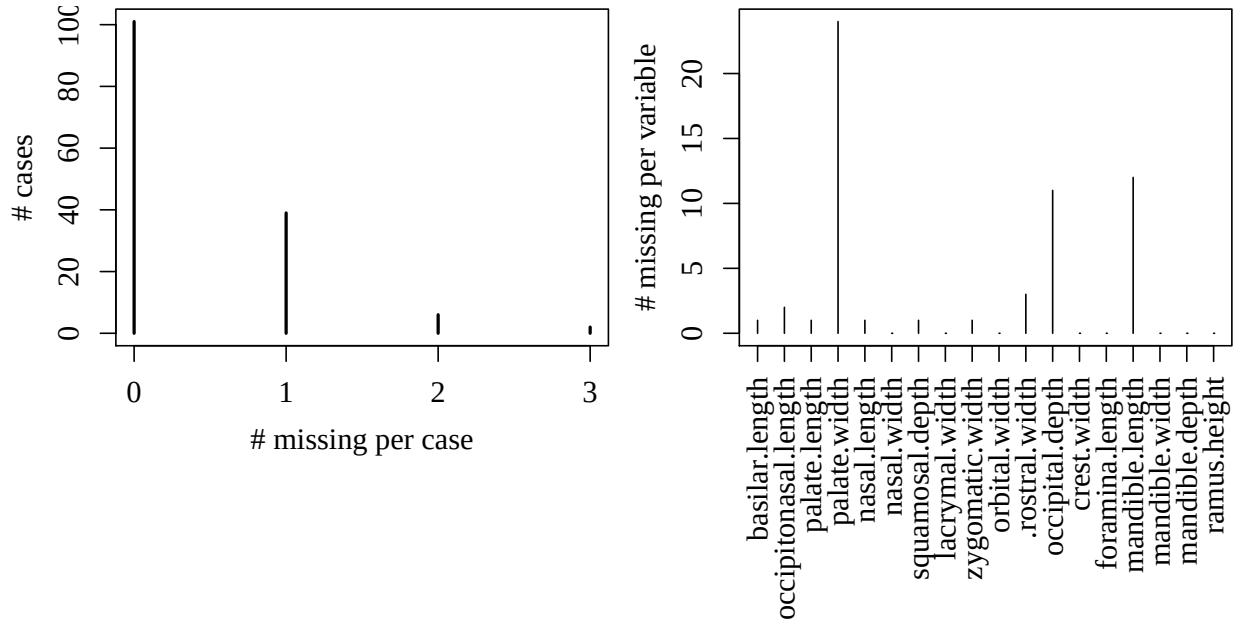
MI variance. The variance of the combined MI estimate $\widehat{\beta}_j = \frac{1}{m}\sum_i \widehat{\beta}_{ij}$ is not a straightforward calculation of $\text{var}\,\widehat{\beta}_j = \frac{1}{m^2}\,\text{var}\sum_i \widehat{\beta}_{ij}$. If it were, we could just increase the number of imputations m and artificially reduce the variance. The actual formula given in the text has two sources of variation: the usual sampling error and added variance from the imputation process $\text{var}\,\widehat{\beta}_j$.

Exercises

Exercise 1: Patterns of missingness in strongly correlated data.

(a) The distribution of the number of missing values per case and per variable are plotted below.

```
library(faraway)  # For data.
df <- kanga[, !(names(kanga) %in% c('species', 'sex'))]
plot(
  table(rowSums(is.na(df))), xlab = '# missing per case', ylab = '# cases'
)
plot(
  colSums(is.na(df)),
  type = 'h', xaxt = 'n', xlab = '', ylab = '# missing per variable'
)
axis(1, at = 1:ncol(df), labels = names(df), las = 2)
```



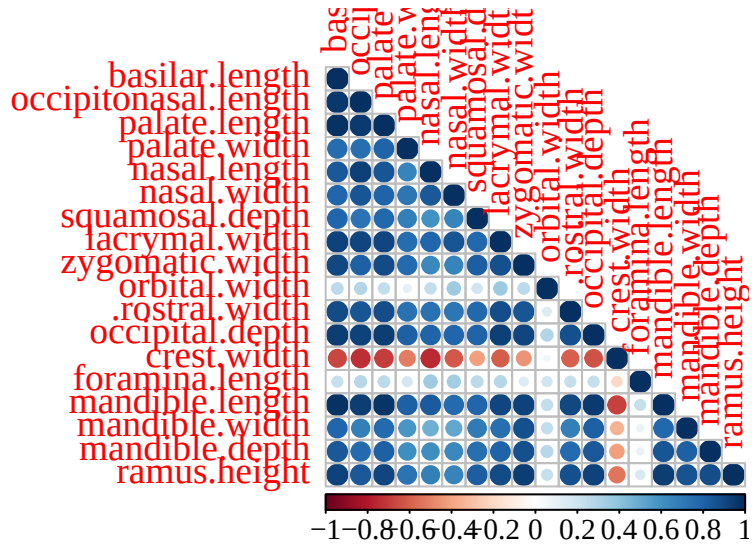
Most cases are only missing 1 variable. The most common missing variables are `palate.width`, `mandible.length`, and `occipital.depth`. These observations are reinforced with a `image(is.na(df))` plot.

(b) Because the most common missing variables exhibit many strong correlations with non-missing variables, as seen in a correlation plot,

```
library(corrplot)  # For corrplot().
```

```
## corrplot 0.84 loaded
```

```
correlations <- cor(df, use = 'pairwise.complete.obs', method = 'pearson')
corrplot(correlations, type = 'lower')
```



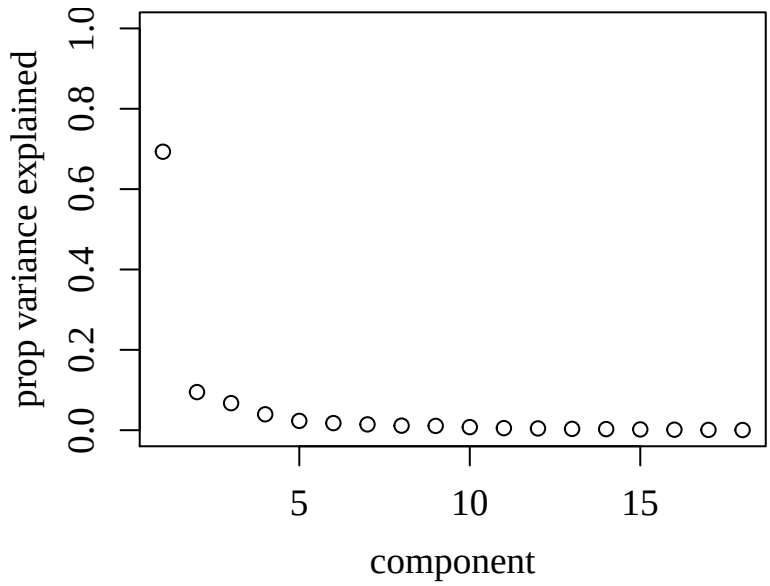
it is reasonable to simply drop them to preserve data.

```
drop <- c('palate.width', 'mandible.length', 'occipital.depth')
df_reduced <- df[, !(names(df) %in% drop)]
```

This reduces the number of incomplete cases from 31.76% to 6.08%.

(c) A scree plot of proportion of variance explained for each component is plotted for PCA on complete cases only.

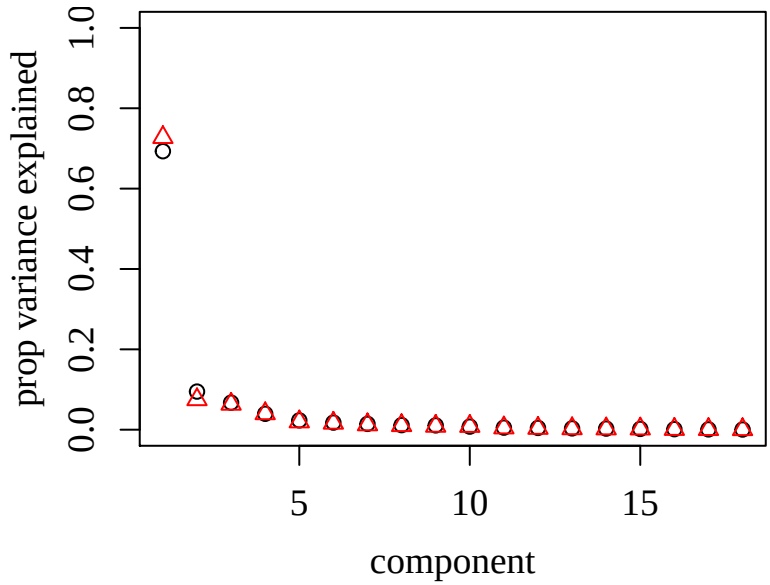
```
scree_plot <- function(pc) {
  plot(
    pc$sdev**2/sum(pc$sdev**2), type = 'p', ylim = c(0, 1),
    xlab = 'component', ylab = 'prop variance explained'
  )
}

df_complete <- df[rowSums(is.na(df)) == 0, ]
pc <- prcomp(df_complete, center = TRUE, scale. = TRUE)
scree_plot(pc)
```



The variance explained drop sharply after the first PC, and is negligible after the fifth PC.

(d) Repeating the previous question using MI and averaging the proportion of explained variance of all imputations,

```
library(Amelia)  # For amelia().
set.seed(1)
m = 25
df_mi <- amelia(df, m)
prop_var <- array(NA, dim = c(m, ncol(df)))
for (i in 1:m) {
  pc_mi <- prcomp(df_mi$imputations[[i]], center = TRUE, scale. = TRUE)
  prop_var[i, ] <- pc_mi$sdev**2 / sum(pc$sdev**2)
}
mean_prop_var <- apply(prop_var, 2, mean)
scree_plot(pc)
points(1:ncol(df), mean_prop_var, col=2, pch=2)
```



we observe nearly identical results except for PC1 and PC2.

Chapter 14

Categorical Predictors

```
library(faraway)  # For data and sumary().
```

14.1 A Two-Level Factor

Unequal variance t-test in the linear model framework. The text demonstrated equivalence between a two-sample equal variance t-test and a linear regression with a binary categorical variable. The equal variance assumption was necessary because in OLS, the residuals are assumed to be normally distributed with constant variance. Consequently, we check diagnostic plots that examine this homoscedasticity assumption. In the event the residuals don't have constant variance, we can correct for it using weights proportional to the inverse variance, i.e. WLS. Indeed WLS is equivalent to Welch's unequal variance t-test, provided we set the weights to the inverse variance of each group. The following simulated data demonstrates this.

```
n <- 100
x <- ifelse(runif(n) > .5, 1, 0)
y <- x + ifelse(x == 1, rnorm(n, 0, 1), rnorm(n, 0, 5))
df <- data.frame(y, x = factor(x))
w0 <- 1/sd(df[x == 0, ]$y)**2
w1 <- 1/sd(df[x == 1, ]$y)**2
lmod <- lm(y ~ x, df, weights = ifelse(x == 1, w1, w0))
sumary(lmod)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.622      0.786    0.79     0.43
## x1             0.391      0.797    0.49     0.62
##
## n = 100, p = 2, Residual SE = 1.000, R-Squared = 0
```

```
t.test(df[x == 1, ]$y, df[x == 0, ]$y, var.equal = FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  df[x == 1, ]$y and df[x == 0, ]$y
## t = 0.491, df = 41.1, p-value = 0.63
## alternative hypothesis: true difference in means is not equal to 0
```

```
## 95 percent confidence interval:
##  -1.2182  2.0008
## sample estimates:
## mean of x mean of y
##    1.0137    0.6224
```

The t-values are identical. There is a minor difference in the p-values because the Welch test estimates the degrees of freedom using the Welch–Satterthwaite equation. I'll compute it directly.

```
n0 <- sum(df$x == 0)
n1 <- sum(df$x == 1)
s0 <- sd(df[x==0, ]$y)
s1 <- sd(df[x==1, ]$y)
dof <- (s0**2/n0 + s1**2/n1)**2 / (s0**4/(n0**2*(n0-1)) + s1**4/(n1**2*(n1-1)))
t <- summary(lmod)$coefficients['x1', 't value']
(1-pt(t, dof))*2
```

```
## [1] 0.62606
```

While on the subject, two more equivalences are:

- one-sample t-test and linear regression with just an intercept.
- paired two-sample t-test and linear regression of the paired differences with just an intercept.

As an example of the latter:

```
y1 <- rnorm(n)
y2 <- y1 + rnorm(n)
sumary(lm(I(y1 - y2) ~ 1))
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.0734     0.0906    0.81     0.42
##
## n = 100, p = 1, Residual SE = 0.906, R-Squared = 0
```

```
t.test(y1, y2, paired = TRUE)
```

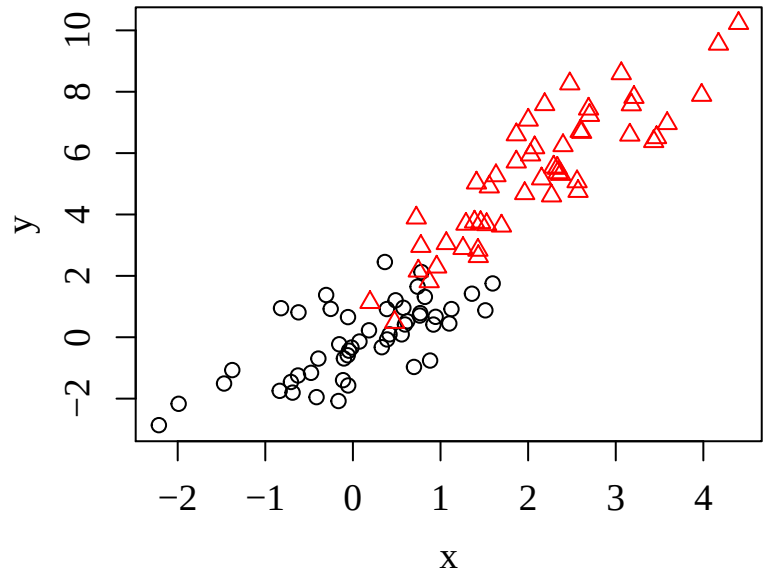
```
##
##  Paired t-test
##
## data:  y1 and y2
## t = 0.81, df = 99, p-value = 0.42
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.10629  0.25305
## sample estimates:
## mean of the differences
##                0.073382
```

14.2 Factors and Quantitative Predictors

Stratified regressions vs. single regression with interaction(s). Stratified regression, i.e. running separate linear regression for each subgroup, is another approach not mentioned

in the text. Consider the following data consisting of two groups; group 0 is generated by $Y = x + \varepsilon$ and group 1 is generated by $Y = 1 + 2x + \varepsilon$.

```
set.seed(1)
n <- 100
g <- c(rep(0, n/2), rep(1, n/2))
x <- c(rnorm(n/2, 0, 1), rnorm(n/2, 2, 1))
y <- c(x[1:(n/2)] + rnorm(n/2), 1 + 2*x[(n/2+1):n] + rnorm(n/2))
plot(y ~ x, pch = g+1, col = g+1)
```



Suppose we want to test if x has the same effect on y in both groups. One method, mentioned in the text, uses a regression model with an interaction term:

$$\mathbb{E}\, Y = \beta_0 + \beta_1 x + \beta_2 g + \beta_3 x g,$$

and tests $\beta_3 = 0$. Regression output is below.

```
lmod_int <- lm(y ~ g*x)
sumary(lmod_int)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.155      0.138   -1.13  0.26151
## g              1.285      0.358    3.59  0.00053
## x              1.028      0.166    6.20  1.4e-08
## g:x            0.947      0.219    4.33  3.6e-05
##
## n = 100, p = 4, Residual SE = 0.965, R-Squared = 0.91
```

Writing out the model for $g = 0$ and $g = 1$,

$$\mathbb{E}\, Y \mid g = 0 = \beta_0 + \beta_1 x, \qquad \mathbb{E}\, Y \mid g = 1 = (\beta_0 + \beta_2) + (\beta_1 + \beta_3) x,$$

we see that the regression model is essentially fitting two lines. This raises the question: why not run two separate regressions on each group?

```
lmod_strat0 <- lm(y ~ x, subset = (g == 0))
lmod_strat1 <- lm(y ~ x, subset = (g == 1))
sumary(lmod_strat0)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.155      0.129   -1.20     0.24
## x              1.028      0.156    6.59  3.2e-08
##
## n = 50, p = 2, Residual SE = 0.909, R-Squared = 0.47
```

```
sumary(lmod_strat1)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.130      0.349    3.23   0.0022
## x              1.975      0.150   13.15   <2e-16
##
## n = 50, p = 2, Residual SE = 1.019, R-Squared = 0.78
```

Indeed, we observe that the point estimates of the interaction and stratification approach are the same. However, the t- and p-values are not comparable because the variance of the error terms are no longer pooled. To test if $\beta_3 = 0$, we can take the difference of slopes and appeal to normality of the coefficient estimates. Denote the stratified slopes as γ_0 and γ_1 with standard errors σ_0 and σ_1. Assuming normality of residuals and independence, $\widehat{\gamma}_0 \sim \mathcal{N}(\gamma_0, \sigma_0^2)$ and $\widehat{\gamma}_1 \sim \mathcal{N}(\gamma_1, \sigma_1^2)$. Their difference $\widehat{\beta}_3 = \widehat{\gamma}_1 - \widehat{\gamma}_0 \sim \mathcal{N}(\beta_3, \sigma_0^2 + \sigma_1^2)$. Under the null $H_0 : \beta_3 = 0$, so test statistic $Z = \widehat{\beta}_3 / \sqrt{\sigma_0^2 + \sigma_1^2} \sim \mathcal{N}(0, 1)$. Replacing $\sigma \to \widehat{\sigma}$, $T = \widehat{\beta}_3 / \sqrt{\widehat{\sigma}_0^2 + \widehat{\sigma}_1^2} \sim \mathcal{T}(n_0 - 2 + n_1 - 2)$. This allows us to compute p-value

```
(1-pt((1.975-1.028)/(.156**2+.150**2)**.5, 96))*2
```

```
## [1] 3.0763e-05
```

which almost matches the interaction case.[1] A nonparametric permutation test could also be applied where we record $\widehat{\gamma}_1 - \widehat{\gamma}_0$ for a random sample of permutations to obtain the distribution of $\widehat{\beta}_3$.

While the example above showed exact agreement in point estimates (and close in p-values) between approaches, introducing other predictors in the model can yield different results. For instance, add a noise predictor u and refit models:

```
u <- rnorm(n)
lmod_int <- lm(y ~ g*x + u)
sumary(lmod_int)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.1571     0.1381   -1.14  0.25809
## g             1.2889     0.3597    3.58  0.00054
## x             1.0294     0.1665    6.18  1.6e-08
## u            -0.0526     0.0943   -0.56  0.57837
## g:x           0.9469     0.2194    4.32  3.9e-05
##
## n = 100, p = 5, Residual SE = 0.969, R-Squared = 0.91
```

```
lmod_strat0 <- lm(y ~ x + u, subset = (g == 0))
lmod_strat1 <- lm(y ~ x + u, subset = (g == 1))
```

[1] The agreement isn't exact because the stratified approach is more flexible by not pooling variance estimates, whereas the interaction model assumes homoscedasticity. The approaches could match exactly by weighting the interaction model by the inverse variance of each group.

```
sumary(lmod_strat0)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.152      0.130   -1.17     0.25
## x              1.027      0.157    6.56  3.8e-08
## u              0.105      0.119    0.88     0.38
##
## n = 50, p = 3, Residual SE = 0.911, R-Squared = 0.48
```

```
sumary(lmod_strat1)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.140      0.342    3.33   0.0017
## x              1.981      0.147   13.44   <2e-16
## u             -0.250      0.146   -1.71   0.0934
##
## n = 50, p = 3, Residual SE = 0.999, R-Squared = 0.8
```

The point estimates no longer agree exactly. This is because the stratified models incorporate an interaction between z and g. However, we can reconcile approaches by adding an interaction term zg in the interaction model:

```
lmod_int <- lm(y ~ g*x + g*u)
sumary(lmod_int)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.152      0.136   -1.11  0.26784
## g              1.292      0.355    3.64  0.00044
## x              1.027      0.164    6.25  1.2e-08
## u              0.105      0.125    0.84  0.40321
## g:x            0.954      0.216    4.41  2.8e-05
## g:u           -0.355      0.187   -1.89  0.06124
##
## n = 100, p = 6, Residual SE = 0.956, R-Squared = 0.92
```

We have seen that point estimates for stratified models and interaction models are the same when all predictors are included as interactions. However in some ways, the interaction model is more general because we can specify which interaction we are interested in, reducing the complexity of the model. While the stratified approach is more flexible with fewer assumptions, it consequently requires a larger sample size. Furthermore, the interaction model can be equivalent by including all pairwise interactions and weighting so that homoscedasticity within groups is not required. The interaction model approach also handles interactions between continuous predictors while stratifying bins somewhat arbitrarily. More discussion can be found in this SE thread.

14.3 Interpretation with Interaction Terms

Effect plot for interaction interpretation. Consider the example in the text, modeled

$$\texttt{Gas} = \beta_0 + \beta_1 \texttt{Temp} + \beta_2 \texttt{Insul} + \beta_3 \texttt{Temp} \times \texttt{Insul}\,.$$

```
library(MASS)  # For data.
lmod <- lm(Gas ~ Temp*Insul, whiteside)
sumary(lmod)
```

```
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       6.8538     0.1360   50.41  < 2e-16
## Temp             -0.3932     0.0225  -17.49  < 2e-16
## InsulAfter       -2.1300     0.1801  -11.83  2.3e-16
## Temp:InsulAfter   0.1153     0.0321    3.59  0.00073
##
## n = 56, p = 4, Residual SE = 0.323, R-Squared = 0.93
```

The interaction term is significant, but how to we interpret it? As mentioned in the text, we cannot interpret β_2 generally; after all, the effect of `Insul` depends on the value of `Temp`, a continuous variable. Mathematically, its effect is

$$\Delta \equiv \texttt{Gas}(\texttt{Insul} = 1) - \texttt{Gas}(\texttt{Insul} = 0) = \beta_2 + \beta_3 \texttt{Temp}\,.$$

Knowing this, we don't need to fit a `Temp` centered model. Indeed,

```
-2.12997 + .11530*mean(whiteside$Temp)
```

```
## [1] -1.5679
```

is the text's β_2 coefficient of the centered model. An effect plot of `Insul` as a function of `Temp` is just a line in this case. For more complex models, the effect may be a multivariable function of the predictor of interest and predictors tied to interactions. In this case, it may be best to plot multiple effect curves stratified by predictor values tied to interactions, e.g. the effect of a unit increase in x_2 for quartiles of x_1 resulting in four curves.

14.4 Factors With More Than Two Levels

Types of ANOVAs. The sequential ANOVA, sometimes called Type I ANOVA, compares models differently than the two-model comparisons `anova(lmod_reduced, lmod)` we've so far encountered. For instance, consider the first row of the sequential ANOVA table:

```
lmod <- lm(longevity ~ thorax*activity, fruitfly)
anova(lmod)
```

```
## Analysis of Variance Table
##
## Response: longevity
##                  Df Sum Sq Mean Sq F value  Pr(>F)
## thorax            1  15003   15003  130.73 < 2e-16 ***
## activity          4   9635    2409   20.99 5.5e-13 ***
## thorax:activity   4     24       6    0.05    0.99
## Residuals       114  13083     115
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

which tests if the addition of `thorax` to a intercept only model is statistically significant. Using the two-model comparison,

```
anova(lm(longevity ~ 1, fruitfly), lm(longevity ~ thorax, fruitfly))
```

```
## Analysis of Variance Table
##
## Model 1: longevity ~ 1
## Model 2: longevity ~ thorax
##   Res.Df   RSS Df Sum of Sq    F  Pr(>F)
## 1    123 37745
## 2    122 22742  1     15003 80.5 4.3e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

we observe a similar but notably different result. The additional explained sum of squares (SS) is identical, but the F-statistic and p-value differ. The difference is illuminated by the `?anova.lm` documentation.

> The table will contain F statistics (and P values) comparing the mean square for the row to the residual mean square.

In the sequential ANOVA, incremental improvements in SS are compared to the full model's residual SS with all possible terms included. The sequential testing of addition of `thorax` doesn't use the residual SS of the `thorax` model, but of the full `thorax*activity` model. The results of the sequential and two-model comparison are exactly the same when the full models are the same, i.e. the last (non residual) row of the sequential ANOVA.

```
anova(
  lm(longevity ~ thorax + activity, fruitfly),
  lm(longevity ~ thorax * activity, fruitfly)
)
```

```
## Analysis of Variance Table
##
## Model 1: longevity ~ thorax + activity
## Model 2: longevity ~ thorax * activity
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
## 1    118 13107
## 2    114 13083  4      24.3 0.05   0.99
```

See this SE thread for a more mathematical discussion.

14.5 Alternative Codings of Qualitative Predictors

Chapter 15

One Factor Models

```
library(faraway)  # For data and sumary().
```

15.1 The Model

15.2 An Example

15.3 Diagnostics

15.4 Pairwise Comparisons

General approach to construct confidence interval for difference in parameters. Consider a linear model with p parameters $\beta_0, \beta_1, \ldots \beta_{p-1}$. We want to construct a confidence interval for $\Delta_{ij} \equiv \widehat{\beta}_i - \widehat{\beta}_j$, $i \neq j$. Assuming normality of residuals, parameter estimates $\widehat{\beta}_i \sim \mathcal{N}(\beta_i, \boldsymbol{\Sigma}_{ii})$ where the variance-covariance matrix of the parameters is estimated as $\widehat{\boldsymbol{\Sigma}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}_{ii}\widehat{\sigma}^2$ where $\boldsymbol{X}$ is the $n \times p$ design matrix and $\widehat{\sigma}^2 = \mathrm{RSS}/(n-p)$ is an estimate of the residual variance with $n-p$ d.f. It follows that the difference of two parameter estimates is normally distributed with mean $\mathbb{E}\,\Delta_{ij} = \beta_i - \beta_j$ and variance $\operatorname{var}\Delta_{ij} = \boldsymbol{\Sigma}_{ii} + \boldsymbol{\Sigma}_{jj} - 2\boldsymbol{\Sigma}_{ij}$. We can then construct pivotal quantity $[\Delta_{ij} - (\beta_i - \beta_j)]/\operatorname{var}\Delta_{ij} \sim \mathcal{N}(0,1)$. However, since $\operatorname{var}\Delta_{ij}$ must be estimated by $\widehat{\operatorname{var}\Delta_{ij}}$, replacing $\boldsymbol{\Sigma} \to \widehat{\boldsymbol{\Sigma}}$, the pivotal distribution is $\mathcal{T}_{n-p}$. The resultant $(1-\alpha)$ interval is

$$\Delta_{ij} \pm t^{\alpha/2}_{n-p}\widehat{\operatorname{var}\Delta_{ij}} = (\widehat{\beta}_i - \widehat{\beta}_j) \pm t^{\alpha/2}_{n-p}(\widehat{\boldsymbol{\Sigma}}_{ii} + \widehat{\boldsymbol{\Sigma}}_{jj} - 2\widehat{\boldsymbol{\Sigma}}_{ij})\,.$$

This applies to any linear model and is simple to implement in R. For example, using the one-factor model in the text, we can construct the pairwise interval $\widehat{\beta}_D - \widehat{\beta}_B$, representing the difference between diet level D and B, as follows.

```
lmod <- lm(coag ~ diet, coagulation)
Delta = lmod$coefficients['dietD'] - lmod$coefficients['dietB']
Sigma <- vcov(lmod)
se <- sqrt(
  Sigma['dietB', 'dietB'] + Sigma['dietD', 'dietD'] - 2*Sigma['dietB', 'dietD']
)
```

```
alpha = .05
Delta + c(-1, 1) * qt(1-alpha/2, lmod$df.residual) * se
```

```
## [1] -7.6659 -2.3341
```

In the one-factor model, the estimated variance simplifies to $\widehat{\text{var}\,\Delta_{ij}} = \widehat{\sigma}^2(1/n_i + 1/n_j)$ where n_i is the number of observations with level i. The proof follows from direction calculation of $\widehat{\boldsymbol{\Sigma}}$. There are a total of p levels which can be coded with $p-1$ dummy variables. For notational convenience, let β_1 correspond to level 1, β_2 correspond to level 2, ..., β_{p-1} correspond to level $p-1$. We will use a coding with the p-th level as reference. The design matrix then takes the form

$$\boldsymbol{X} = \begin{bmatrix} 1 & I(1) & I(2) & \cdots & I(p-1) \\ 1 & I(1) & I(2) & \cdots & I(p-1) \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix},$$

where indicator variable $I(i) = 1$ if the observation is in level i and zero otherwise. It follows that

$$\boldsymbol{X}^T\boldsymbol{X} = \begin{bmatrix} n & n_1 & n_2 & \cdots & n_{p-1} \\ n_1 & n_2 & 0 & \cdots & 0 \\ n_2 & 0 & n_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ n_{p-1} & 0 & 0 & 0 & n_{p-1} \end{bmatrix},$$

a symmetric matrix with only nonzero diagonal and first row/column elements. Inverting and multiplying by $\widehat{\sigma}^2$ yields the covariance matrix

$$\widehat{\boldsymbol{\Sigma}} = \widehat{\sigma}^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} = \frac{\widehat{\sigma}^2}{n_p}\begin{bmatrix} 1 & -1 & -1 & \cdots & -1 \\ -1 & 1+\frac{n_p}{n_1} & 1 & \cdots & 1 \\ -1 & 1 & 1+\frac{n_p}{n_2} & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ -1 & 1 & 1 & 1 & 1+\frac{n_p}{n_{p-1}} \end{bmatrix}.$$

We then read off for any $i \neq j$, $i, j \in \{1, 2, \ldots, p-1\}$

$$\widehat{\text{var}\,\Delta_{ij}} = \widehat{\boldsymbol{\Sigma}}_{ii} + \widehat{\boldsymbol{\Sigma}}_{jj} - 2\widehat{\boldsymbol{\Sigma}}_{ij} = \widehat{\sigma}^2\left(\frac{1}{n_p} + \frac{1}{n_i} + \frac{1}{n_p} + \frac{1}{n_j} - \frac{2}{n_p}\right) = \widehat{\sigma}^2\left(\frac{1}{n_i} + \frac{1}{n_j}\right).$$

The result also holds for direct comparisons level $i = p$ with another level j, where $\widehat{\text{var}\,\Delta_{pj}}$ is just the variance $\widehat{\boldsymbol{\Sigma}}_{jj} = \widehat{\sigma}^2\,(1/n_p + 1/n_j)$. As a sanity check, one can show when $p = 2$, the result is the same as the usual two-sampled pooled t-test.[1] Another sanity check in R shows the standard error agrees with our previous result.

```
rse <- sqrt(sum(lmod$residuals**2) / lmod$df.residual)
se == rse*sqrt(1/sum(coagulation$diet == 'B') + 1/sum(coagulation$diet == 'D'))
```

```
## [1] TRUE
```

In more complex models, no analytic simplifications are possible but R doesn't really care—R can handle models with multiple factors and continuous predictors just fine as long as the model assumptions are reasonably met.

[1] The variance pooling is encoded in the regression constant variance assumption with the correct number of d.f. and (potential) imbalanced sample weighting.

More on Tukey's honest significant difference. In a one-factor model with k levels, we are comparing k group means and can use ANOVA to determine the overall significance of the factor. When comparing differences between pairs of groups, we must account for the multiple comparisons of $\binom{k}{2}$ possible differences. Consequently, conducting $\binom{k}{2}$ pairwise t-tests is unsatisfactory. Bonferroni corrections, while simple, are also usually too conservative; a nominal $(1-\alpha)$ interval would tend to have larger than $(1-\alpha)$ true coverage. Tukey's HSD extends the t-test to account for multiple comparisons. Its assumptions are similar:

- Denote the independent group means as $\overline{X}_1, \overline{X}_2, \ldots, \overline{X}_k$.
- Assume the groups have equal variance and that $\overline{X}_i$ are normally distributed.
- Under the null H_0 : all group means are equal, the group means $\overline{X}_i \sim \mathcal{N}(\mu, \sigma^2)$.

Tukey's HSD accounts for multiple comparisons by taking the largest difference among all $\binom{k}{2}$ pairs:

$$R = \max(\overline{X}_1, \overline{X}_2, \ldots, \overline{X}_k) - \min(\overline{X}_1, \overline{X}_2, \ldots, \overline{X}_k)\,.$$

Under H_0, we can quantify what the largest expected range R is, a quantity that encodes information from all pairwise comparisons. The corresponding confidence interval for the difference in means of groups i and j becomes

$$\left[\Delta_{ij} \pm t_{n-p}^{\alpha/2}\widehat{\text{var}\,\Delta_{ij}}\right] \longrightarrow \left[\Delta_{ij} \pm \tau_{n-p}^{\alpha/2}\widehat{\text{var}\,\Delta_{ij}}\right]$$

where $\tau_{n-p}^{\alpha/2}$ is a quantile of the Tukey range distribution. All pairwise intervals are extended by this factor, enabling simultaneous coverage. Let's compare compare standardized interval (half) widths for the t-test and the Tukey and Bonferroni correction using the text's example.

```
lmod <- lm(coag ~ diet, coagulation)
qt(1-alpha/2, lmod$df.residual)  # No adjustment.
```

```
## [1] 2.086
```

```
qtukey(0.95, 50, 24-4)/sqrt(2)  # Tukey.
```

```
## [1] 4.6501
```

```
qt(1-alpha/2/1225, lmod$df.residual)  # Bonferroni.
```

```
## [1] 5.2277
```

The Tukey increase is substantial, but not as much as Bonferroni. In the case of two groups, all methods are identical, for instance

```
qtukey(0.95, 2, 20)/sqrt(2)  # 2-group Tukey.
```

```
## [1] 2.086
```

15.5 False Discovery Rate

An overview of the advantages FDR has over traditional methods is discussed here.

Chapter 16

Models with Several Factors

16.1 Two Factors with No Replication

16.2 Two Factors with Replication

Fixed vs. random effects. Consider a one-factor study on student test scores across a sample of different schools in the country. We've so far only considered *fixed* effects models like ANOVA, which model the score of the i-th student from the j-th school as

$$Y_{ij} = \alpha_j + \varepsilon_{ij} \,.$$

α_j are the fixed effects we are interested in estimating, representing the mean scores of the j-th school. ε_{ij} are errors representing deviations in score of a student from their school's mean. The constant variance assumption $\operatorname{var} \varepsilon_{ij} = \sigma^2$ implies the variance of student scores is the same across all schools.

The fixed effects model also implicitly assumes schools across the country are so different their effects need to be estimated separately. In actuality, it is reasonable to assume that schools are similar enough that we can model them as follows:

- The mean of all test scores is μ.
- A specific school j has a random deviation from μ of size A_j with variance σ_A^2. One possible assumption is $A_j \sim \mathcal{N}(0, \sigma_A^2)$.
- A specific student i has a random deviation from their school's mean of size B_j with variance σ_B^2. One possible assumption is $B_{ij} \sim \mathcal{N}(0, \sigma_B^2)$.

The effects are now random variables A_j and B_{ij}, hence *random* effects model

$$Y_{ij} = \mu + A_j + B_{ij} \,.$$

B_{ij} is similar to ε_{ij} in the fixed effects model while estimation of school means A_j now incorporates distributional assumptions rather than just subgroup means α_j. This is particularly useful if some groups have small samples as information from other groups can be used in estimation.

The random effects model can also incorporate fixed effects, resulting in a *mixed* model. For instance, we could add a continuous predictor x representing average years of education of their parents:

$$Y_{ij} = \mu + \beta x + A_j + B_{ij} \,.$$

Even if we categorized x into high school or less, some college, or college degree and above, we could add it as a fixed effect

$$Y_{ijk} = \mu + \alpha_k + A_j + B_{ijk}$$

where k indexes one of the three categories. A fixed effect is justified because the effect of parental education is likely to be substantially different between groups. Consequently, deviations from the mean for parents with college degrees or above are likely to be greater than the mean of all cases, while the opposite holds true for high school or less educated parents. A random effect that is normally distributed doesn't really make sense.

While random (and mixed) effects models tend to use more information, when their assumptions don't hold fixed effects models may be better. In particular, random effects should be uncorrelated with other predictors. For example, if we added income to the model, we could run into issues where higher income students attend better schools, resulting in a tendency for $A_j > 0$ for those students. Similarly, there's a tendency for $A_j < 0$ for low income students. This violates the $A_j \sim \mathcal{N}(0, \sigma_A^2)$ assumption and can lead to biases.

16.3 Two Factors with an Interaction

Interpreting interactions. The text's "combined factor with number of levels equal to the product of the two factor levels" results in point estimates equivalent to predictions at all combinations of factor levels.

```
library(faraway)  # For data.
lmod <- lm(sqrt(breaks) ~ wool*tension, warpbreaks)
x <- data.frame(
  wool = c(rep('A', 3), rep('B', 3)),
  tension = rep(c('L', 'M', 'H'), 2)
)
pred <- predict(lmod, newdata = x)
names(pred) <- paste(x$wool, x$tension)
pred
```

```
##    A L    A M    A H    B L    B M    B H
## 6.5476 4.8259 4.8564 5.2381 5.2987 4.3022
```

These can be used in guiding recommendations. (Squaring the predictions puts them on the original scale.) If external factors like cost are an issue, pairwise comparisons can help guide further. For example, suppose wool A and medium tension are more economical. Is the wool B / high tension pairing significantly better than the wool A / medium tension pairing that warrants the higher cost?[1] The Tukey HSD intervals yields an adjusted p-value of 0.91 so if economical pairings are desired, we may be better off with wool A / medium tension.

16.4 Larger Factorial Experiments

[1] In the presence of interaction, it's important to compare wool/tension pairings together. Comparing wool A to B is misleading because of the interaction—the effect depends on the level of tension.

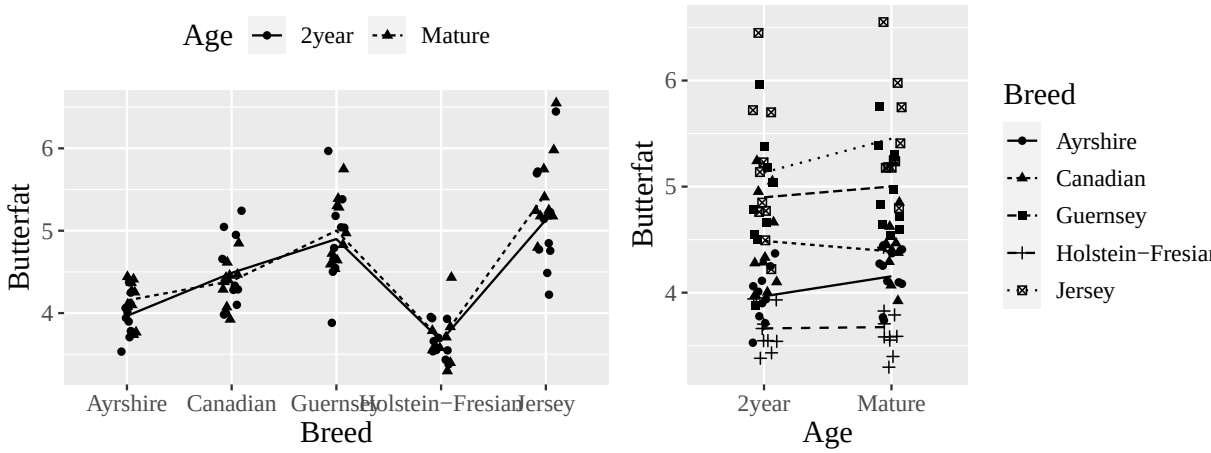
Exercises

```
library(faraway)  # For data and sumary().
```

Exercise 1: Optimizing the response in a two-factor study with potential interaction.

```
df <- butterfat
```

(a)
```
library(ggplot2)  # For ggplot() and associated functions.
library(gridExtra)  # For grid.arrange().
p1 <- ggplot(df, aes(x = Breed, y = Butterfat, group = Age)) +
  geom_point(mapping = aes(shape = Age), position = position_jitter(w=.1)) +
  stat_summary(fun = 'mean', geom = 'line', mapping = aes(linetype = Age)) +
  theme(legend.position = 'top')
p2 <- ggplot(df, aes(x = Age, y = Butterfat, group = Breed)) +
  geom_point(mapping = aes(shape=Breed), position = position_jitter(w=.1)) +
  stat_summary(fun = 'mean', geom = 'line', mapping = aes(linetype = Breed))
grid.arrange(p1, p2, ncol = 2)
```



`Butterfat` differs between cow breeds, but not so much between cow ages. The curves are roughly parallel, indicating little interaction. The variance of `Butterfat` within `Age` is similar but is slightly different within `Breed`. There is less variation within breeds overall.

(b)
```
lmod <- lm(Butterfat ~ Breed * Age, df)
anova(lmod)
```

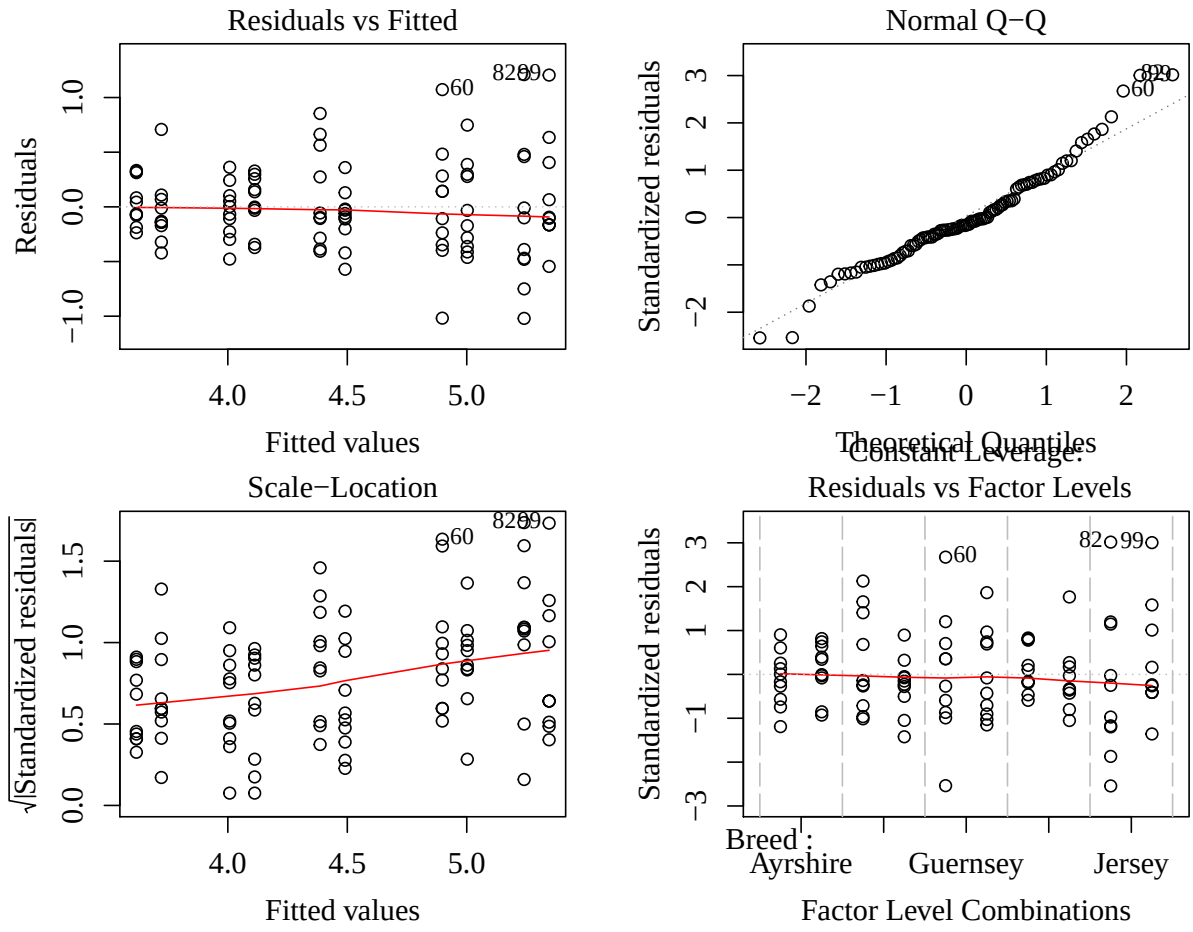
```
## Analysis of Variance Table
##
## Response: Butterfat
##           Df Sum Sq Mean Sq F value Pr(>F)
## Breed      4   34.3    8.58   49.57 <2e-16 ***
## Age        1    0.3    0.27    1.58   0.21
## Breed:Age  4    0.5    0.13    0.74   0.57
## Residuals 90   15.6    0.17
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA shows the interaction is not significant. This is not surprising from plots in (a).

(c) The ANOVA in (b) also shows that only `Breed` is significant. I'll drop the interaction term but keep `Age` since there is enough data to reasonably estimate both main effects.

```
lmod <- lm(Butterfat ~ Breed + Age, df)
```

(d)
```
plot(lmod)
```

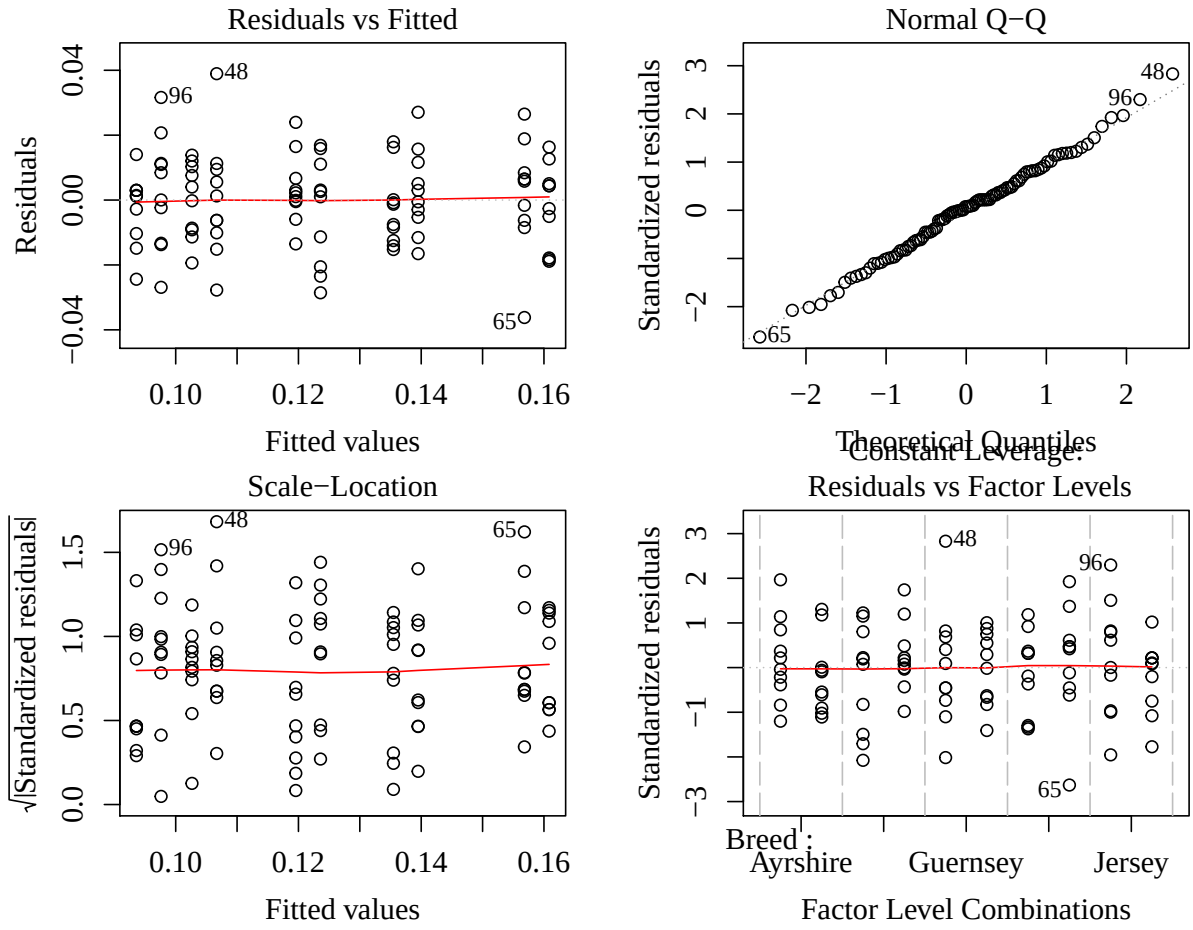


Diagnostic plots reveal slight deviations from normality and more seriously, a slight increase in variance for large fitted values. I'll try to address both with a Box–Cox transformation.

```
library(MASS)  # For boxcox().
box <- boxcox(lmod, lambda = seq(-3, 0, length = 20), plotit = F, data = df)
(lambda <- box$x[which.max(box$y)])
```

```
## [1] -1.4211
```

```
f <- function(y) {y**lambda}
boxmod <- lm(f(Butterfat) ~ Breed + Age, df)
plot(boxmod)
```



The updated diagnostics look excellent, though we must be careful with interpretation. High values of `Butterfat` y now correspond to small $f(y)$.

(e)
```
sumary(boxmod)
```

```
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               0.13950    0.00348   40.12  < 2e-16
## BreedCanadian            -0.01591    0.00449   -3.54  0.00062
## BreedGuernsey            -0.03282    0.00449   -7.31  8.6e-11
## BreedHolstein-Fresian     0.02134    0.00449    4.75  7.2e-06
## BreedJersey              -0.04188    0.00449   -9.33  4.9e-15
## AgeMature                -0.00403    0.00284   -1.42  0.15947
##
## n = 100, p = 6, Residual SE = 0.014, R-Squared = 0.73
```

The model summary tells us mature Jersey cows have the highest butter fat content milk. The next best would be young Jersey cows, but we can read off from the summary that it's not significantly different. However since age is the smallest effect, we may wonder what the next best breed would be. To adjust for multiple comparisons, I'll use Tukey HSD intervals.

```
TukeyHSD(aov(boxmod))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = boxmod)
```

```
##
## $Breed
##                                 diff        lwr        upr   p adj
## Canadian-Ayrshire         -0.0159090 -0.0283947 -0.0034234 0.00545
## Guernsey-Ayrshire         -0.0328244 -0.0453101 -0.0203388 0.00000
## Holstein-Fresian-Ayrshire  0.0213421  0.0088565  0.0338277 0.00007
## Jersey-Ayrshire           -0.0418813 -0.0543669 -0.0293956 0.00000
## Guernsey-Canadian         -0.0169154 -0.0294010 -0.0044297 0.00260
## Holstein-Fresian-Canadian  0.0372511  0.0247655  0.0497368 0.00000
## Jersey-Canadian           -0.0259722 -0.0384579 -0.0134866 0.00000
## Holstein-Fresian-Guernsey  0.0541665  0.0416809  0.0666522 0.00000
## Jersey-Guernsey           -0.0090568 -0.0215425  0.0034288 0.26571
## Jersey-Holstein-Fresian   -0.0632234 -0.0757090 -0.0507377 0.00000
##
## $Age
##                      diff       lwr       upr   p adj
## Mature-2year -0.0040261 -0.009663 0.0016109 0.15947
```

We see that Jersey cows are not clearly superior than Guernsey cows, though they are statistically significantly better compared to all other cow breeds.

Sensitivity analysis. Are these conclusions dependent on model? What if we kept the interaction or dropped `Age`? The answer is no—the conclusions are the same. Relevant output is displayed for a `Breed` only model and a two-factor with interaction model.

```
lmod <- lm(f(Butterfat) ~ Breed, df)
TukeyHSD(aov(lmod))$Breed['Jersey-Guernsey', ]
```

```
##       diff        lwr        upr      p adj
## -0.0090568 -0.0216062  0.0034925  0.2706666
```

```
lmod <- lm(f(Butterfat) ~ Breed*Age, df)
TukeyHSD(aov(lmod))[['Breed:Age']]['Jersey:Mature-Guernsey:Mature', ]
```

```
##       diff        lwr        upr      p adj
## -0.0116326 -0.0323589  0.0090937  0.7203322
```

The larger p-value in the interaction model despite a larger effect is due to correcting for more comparisons.

Exercise 6: Five way ANOVA with potential interactions.

```
df <- hsb
```

(a)
```
lmod <- lm(
  math ~ gender + race + ses + schtyp + prog + gender:race + gender:ses
    + gender:schtyp + gender:prog + race:ses + race:schtyp + race:prog
    + ses:schtyp + ses:prog + schtyp:prog,
  data = df
)
```

Accounting for all second-order effects results in $\binom{5}{1} + \binom{5}{2} = 15$ formula terms in the model and a total of $1 + \sum_{i=1}^{5} n_i + \sum_{i=1}^{5} \sum_{j=i+1}^{5} n_i n_j = 41$ total parameters, where n_i is one less than the number of distinct levels in the five factors. The following calculation verifies the

above formula, which can be checked with the model summary.

```
n <- c(
  length(levels(df$gender)) - 1,
  length(levels(df$race)) - 1,
  length(levels(df$ses)) - 1,
  length(levels(df$schtyp)) - 1,
  length(levels(df$prog)) - 1
)
s <- 1
for (i in 1:length(n)) {
  s <- s + n[i]
  for (j in (i+1):length(n)) {
    if (i < j & j <= 5) s <- s + n[i] * n[j]
  }
}
s
```

```
## [1] 41
```

(b)

```
anova(lmod)
```

```
## Analysis of Variance Table
##
## Response: math
##                Df Sum Sq Mean Sq F value  Pr(>F)
## gender          1     15      15    0.24  0.6243
## race            3   1849     616    9.87 5.2e-06 ***
## ses             2    744     372    5.96  0.0032 **
## schtyp          1     55      55    0.89  0.3473
## prog            2   2946    1473   23.59 1.1e-09 ***
## gender:race     3    113      38    0.61  0.6125
## gender:ses      2    303     152    2.43  0.0913 .
## gender:schtyp   1      1       1    0.01  0.9148
## gender:prog     2      5       3    0.04  0.9598
## race:ses        6    192      32    0.51  0.7982
## race:schtyp     3     95      32    0.51  0.6763
## race:prog       6    770     128    2.06  0.0614 .
## ses:schtyp      2    140      70    1.12  0.3282
## ses:prog        4    177      44    0.71  0.5867
## schtyp:prog     2    134      67    1.07  0.3439
## Residuals     159   9926      62
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`anova()` finds no significant interactions at the $\alpha = 0.05$ level.

(c)

```
drop1(lmod, test = 'F')
```

```
## Single term deletions
##
```

```
## Model:
## math ~ gender + race + ses + schtyp + prog + gender:race + gender:ses +
##     gender:schtyp + gender:prog + race:ses + race:schtyp + race:prog +
##     ses:schtyp + ses:prog + schtyp:prog
##               Df Sum of Sq   RSS AIC F value Pr(>F)
## <none>                     9926 863
## gender:race    3        95 10021 859    0.51  0.678
## gender:ses     2       241 10166 864    1.93  0.149
## gender:schtyp  1         3  9929 861    0.05  0.817
## gender:prog    2         6  9931 859    0.05  0.956
## race:ses       6       472 10398 860    1.26  0.278
## race:schtyp    3        31  9957 858    0.17  0.920
## race:prog      6       861 10786 868    2.30  0.037 *
## ses:schtyp     2       101 10027 861    0.81  0.447
## ses:prog       4       117 10042 857    0.47  0.760
## schtyp:prog    2       134 10060 862    1.07  0.344
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`drop1()` finds one significant `race:prog` interaction. The main difference is that `anova(lmod)` tests terms in sequential order, so the `race:prog` interaction is tested against a model without `ses:schtyp`, `ses:prog`, and `schtyp:prog`, i.e.

```
anova(
  update(lmod, . ~ . - schtyp:prog - ses:prog - ses:schtyp - race:prog),
  update(lmod, . ~ . - schtyp:prog - ses:prog - ses:schtyp)
)
```

```
## Analysis of Variance Table
##
## Model 1: math ~ gender + race + ses + schtyp + prog + gender:race + gender:ses
##     gender:schtyp + gender:prog + race:ses + race:schtyp
## Model 2: math ~ gender + race + ses + schtyp + prog + gender:race + gender:ses
##     gender:schtyp + gender:prog + race:ses + race:schtyp + race:prog
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
## 1    173 11147
## 2    167 10377  6       770 2.06   0.06 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`drop1()` instead compares the full model with all interactions to the model missing `race:prog`, i.e.

```
anova(update(lmod, . ~ . - race:prog), lmod)
```

```
## Analysis of Variance Table
##
## Model 1: math ~ gender + race + ses + schtyp + prog + gender:race + gender:ses
##     gender:schtyp + gender:prog + race:ses + race:schtyp + ses:schtyp +
##     ses:prog + schtyp:prog
## Model 2: math ~ gender + race + ses + schtyp + prog + gender:race + gender:ses
##     gender:schtyp + gender:prog + race:ses + race:schtyp + race:prog +
```

```
##     ses:schtyp + ses:prog + schtyp:prog
##   Res.Df   RSS Df Sum of Sq   F Pr(>F)
## 1    165 10786
## 2    159  9926  6       861 2.3  0.037 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For this reason, `drop1()` is better for our purposes. A smaller difference is that the `anova()` acting on a `lm` object makes comparisons to the full model in the `lm` object at all stages, hence why the previous two-model comparison ANOVA's p-value doesn't match `anova(lmod)` exactly despite sums of squares matching. More on this distinction was discussed in Section 14.4.

(d) I'll measure if a main effects model only is preferred in two ways: the usual two-model ANOVA comparison and AIC.

```
mainmod <- lm(math ~ gender + race + ses + schtyp + prog, df)
anova(mainmod, lmod)
```

```
## Analysis of Variance Table
##
## Model 1: math ~ gender + race + ses + schtyp + prog
## Model 2: math ~ gender + race + ses + schtyp + prog + gender:race + gender:ses
##     gender:schtyp + gender:prog + race:ses + race:schtyp + race:prog +
##     ses:schtyp + ses:prog + schtyp:prog
##   Res.Df   RSS Df Sum of Sq  F Pr(>F)
## 1    190 11857
## 2    159  9926 31      1931  1   0.48
```

```
paste(AIC(mainmod), AIC(lmod))
```

```
## [1] "1406.04487835489 1432.48824050208"
```

Both measurements indicate the main effects model is preferred. The formal test of testing if all interaction coefficients is zero is not significant, and the AIC suggests the additional d.f. associated with extra coefficients hinders the model. A simultaneous pairwise test within each category can be mad with Tukey HSD intervals. For instance,

```
TukeyHSD(aov(mainmod))$prog
```

```
##                       diff       lwr     upr       p adj
## general-academic  -5.6698  -8.9947 -2.3449 2.3874e-04
## vocation-academic -8.2488 -11.4552 -5.0425 1.9656e-08
## vocation-general  -2.5790  -6.4135  1.2554 2.5294e-01
```

we observe academic programs significantly outperform vocation and general programs. There is no significant difference between vocation and general programs.

(e)

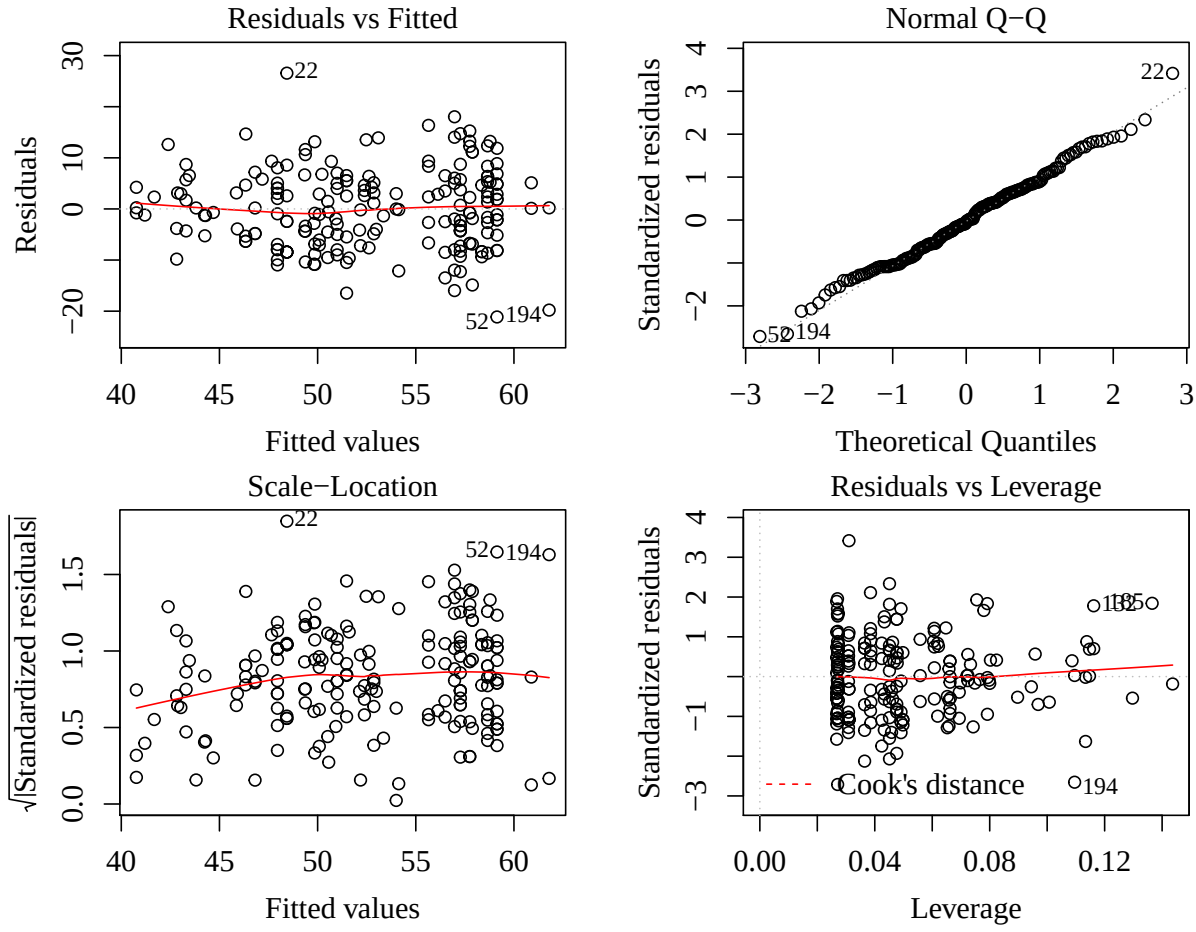
```
drop1(mainmod, test = 'F')
```

```
## Single term deletions
##
## Model:
## math ~ gender + race + ses + schtyp + prog
```

```
##        Df Sum of Sq   RSS AIC F value  Pr(>F)
## <none>              11857 836
## gender  1        11 11868 835    0.18 0.67568
## race    3      1060 12917 848    5.66 0.00098 ***
## ses     2       201 12057 836    1.61 0.20314
## schtyp  1        15 11872 835    0.24 0.62629
## prog    2      2946 14803 877   23.60   7e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Only `race` and `prog` are significant. The same holds true if we use the model with all interaction terms. Testing, for instance, `race` amounts to performing an ANOVA with `lmod` and a nested model missing `race` and all interaction terms with `race`.

(f)
```
plot(mainmod)
```



The diagnostics look good. There is slight heteroscedasticity and deviation from normality. A Box–Cox transformation may be useful.

Chapter 17

Experiments with Blocks

17.1 Randomized Block Design

17.2 Latin Squares

17.3 Balanced Incomplete Block Design

Exercises

Exercise 8: Sudoku as a Latin square? Standard Sudoku solutions are subsets of possible 9×9 Latin squares. The entries $1, 2, \ldots, 9$ are the nine 'treatments', with two blocking variables along the rows and columns. Sudoku solutions have the additional constraint that divides the square into nine 3×3 subsquares which must contain all nine treatments. This allows for a third blocking variable. For instance, suppose we are interested in 9 crop varieties represented by the Sudoku solution entries. The row block could represent 9 different farming companies, the column block could represent 9 different fertilizers, and the additional subsquare constraint could represent 9 different plots of land.